Doctoral thesis submitted to
the Faculty of Behavioural and Cultural Studies
Heidelberg University
in partial fulfillment of the requirements of the degree of
Doctor of Philosophy (Dr. phil.)
in Psychology

Title of the publication-based thesis

Deep Learning Architectures for Amortized Bayesian Inference in Cognitive Modeling

presented by
Stefan T. Radev, M.Sc.

year of submission
2021

Dean: Prof. Dr. Dirk Hagemann
Advisor: Prof. Dr. Andreas Voss

Contents

Acknowledgements

The intellectual wanderings transcribed in this thesis would not have been possible without the invaluable support and contributions of many people. First, I would like to thank my supervisor Andreas Voss for providing me with the unbounded freedom to choose and pursue my ideas and whims; for bearing with me and my diverging questions from the very start of my Masters'; for the indispensable discussions and for keeping an open mind for the future of our discipline.

Two other persons played a major supervisory role throughout my PhD life. Ullrich Köthe, whom I met during my computer science studies and whose teaching style and attitude towards students (and scientific life in general) I wholeheartedly admire, was always there for me (sometimes up at night) to discuss intellectual stuff, lift me up from reviewer and journal strikes, involve me in all sorts of crazy projects, and, most importantly, make me aware of my blind spots as a person and researcher. Paul-Christian Bürkner, whose workshop in Mannheim I accidentally attended, just as accidentally became my informal advisor in Bayesian statistics. During many discussions, he was able to clear up a whole lot of murkiness from my ideas, help me overcome a great deal of unknowledge, and, last but not least, repeatedly show me the need for having a Twitter account as a scientist (I still don't have one though, holding up...).

Perhaps I would not have commenced my PhD without the key influence of two very special psychologists. During my Masters', Ulf Mertens (together with the bold Ivan Marević) got me into deep learning and helped me discover my passion for programming languages, abstract problems, and long coding nights. Moreover, he and his wife, Alica Mertens, became good friends, beverage sharers, and idea tinkerers (they invited me to my first German wedding as well!). Also during my Masters', Veronika Lerche motivated me to fly to my first ever scientific conference in Coventry, where I, as an overdressed student assistant, presented a funny-looking graphical software to a bunch of stern researchers. Back then, she drew my attention to the bright sides of academia and got me thinking about staying for a bit longer than originally intended (unfortunately, we never started our very promising rock band).

I am thankful to my colleagues-turned-friends who accompanied me through this stage of life: Annika Stump, who represents the non-linearity of a creative life, who basically defined the taste of whiskey and always has an open ear for me; Marco D'Alessandro, the all-round Italian genius with an unceasing well of ideas in his pocket; Mischa von Krause, the master of work-life-balance; and Lukas Schumacher, my fellow investor, who reminded me of the unceasing enthusiasm and curiosity a scientist has to carry in his or her heart. Extra credits go to my friend-turned-colleague, Maximilian Knapp, and his amazing family.

In one way or another, I am indebted to all my international friends, collaborators, student assistants, and the entire SMiP community (especially David Izydorczyk and Martin Schnürch, who know how and what to drink during workshops and conferences).

At a whole different level, I am deeply grateful to my loving family, who never really let me feel away, and especially to my mother, who bravely struggled through all my boring papers and did not miss a single error.

Last but not least, this whole journey would have been unthinkable without Karin Prillinger, who was a bit of everything and everywhere.

1 Introduction

What I cannot create, I do not understand.
— Richard Feynman

Mathematical models are becoming increasingly important for describing, explaining, and predicting human behavior in terms of underlying mechanisms and systems of mechanisms. Although the ontology of such mechanisms remains largely unknown, their epistemic value and inferential power are now widely acknowledged throughout the behavioral sciences. Broadly speaking, whenever an assumed mechanism transforms information into behavior, it is referred to as a *cognitive process*. Cognitive processes are the conceptual fabric used to fill the explanatory gap between the mysterious firing of neurons and the mundane recognition of a long-forgotten acquaintance in the morning train. Consequently, modelers of cognitive processes earn their livelihood in an attempt to make the "ghost in a machine" tractable by replacing the ghost with *hidden parameters* embedded in an abstract functional framework.

The purpose of such parametric models is twofold. On the one hand, they can be viewed as formal expedients for understanding the messy and noisy human data in much the same way as the models physicists employ to make sense of the data coming from spiral galaxies and interstellar clouds. On the other hand, parametric models can be viewed as *behavioral simulators* and used to mimic the output of cognitive processes by generating synthetic behavior. Interestingly, there is a strange asymmetry in the challenges surrounding these two goals. Simulating behavior requires only specifying a cognitive model as a computer program and running the program with a desired parameter configuration. It is thus a generative process mainly constrained by the creativity and imagination of individual modelers. Differently, reverse engineering human data to recover hidden parameters is hampered by two external factors: the resolution and abundance of data and the availability of universal and efficient inferential methods. As for the latter, behavioral scientists have often sacrificed fidelity and complexity in order to adjust their models not to reality but to the limitations of existing inferential methods. Such a strategy is definitely viable in the early (often linear and beguilingly clear) stages of scientific inquiry, but it does not live up to the challenges and questions posed by later (often non-linear and disconcertingly fuzzy) stages.

The main argument of this thesis is that *questions of inferential tractability are of secondary importance for enhancing our understanding of the processes under study*. Accordingly, the core purpose of this thesis is to develop frameworks which leave such questions to specialized "black-box" artificial neural networks and enable researchers to focus on developing and validating faithful "white-box" models of cognition. Instead of a ready-made solution, the thesis explores a beginning of a solution. It presents a potentially fruitful coupling between human and artificial intelligence, an approach which is expected to gain more and more momentum as the world fills with artificial agents. Ultimately, this thesis strives to increase creativity by embracing complexity.

1.1 Motivation and Scope

This thesis is motivated by the question of how to offload *parameter estimation* and *model comparison* onto specialized neural network architectures. Undoubtedly, parameter estimation and model comparison are two of the most common and most challenging tasks in model-based behavioral sciences. Accordingly, for each of the two tasks, we will offer a general framework for composing neural networks capable of bootstrapping a wide range of inference tasks. The main principle will be to utilize prior domain expertise and guide these networks through simulations to become "experts" in inferring hidden parameters from data or selecting between plausible models of the data. The proposed frameworks are themselves embedded into the meta-framework of Bayesian inference which embraces probability theory as *the logic of science* [77].

Why probability theory? Simply put, a probabilistic approach to inference is appealing, as it provides consistent equations and principled methods for quantifying and communicating uncertainty [22]. Correspondingly, doing Bayesian inference and data analysis is nothing but applying the basic rules[1] of probability theory to amount of information gained through empirical inquiry. Curiously, it is primarily in the behavioral sciences that researchers applying probability theory are given the cultist label *Bayesians* and often seen as representatives of a statistical opposition against traditional (the cultist label being *frequentist*) methods. Accordingly, whenever the reader encounters the term Bayesian in this thesis, it should be read as *using the rules of probability theory to express uncertainty, update beliefs, revise knowledge, and inform scientific conclusions*.

Probabilistic reasoning is hard and time-consuming. It was not until general-purpose computers had shrunk considerably in size that Bayesian inference became useful for handling non-trivial practical problems. Until then, practitioners could leverage only a limited subset of the tools probability theory had to offer. Moreover, this restricted inference was further constrained by the ability to solve complicated integrals, or, as David MacKay puts it: "...a macho activity enjoyed by those who are fluent in definite integration" [103, p. 319].

With the advent of high-performance computing[2], Monte Carlo methods came to the rescue of probabilistic inference, the most prominent algorithmic family being Markov chain Monte Carlo (MCMC) methods [109]. Initially, MCMC proved instrumental in approximating the unimaginable integrals which had been thwarting the solution of relevant problems in chemical physics [141]. Today, the heirs of those rather crude grandfather sampling methods have become the Bayesian gold standard across the sciences, with novel and interesting modifications spawning in scientific journals on a regular basis. The behavioral sciences are no exception to this trend, being a field of both active application and development of novel Bayesian methods.

Notwithstanding the major contributions of MCMC methods to large-scale inference problems, they remain notoriously slow and sequential in nature [12]. These drawbacks can render inference with highly complex models practically infeasible, but they also transfer to applications of relatively simple models to big data where relevant computations need to be repeated for each observation. Consequently, it is not uncommon for researchers to wait a week or two for estimation algorithms to finish, only to notice afterwards that critical adjustments to the algorithm or the

[1] Even though the rules of probability theory themselves are basic and intuitive, the application of these rules in practice can be anything but basic.

[2] The term *high-performance* having, of course, only a temporal meaning within the context of a computing generation.

model necessitate rerunning the entire loop from scratch. The inferential matters become even worse, when the model itself cannot be specified in a nicely closed, analytic form, but is only available as a Monte Carlo simulator program. The latter can often leave potentially relevant modeling territories vastly unexplored and confine certain model classes to a purely Platonic playground. Our goal is to provide an efficient and scalable framework for designing and testing solutions to precisely those challenging situations encountered frequently by behavioral scientists. Moreover, we believe that our ideas can positively impact computational modeling in research areas not solely confined to the behavioral sciences. At a high level, our framework is purely simulation-based and leverages the representational power of deep learning methods to build reusable estimators for two of the most important constructs in Bayesian inference: the *posterior distribution* and the *evidence*. Further, it utilizes the concept of *amortized inference* to increase the inferential efficiency of these estimators at every modeling step, from model development, to model selection. Importantly, our framework includes methods for self-diagnosis of miscalibrated inference due to algorithmic errors, which is an essential precondition for computational faithfulness in any Bayesian data analysis pipeline.

Beyond the development of a novel Bayesian framework for simulation-based inference, this thesis presents some concrete applications to relevant research questions in cognitive modeling and beyond. These applications demonstrate the utility of the framework for tackling challenging cognitive models dealing with both simple (typically independent and identically distributed, or *i.i.d.*) and more complex probabilistic structure (typically exhibiting temporal dependencies, or non-*i.i.d.*). Moreover, the models considered in these applications are themselves novel and serve the purpose to inspire further research and exploration in the corresponding areas.

Finally, the thesis includes a starter Python library for building own estimation or model comparison networks with minimal programming skills.

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- Chapter 3 provides the necessary background and conceptual machinery for understanding uncertainty quantification in the context of Bayesian inference. It then discusses the challenges faced by standard Bayesian methods when applied to complex models and how these affect the field of cognitive modeling in particular. We introduce the concepts of simulation-based and amortized inference with neural samplers and set the stage for presenting our BayesFlow framework.

- Chapter 5 introduces our general BayesFlow framework for solving the task of amortized Bayesian parameter estimation. We demonstrate how to perform inference on data sets with different sizes and probabilistic structure by using specialized network architectures which preserve the probabilistic symmetry of the target Bayesian posterior. We formally derive a training procedure which ensures that neural networks in our framework recover the true target posteriors under perfect convergence of the optimization algorithm. We end the chapter with a simulation-study demonstrating the utility of our method.

- Chapter 6 introduces our *Dirichlet* evidence network for solving the task of amortized Bayesian model comparison. We explore a method to quantify absolute evidence as compared to relative evidence through a specific form of regularization in a meta-probabilistic framework. As in chapter 5, we show how to deal with variable numbers of observations and different model/data types. We also derive a simulation-based training method which ensures that evidential networks in our framework recover the true model probabilities under perfect convergence of the optimization algorithm. We end the chapter with two simulation studies using complex computational models from cognitive science and neuroscience.

- Chapter 7 introduces a visionary approach towards *meta-amortized inference*. It combines both parameter estimation and model comparison into a single unifying framework and presents initial conceptual results.

- Chapter 8 presents applications of the proposed Bayesian frameworks for model-based inference on real data. It starts with a direct estimation of an information-theoretic model of adaptive performance inspired by the Bayesian Brain Theory (BBT). Then, we describe a parameter estimation study concerned with a set of novel models of decision making. This is followed by an application of a custom diffusion model to a massive data set of human response times to disentangle questions of cognitive aging. Finally, we present an application to Covid-19 outbreak modeling with a version of BayesFlow for dynamic models.

1.3 List of Scientific Publications of the Publication-Based Dissertation

The central ideas put forward in this thesis have been explored in the following publications by the author and his cooperators:

- S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe. “BayesFlow: Learning complex stochastic models with invertible neural networks”. *IEEE Transactions on Neural Networks and Learning Systems*, 2020, pp. 1–15. DOI: 10.1109/TNNLS.2020.3042395

- S. T. Radev, M. D’Alessandro, P.-C. Bürkner, U. K. Mertens, A. Voss, and U. Köthe. “Amortized Bayesian model comparison with evidential deep learning”, Manuscript submitted for publication

- S. T. Radev, A. Voss, E. M. Wieschen, and P.-C. Bürkner. “Amortized Bayesian inference for models of cognition”. *International Conference on Cognitive Modelling (ICCM) Conference Proceedings*, 2020

- M. D’Alessandro, S. T. Radev, A. Voss, and L. Lombardi. “A Bayesian brain model of adaptive behavior: an application to the Wisconsin Card Sorting Task”. *PeerJ* 8, 2020, e10316

The author also contributed to the following publications which are related to the core topics of the current thesis:

- M. von Krause, S. T. Radev, and A. Voss. "Processing speed is high until age 60: insights from Bayesian modeling in a one million sample (with a little help of deep learning)", Manuscript submitted for publication
- S. T. Radev, U. K. Mertens, A. Voss, and U. Köthe. "Towards end-to-end likelihood-free inference with convolutional neural networks". *British Journal of Mathematical and Statistical Psychology* 73:1, 2020, pp. 23–43
- E. M. Wieschen, A. Voss, and S. Radev. "Jumping to conclusion? a lévy flight model of decision making". *TQMP* 16:2, 2020, pp. 120–132
- S. T. Radev, F. Graw, S. Chen, N. Mutters, V. Eichel, T. Bärnighausen, and U. Köthe. "Model-based Bayesian inference of disease outbreak with invertible neural networks". *arXiv preprint arXiv:2010.00300*, 2020
- S. Bieringer, A. Butter, T. Heimel, S. Höche, U. Köthe, T. Plehn, and S. T. Radev. "Measuring QCD splittings with invertible networks". *arXiv preprint arXiv:2012.09873*, 2020
- L. Konicar, S. Radev, K. Prillinger, M. Klöbl, R. Diehm, N. Birbaumer, R. Lanzenberger, P. Plener, and L. Poustka. "Volitional modification of brain activity in adolescents with Autism Spectrum Disorder: A Bayesian analysis of Slow Cortical Potential neurofeedback". *NeuroImage: Clinical*, 2021, p. 102557
- U. K. Mertens, A. Voss, and S. Radev. "ABrox—A user-friendly Python module for approximate Bayesian computation with a focus on model comparison". *PloS one* 13:3, 2018, e0193981

1.4 Notes on Notation

Throughout this thesis, we will follow some simple conventions for consistent mathematical notation. We will denote scalar variables by lowercase italic, e.g., x, y, z, vectors by lowercase bold italic, e.g., $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, and matrices by uppercase bold italic letters, e.g., $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$. Data sets comprising multiple observations (e.g., multivariate responses of a single participant to a particular task) will be denoted as $\{\boldsymbol{x}_n\}_{n=1}^N = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\} \equiv \boldsymbol{x}_{1:N}$, where N indicates the number of observations. Occasionally, and when possible, we might stack all observations comprising a data set row-wise into a matrix, $\{\boldsymbol{x}_n\}_{n=1}^N \equiv \boldsymbol{X}$. Whenever the observations are assumed to be *time-dependent*, we will use T to denote the total number of observations in the resulting (multivariate) time-series $\boldsymbol{x}_{1:T}$. Occasionally, we will include the superscript *obs* to denote an actually observed data set, in contrast to a simulated one (i.e., $\boldsymbol{x}_{1:N}^{(obs)}$ vs. $\boldsymbol{x}_{1:N}$).

We will always collect the parameters of a mathematical model into a vector $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_D)$, and reserve the letter D for the dimensions of the parameter space. Finally, we will collectively refer to all trainable parameters of a neural network (e.g., weight matrices, biases, activation function parameters) as a vector (e.g., $\boldsymbol{\phi}, \boldsymbol{\psi}, ...$) even though these might be distributed across different functional components or layers of the network. Importantly, neural network parameters are *not to be confused* with the parameters of the mathematical model of interest, as the former are uninterpretable and high-dimensional, whereas the latter are carriers of theoretical value and usually low-dimensional.

Table 1.1: Table of important symbols and their corresponding description

Notation (Symbol)	Meaning
$\boldsymbol{x}, \boldsymbol{x}_{1:N}$	observed or simulated data point, data set
$\boldsymbol{\theta}, \boldsymbol{\omega}$	parameters of a mathematical model (simulator)
$\boldsymbol{z}$	latent variable learned by a (deep) generative model
$\mathcal{X}^N, \Theta, \Xi$	data space, parameter space, noise space
$\mathcal{M}, \mathcal{M}_j$	candidate model set, model index
g	generative (forward) model / simulator
p, q	probability density (mass) functions
$\boldsymbol{\phi}, \boldsymbol{\psi}$	trainable parameters of neural networks
$f_{\boldsymbol{\phi}}, h_{\boldsymbol{\psi}}$	functions parameterized via neural networks
$\mathbb{E}[\cdot]$	expected value of a random variable (vector)
$\mathbb{KL}[p \,\|\|\, q]$	Kullback-Leibler divergence between densities p and q
N	number of observations in a (simulated) data set
D	number of parameters / dimensions of the parameter space
B	number of simulations per training step / batch size

For the most part of this thesis, we will be concerned with (absolutely) continuous random vectors and their associated probability density functions (pdfs). For the sake of readability, the latter will be denoted by p even when they refer to pdfs of different random vectors defined on different spaces, which will be clear from the function arguments. For instance, we will write $p(\boldsymbol{\theta})$ for the (prior) probability density of the parameter vector $\boldsymbol{\theta} \in \Theta$ instead of $p_\Theta(\boldsymbol{\theta})$. Additionally, each pdf of interest will be implicitly associated with a corresponding probability measure P. Throughout the text, we will use density and distribution interchangeably.

By means of a slight abuse of notation, when a density function is approximated via a neural network with trainable parameters $\boldsymbol{\phi}$, we will often write $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \equiv q(\boldsymbol{\theta} \mid \boldsymbol{\phi})$, or, for a conditional density, $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x}) \equiv q(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{\phi})$. In this way, (i) we align our notation to the predominant notation in the literature on deep generative modeling; (ii) implicitly denote the dependence of the approximate density on the neural network parameters; (iii) make it immediately clear which is the density being approximated, e.g., $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})$ approximates $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ by means of neural network parameters $\boldsymbol{\phi}$.

The most important symbols and notation used throughout the text are summarized in Table 1.1.

2 Models of Cognition

Murky thoughts, like murky waters, can serve two purposes only: to hide what lies beneath, which is our ignorance, or to make the shallow seem deep.
— Giulio Tononi

Reasoning with models of empirical phenomena lies at the very heart of science. Models abstract away irrelevant details and focus attention on the theoretically relevant aspects of complex systems. Ideally, they simplify, but do not oversimplify reality. The importance of models for scientific progress is twofold. On the one hand, theories can be systematically instantiated and tested by specifying a mathematical model and inferring its hidden properties from data. On the other hand, competing theories can be tested against one another via formal model comparison. Thus, model-based reasoning complements verbal reasoning insofar as it reduces ambiguity and translates "murky" statements into precise and directly quantifiable hypotheses. Whether the latent properties of cognitive models represent faithful descriptors of the *unobservable causes* of behavior remains an open question whose surface we will only scratch here. The main purpose of this chapter is to establish the notion of a cognitive model, fix a useful notation, and introduce the concept of a likelihood function.

2.1 Cognition and Computation

Cognitive models exist to help cognitive scientists make sense of observed behavioral data in terms of unobservable (latent) cognitive processes, such as attention, memory decay, evidence accumulation, or belief updating, to name just a few [45]. Such models have been around for centuries (see Figure 2.1), with the most notable modern twists being a shift in contextualization (i.e., in the reference theoretical framework) and an increase in mathematical formalism. Whether cognitive processes actually exist as functional entities or simply represent useful metaphors psychologists live by, is a question that currently extends from philosophy down to single-cell neuroscience.

In the year 1994, Francis Crick, essentially reinventing naturalism, formulated his *astonishing hypothesis* in a rather flowery way:

> The Astonishing Hypothesis is that "You", your joys and your sorrows, your memories and your ambitions, your sense of personal identity and free will, are in fact no more than the behavior of a vast assembly of nerve cells and their associated molecules. As Lewis Carroll's Alice might have phrased it: "You're nothing but a pack of neurons". [28, p. 3]

At the time of writing, hardly any cognitive scientist would find the relationship between brain, cognition, and behavior alien or astonishing. In fact, this relationship is the explicit or implicit

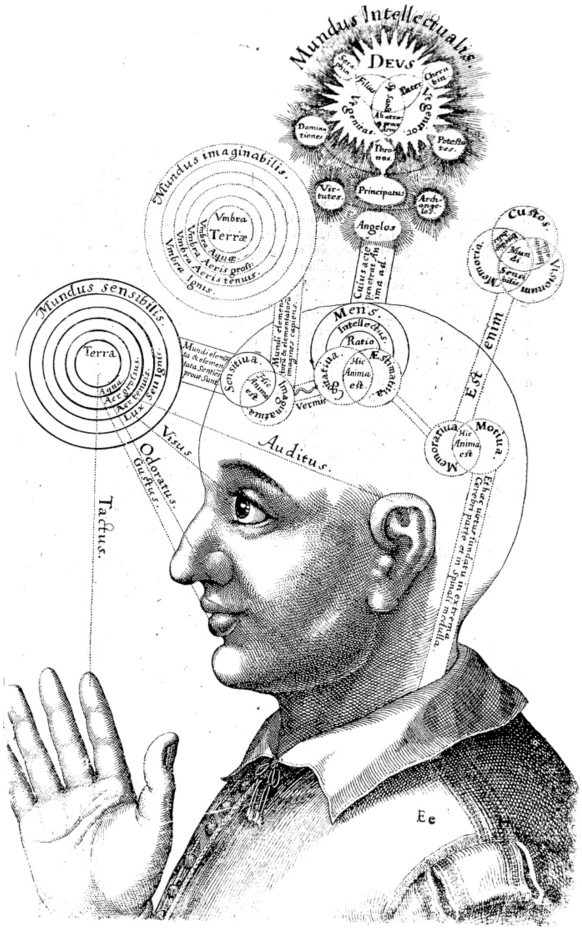

Figure 2.1: Robert Fludd's "model" of a mind in the world from 1619 [15]. The diagram can be seen as an early predecessor of modern cognitive architectures incorporating cognitive functions such as perception, memory as well as an explicit flow of information between the mind and the world. Cognitive functions are localized in the head, even though neurons are yet to be discovered.

working hypothesis behind some of the most prominent cognitive architectures and model classes [91, 153, 162]. However, few would contend that even a complete description of all microtubules in each and every neuron in the brain will ever be sufficient to explain why a grandmaster under time pressure failed to spot an obvious checkmate in a crucial game of chess. Invoking such an explanation appears to require more than just a description of the behavior of molecules and neurons. In other words, the "no more" and the "nothing but" parts of the astonishing hypothesis are what provokes a certain theoretical dissatisfaction. Perhaps going up to a complete understanding of neural firing patterns would resolve the dissatisfaction. But how far up should one climb the reductionist ladder until a satisfactory level of analysis for cognition is reached? This is an example of the so called *bottom-up* problem, starting from the small and tractable constituents of a presumably complex process and building *up* towards its synthesis.

In contrast, one may start with a handy catalogue of cognitive processes inherited from the dictionary of psychology and look *down* for the "neural correlates" of these processes. Necessarily, such an approach rests on the assumption that there will be a serendipitous one-to-one correspondence between cognitive and neural processes. This is an example of the *top-down* problem, starting from the processes and analysing them in terms of their constituents.

Undoubtedly, both approaches bear obvious risks. Following a purely bottom-up research agenda, we might run into the problem of being unable to reconstruct the entire list of assumed cognitive processes. Following a purely top-down approach, we might not end up discovering the neatly corresponding constituents we are looking for, leaving us with a list of substance-free metaphors.

It is perhaps no coincidence, that neuroscientists favor a predominantly bottom-up approach. For instance, György Buzsáki [18] outlines the three missing pieces for a purportedly complete understanding of how the brain generates behavior. These are the understanding of (i) the dynamical structural organization of the brain; (ii) the physiological functions of its constituents; (iii) and the computational mode of operation that enables the neurons in a given anatomical hardware to execute actions [18, p. 24]. The eventual synthesis of this knowledge is expected to provide a satisfactory explanation for all kinds of behavior in terms of underlying neural mechanisms. In the process of studying and systematizing these mechanisms, only the neurophysiologically plausible cognitive constructs would therefore stand the test of rigorous empirical verification.

Cognitive scientists, on the other hand, prefer Marr's interpretative framework for talking about cognitive processes [106]. Its starting point is the basic concept of *information processing*, that is, the detectable (i.e., the difference that makes a difference) transformation of information (e.g., wavelengths becoming cone cell responses or the ringing of the phone becoming an increase in heart-rate variability). It then distinguishes three levels of analysis for understanding any information processing system: (i) *computational theory* - concerning the goals (the why) of the system and its computational logic; (ii) *representation and algorithm* - concerning the representation of input and output as well as the step-by-step instructions for carrying out the input-output transformation (the how); (iii) *hardware implementation* - concerning the physical realization of the algorithm [106, p. 25].

Accordingly, we can equate cognitive processes with the algorithms transforming neural representations into observed behavior (or into further representations), without committing to a particular physical ontology. Thus, computational models of cognition (cognitive models, for short) are our best instruments to learn something about these algorithms from behavioral data alone, without resorting to the expensive methods of neuroscience. Ideally, the functional form and parameters of cognitive models would capture the most important algorithmic and representational characteristics of the system under study. At the same time, the assumed hardware underpinnings of a cognitive process should impose a number of parametric and functional constraints (e.g., the maximal speed of information processing), if a model of the process is to be taken seriously by substantive neuroscientists.

Cognitive models are occasionally termed *computational models*, highlighting their algorithmic essence and their conceptual relatedness with notions borrowed from computer science. Even though neurophysiological plausibility seems to be a prerequisite for the ultimate validation of any cognitive model, it is often ignored in favor of a strong embedding in the nomological nexus of psychology. Accordingly, as long as the parameters of a cognitive model are interpretable with a reference to an overarching psychological theory, the actual neuroanatomical hardware becomes of secondary importance. In addition, neurophysiological plausibility becomes even less important if the presumed cognitive processes are eventually to be implemented by goal-directed artificial agents. In the latter case, an artificial agent may perfectly mimic human or animal behavior without the need or physical possibility to invoke any neural representations. As a consequence, the concept of a cognitive process may become decoupled from its neural realization and thus become synonymous to an algorithm, which is per definition independent of its implementation. Nevertheless, for those striving to infer something about real brains from real behavioral data, the hope remains that their models are at least able to tap into the distant echo of neural system parameters within a coherent "neurocognitive" framework.

Future theoretical developments may also see *phenomenological plausibility* come as an additional criterion for models representing cognitive processes which are *experienced* in some way (e.g., the experience of coming to a decision as compared to all non-experienced factors that contributed to the decision). However, the fact that information processing is experienced in a particular way seems less important for cognitive modelers than the functional task of describing how information is transformed in a way that ultimately leads to manifest behavior. And even though simulating experience appears to be unimaginably hard (except perhaps for the brain), we can easily "build" abstract machines which simulate behavior in various experimental and observational contexts by executing a series of well-defined computational steps. It is this property which makes model building a creative process and cognitive models *generative* in nature.

2.2 Behavioral Simulators

Formally, we can represent a cognitive model as a function $g : \Theta \times \Xi \rightarrow \mathcal{X}$ which generates observable quantities $\boldsymbol{x} \in \mathcal{X}$ from a particular configuration of the hidden parameters $\boldsymbol{\theta} \in \Theta$ and an independent source of noise $\boldsymbol{\xi} \in \Xi$. The function is typically realized as a Monte Carlo computer simulation which mimics quantifiable manifestations of actual behavior. Simulation programs involving random number generation are also known as Monte Carlo engines or probabilistic programs.

Since humans rarely behave the same way even when provided with the same information, the stochastic component $\boldsymbol{\xi}$ in the model formulation ensures that g is non-deterministic given the same time-invariant parameter configuration $\boldsymbol{\theta}$. Ideally, the noise in a cognitive model should capture all non-modeled factors which nevertheless influence the generation of behavior, but it may also reflect inherent randomness of the cognitive system under study. In the latter case, the noise distribution might itself contain learnable parameters which are part of $\boldsymbol{\theta}$, so it should be denoted as $p(\boldsymbol{\xi} \mid \boldsymbol{\theta})$. Thus, the general functional form of a (stochastic) cognitive model is:

$$\boldsymbol{x}_n = g(\boldsymbol{\theta}, \boldsymbol{\xi}_n) \quad \text{with} \quad \boldsymbol{\xi}_n \sim p^*(\boldsymbol{\xi}) \tag{2.1}$$

where the subscript n indicates that the simulator can be run repeatedly with the same parameter vector to yield an entire data set $\{\boldsymbol{x}_n\}_{n=1}^N \in \mathcal{X}^N$ and p^* denotes the true underlying noise distribution. Usually, this distribution is unknown and its form needs to be either theoretically deduced or approximated *ad hoc* via a simpler distribution p from which random samples can easily be obtained (e.g., Gaussian, Poisson).

The model form in Equation 2.1 represents a *memoryless* or a stateless process: each run of the simulator with a fixed parameter combination is independent of the previous runs and does not influence future runs. As we will show later, such models generate data sets consisting of independent and identically distributed (*iid*) observations. More complex models involving some form of memory can be formulated by a recursive dependence of the simulator on the previous observation:

$$\boldsymbol{x}_n = g(\boldsymbol{x}_{n-1}, \boldsymbol{\theta}, \boldsymbol{\xi}_n) \quad \text{with} \quad \boldsymbol{\xi}_n \sim p^*(\boldsymbol{\xi}) \tag{2.2}$$

or by introducing a persistent memory variable $\boldsymbol{\omega}_n$ which is updated after each simulation run and may itself be unobserved and treated as a time-varying parameter:

$$(\boldsymbol{x}_n, \boldsymbol{\omega}_n) = g(\boldsymbol{\omega}_{n-1}, \boldsymbol{\theta}, \boldsymbol{\xi}_n) \quad \text{with} \quad \boldsymbol{\xi}_n \sim p^*(\boldsymbol{\xi}) \tag{2.3}$$

Importantly, such *stateful* models give rise to more complex sets of observations which are nolonger *i.i.d.* and thus need to be tackled differently than *i.i.d.* observations. Accordingly, a major aim of this thesis is to develop and validate a framework for performing model-based inference on both *i.i.d.* and non-*i.i.d.* observations. In anticipation of future modeling challenges including joint models of neural and behavioral data as well as more unstructured data such as graphs, text, or motion time-series, we will ensure that our framework is extendable to incorporate these future challenges. We further anticipate that our ideas will be potentially useful in different fields having embraced model-based reasoning and inference.

Scientifically useful cognitive models should work both forward and backward. Given a set of parameters, researchers should be able to generate artificial observations and data sets which are indistinguishable from their real counterparts even when scrutinized by expert observers or subjected to rigorous statistical tests. Conversely, given only a set of observations, researchers should be able to recover the hidden data-generating parameters which are seen as epistemically valuable *explanans* of the data. We call this the *inverse inference* problem.

As already briefly mentioned in the introduction, the two tasks are notably asymmetric with respect to the computational and epistemic burden they carry. To further appreciate this asymmetry, consider the example of an ice cube left at room temperature to eventually become a puddle of water [154, p. 196]. Having a model of the process of fusion, one can feasibly simulate how an arbitrary ice cube transitions into a puddle over time (forward inference). However, suppose that you are presented with only a puddle and tasked to recover an unobserved cube (inverse inference). Even though you can infer something about the volume or the density of the ice cube, the task as a whole appears insurmountable, since there are different cubes that could have melted into the same puddle, which presents a case of inherent non-identifiability. What is more, there is uncertainty about the model of the ice cube itself, since there is a multitude of ice forms, not necessarily cubic, that could have resulted in the same puddle.

Researchers striving to understand and model how "the mind can occur in a physical universe" [2] face a similarly tough challenge. When it comes to modeling cognition and behavior, this challenge is further aggravated by the possibility that *there could have been no ice cube to begin with*, but an entirely different puddle-generating process.

In short, forward inference is easy, whereas inverse inference is hard. Forward inference requires "only" the ability to write down the model as a simulator program in a programming language and run the simulation with the desired set of parameters. It also requires creativity and theoretical insight (as well as acceptance by the community of other modelers). Inverse inference, on the other hand, requires invoking additional estimation procedures, largely external to the model or the modelers themselves. Further, it implies facing a large portion of epistemic uncertainty. In addition, estimation procedures are impeded by the following properties of complex simulators:

1. The simulator is typically non-deterministic, so that there is intrinsic uncertainty about the true value of $\boldsymbol{\theta}$.

2. The simulator is typically not information-preserving, so that there is ambiguity among possible values of $\boldsymbol{\theta}$.

3. The simulator is typically too complex to admit a closed-form mathematical expression for evaluating the likelihood of $\boldsymbol{\theta}$.

A multitude of inference frameworks with a varying degree of generality have been proposed for addressing problems of inverse inference. The most prominent among these are manual tuning, parameter search methods, kernel methods, optimization methods, maximum likelihood, maximum-a-posteriori (MAP inference), variational inference, fully Bayesian inference, approximate Bayesian computation (ABC), machine learning approaches as well as various hybrid methods [12, 29, 52, 105, 123, 136, 140, 142, 150]. A comprehensive review of the multiverse of methods for inverse inference is beyond the scope of this work (but see Chapter 4 or the excellent review by [27] for a broad classification). Essentially, all approaches for inverse inference optimize a trade-off between efficiency, scalability, accuracy, and practical utility. As there is no free lunch in capital markets, there is no silver bullet in statistical inference. Put differently, there is no method or framework that simultaneously maximizes all of the above criteria and the *best method* will be application-dependent.

This thesis focuses on scaling and utilizing fully Bayesian inference for very complex (often deemed intractable) models exhibiting all three inferential predicaments. In the next chapter, we will see why such models necessitate a *simulation-based* approach to Bayesian inference. Before we conclude this chapter, however, a word on the concept of *likelihood* seems warranted.

2.3 The Likelihood

When looked upon through a probabilistic lens, the outputs of a cognitive model can be associated with some (potentially very complex) probability distribution. This distribution is referred to as the *likelihood* and denoted as $p(\boldsymbol{x} \mid \boldsymbol{\theta})$. Loosely speaking, when evaluated, the likelihood returns the relative probability of an observation $\boldsymbol{x}$ given a set of parameters $\boldsymbol{\theta}$. When the parameters are systematically varied, the likelihood can be used to quantify how well each model instantiation fits the data.

If the likelihood has a known distributional form (e.g., Gaussian, Laplace, Dirichlet), the model in Equation 2.1 can be formulated entirely in terms of the likelihood. Moreover, in these simple cases, the likelihood can be evaluated analytically or numerically for any pair $(\boldsymbol{x}, \boldsymbol{\theta})$. In a sense, all stochastic models have a *dual generative representation*:

$$\boldsymbol{x}_n \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}) \quad \Longleftrightarrow \quad \boldsymbol{x}_n = g(\boldsymbol{\theta}, \boldsymbol{\xi}_n) \;\; \text{with} \;\; \boldsymbol{\xi}_n \sim p(\boldsymbol{\xi}) \tag{2.4}$$

where the likelihood function $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, being a probability distribution itself, naturally captures the effects of the extrinsic stochastic factor $\boldsymbol{\xi}$. Note, that this representation can be trivially extended to incorporate models with memory (Equations 2.2 and 2.3). In fact, each stochastic model viewed as a Monte Carlo simulator defines an implicit likelihood given by the relationship:

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \int_{\Xi} \delta(\boldsymbol{x} - g(\boldsymbol{\theta}, \boldsymbol{\xi}))\, p(\boldsymbol{\xi} \mid \boldsymbol{\theta})\, d\boldsymbol{\xi} \tag{2.5}$$

where $\delta(\cdot)$ is the Dirac delta function and the integral runs over all possible execution paths of the stochastic simulation for a fixed $\boldsymbol{\theta}$. For most complex models, this integral is analytically intractable or too expensive to approximate numerically, so it is much easier to specify the model directly in terms of the simulation program g instead of using the likelihood. However, according to Equation 2.4, we can still *sample* from the likelihood by running the simulator with different Monte Carlo realizations of $\boldsymbol{\xi}$.

It is at this point where likelihood-based and simulation-based inference methods diverge. The former require the ability to evaluate the likelihood for any pair $(\boldsymbol{x}, \boldsymbol{\theta})$. The latter require only the ability to sample (simulate) arbitrary pairs $(\boldsymbol{x}, \boldsymbol{\theta})$ from the likelihood[1]. In the next chapter, we will see how the inability to evaluate or derive the likelihood prohibits standard Bayesian methods. We will then briefly peruse the frontier of simulation-based inference before introducing our frameworks for parameter estimation and model comparison.

[1] Although the literature has adopted the term *likelihood-free* inference, we will completely avoid it in this text, since it incorrectly implies the absence of a likelihood, even though the likelihood is implicitly defined via the action of the simulation program. It is the calculation of its actual numerical value for simulated or real observations which is impossible. We will therefore prefer the term *simulation-based* inference, since it unambiguously captures the essence of the task.

3 Bayesian Inference

It is an extraordinary feature of science that the most diverse, seemingly unrelated, phenomena can be described with the same mathematical tools.
— Benoit Mandelbrot

Probability theory is the language for describing uncertainty with mathematical objects. Uncertainty is not merely a philosophical phantasm or a convenient excuse for the erroneous forecasts of meteorologists, but rather a fundamental epistemic basis for decision making in an incomplete universe. Scientists face uncertainty both in their academic and private lives. Finite data, approximation errors, noisy measurements, and inherently stochastic models are the common culprits for cultivating the habit of reporting some confidence measures alongside point estimates. But also everyday questions such as "Will it rain tomorrow?", "Which party is likely to win the election?", or "What is the likelihood of encountering a dragon in the park tonight?" call for reasoning with a varying degree of confidence. Since the core topic of this thesis is model-based inference, we will see in this chapter how Bayesian methods provide a self-consistent framework for uncertainty quantification and communication when trying to extract cognitive models from behavioral data. Apart from the parlor of behavioral sciences, Bayesian methods have been employed for tasks as diverse as predicting global equity indices [73], inferring latent infectious disease dynamics [42], and evaluating forensic DNA profiles [9]. Henceforth, we will assume the utility of Bayesian methods as given and proceed to the conceptual and mathematical details of Bayesian inference.

3.1 From Prior to Posterior

The core mathematical workhorse in Bayesian probabilistic reasoning is the famous Bayes' rule [52], which specifies how to update prior knowledge about a given quantity upon making an informative observation. In cognitive modeling, the quantities of interest are the parameters of a cognitive model, which capture relevant computational characteristics of the process under scientific scrutiny. Thus, when collecting behavioral data, we ultimately strive to increase our knowledge about these parameters as a proxy for understanding the assumed cognitive processes.

The point of departure in Bayesian inference is the *prior distribution* (or just *prior*[1], for short), denoted as $p(\boldsymbol{\theta})$. Ideally, the prior is supposed to capture both what we already (believe to) know about the parameters, but also what we still do not (believe to) know. Knowledge is expressed through a reasonable domain and concentration of probability density (i.e., our "best" guess). Lack of knowledge is expressed through the spread of probability density over the domain (i.e.,

[1]Whenever a model has a multi-dimensional parameter space, one speaks of a *joint prior* highlighting the fact that the the distribution p extends over multiple parameters (dimensions). Correspondingly, one speaks of a *joint posterior* when referring to the updated distribution of multiple parameters.

our uncertainty about the "best" guess). To make this idea concrete, Figure 3.1 depicts three one-dimensional priors with the same average value of zero but different allocations of uncertainty and belief.

Even though the prior is typically defined as an unconditional density, this definition is merely an aesthetic consideration, since knowledge and uncertainty are fundamentally contextual. Thus, $p(\boldsymbol{\theta})$ is really a shorthand for $p(\boldsymbol{\theta} \mid C)$, where C is a placeholder for all contextual information, such as previous research, theoretical constraints, model assumptions, and cultural preferences. Indeed, one of the greatest appeals (and, perhaps, at the same time, greatest deterrents) of Bayesian inference is that it allows to systematically and explicitly incorporate contextual information.

In a sense, Bayesian inference does away with the illusion of objectivity by allowing subjectivity to be expressed explicitly and transparently. Accordingly, one might question the specification of $p(\boldsymbol{\theta} \mid C)$ on empirical or theoretical grounds and propose $p(\boldsymbol{\theta} \mid C')$ as an alternative. In this case, the difference in contextual information leads to different epistemic states *prior* to observing any new data and potentially different conclusions *after* observing some data[2].

The process of knowledge updating in Bayesian inference essentially consists in transforming the prior into the *posterior* according to the well known Bayes' rule:

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{\theta}, \boldsymbol{x})}{\mathbb{E}_{p(\boldsymbol{\theta})}[p(\boldsymbol{x} \mid \boldsymbol{\theta})]} \tag{3.1}$$

where $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ denotes the likelihood function, as discussed in the last chapter, and $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ denotes the posterior given some observation $\boldsymbol{x}$. In the context of model-based inference, attaining the posterior corresponds to Bayesian parameter estimation and is sometimes referred to as *inverting the likelihood*, due to the reflection of the arguments across the $\mid$ symbol. The denominator $p(\boldsymbol{x})$ in the second term of Equation 3.1 is known as the *evidence* and represents a normalizing constant for the posterior. The equivalent denominator in the third term rephrases the evidence as the expectation of the likelihood with respect to the prior, that is, $\mathbb{E}_{p(\boldsymbol{\theta})}[p(\boldsymbol{x} \mid \boldsymbol{\theta})] = \int_{\Theta} p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$. The latter definition underlines the useful interpretation of the evidence as an *average* over all possible $\boldsymbol{\theta}$ or, equivalently, as a *marginal likelihood*. Even though the marginal likelihood is usually bypassed in parameter estimation tasks due to the obvious proportionality

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}), \tag{3.2}$$

it becomes a key object in the context of Bayesian model comparison.

Importantly, all distributions in Equation 3.1 are implicitly conditioned on a given generative model g, in addition to unspecified contextual information C. Looking at the computational definition of the posterior, another consequence of Bayesian inference stands out, namely, that different observations will lead to different updated epistemic states. Thus, observing $\boldsymbol{x}'$ will generally elicit a different change in knowledge, that is, $p(\boldsymbol{\theta} \mid \boldsymbol{x}) \neq p(\boldsymbol{\theta} \mid \boldsymbol{x}')$, and, again, a potentially different conclusion based on that modified knowledge[3]. This property seems desirable, as

[2] The influence of different prior specifications on model-based inference and model-derived decisions can be systematically investigated and quantified via *prior sensitivity analysis* [7]. More on this in Chapter 7.

[3] However, if the observations are equally informative or equally uninformative, the two posteriors may be the same. In general, the information gained by observing different states of the world will be different.

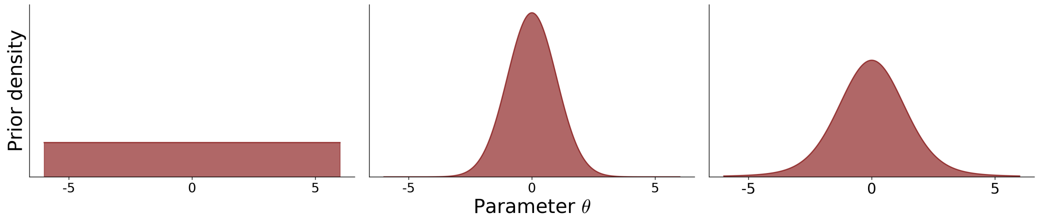

Figure 3.1: Three different ways to express prior knowledge and uncertainty about a single parameter θ in a probabilistic way. The left panel depicts a uniform prior around 0 which assigns constant density to values within the interval $[-6, 6]$ and treats all values outside this interval as literally impossible. The middle and right panel depict symmetric stable priors [179] around 0 which assign highest density to the central value of 0 and exponentially decaying density to values diverging from the center. Note, that the distribution in the right panel appears *shorter* than its *normal* counterpart in the middle panel, since it has thicker tails and treats values farther from 0 as less improbable.

it captures the (rather trivial) intuition that two learners presented with different facts will learn different things.

A further desirable property of Bayesian inference is that it allows the simultaneous or sequential integration of all available information (i.e., yesterday's posteriors become today's priors). Provided that information does not decay between sequential updates, both simultaneous and sequential updating should lead to the same endpoint posterior. The latter method is especially useful when data are collected at different points in time or arrive as a stream of observations. Curiously, even though often cited as a crucial asset of Bayesian inference, sequential updating is largely underutilized in behavioral research[4].

3.2 Uncertainty Reduction and Bayesian Surprise

The transition from prior to posterior essentially conveys a *reduction in uncertainty* brought about by observing reality. Equivalently, it can be seen as communicating the *gain in information* achieved by querying nature through an empirical endeavor. Accordingly, we expect the posterior to be *narrower* or *sharper* than the prior, as the opposite would imply a loss of information through observation - a scenario which appears rather paradoxical. However, it is reasonable to expect circumstances when the posterior will exactly equal the prior, namely, whenever the data carry no information about the parameters of interest.

The concept of *posterior contraction* formalizes the idea that the posterior should get sharper as the number N of available observations increases. In the simplest case, the posterior variance should decrease at rate $1/N$, but a more nuanced behavior can occur for more complex models inducing, for example, multi-modal or asymmetric posteriors. In general, the posterior contraction (PC) for multivariate posteriors can be formally expressed as a ratio between generalized variances:

[4] And for good reasons, since updating turns out to be technically hard if the posterior is represented only as random draws (e.g., as provided by MCMC algorithms).

$$\mathrm{PC}[\boldsymbol{\theta} \mid \boldsymbol{x}] = 1 - \frac{\det(\mathrm{Cov}[\boldsymbol{\theta} \mid \boldsymbol{x}])}{\det(\mathrm{Cov}[\boldsymbol{\theta}])} \tag{3.3}$$

where $\mathrm{Cov}[\boldsymbol{\theta} \mid \boldsymbol{x}]$ and $\mathrm{Cov}[\boldsymbol{\theta}]$ denote the posterior and prior covariance matrices, respectively. Posterior contraction near zero indicates that the data contribute little to nothing to reducing prior uncertainty about $\boldsymbol{\theta}$. Posterior contraction near one indicates a large reduction in prior uncertainty after accounting for the data [144]. Note, that Equation 3.3 provides *local* (i.e., per-observation) information about information gain. If one is interested about *global* (i.e., in expectation over all possible observations) information gain for a particular model, then the expected posterior contraction (EPC) should be considered:

$$\mathrm{EPC}[\boldsymbol{\theta} \mid \boldsymbol{x}] = 1 - \mathbb{E}_{p(\boldsymbol{x})}\left[\frac{\det(\mathrm{Cov}[\boldsymbol{\theta} \mid \boldsymbol{x}])}{\det(\mathrm{Cov}[\boldsymbol{\theta}])}\right] \tag{3.4}$$

Accordingly, different model parameterizations can be compared with respect to their expected information gain by computing the corresponding EPCs. However, the EPC will usually be intractable for complex models, so it needs to be approximated via its empirical mean over a sufficiently large number of different observations. For most non-trivial models, this will only be feasible in an *amortized inference* setting (to be discussed in the next chapter) which makes inference globally efficient.

Notably, posterior contraction only considers the variance, that is, the second moments of the prior and posterior distributions. However, other differences between prior and posterior may manifest themselves in differences between higher-moments of the distributions. A concept for quantifying arbitrary differences between prior and posterior is the *Bayesian surprise*, which can be defined as the Kullback-Leibler (KL) divergence between the two densities:

$$\mathcal{B} = \mathbb{KL}[p(\boldsymbol{\theta} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{\theta})] \tag{3.5}$$

$$= \int_{\Theta} p(\boldsymbol{\theta} \mid \boldsymbol{x}) \log\left(\frac{p(\boldsymbol{\theta} \mid \boldsymbol{x})}{p(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} \tag{3.6}$$

Importantly, the Bayesian surprise is non-negative and equals zero if and only if the two densities are equal. In information theory, this quantity is termed the *relative entropy*, which, in a Bayesian context, represents the information gained by replacing the prior with the posterior. The units of information are then determined by the base of the logarithm. The *expected Bayesian surprise* (EBS) then encodes the average information gained from applying a particular model to the entire data space. If evaluation of the prior and posterior densities is analytically tractable, the empirical approximation of the EBS can therefore be used to quantify the impact of the data on Bayesian updating[5].

Global information gain and uncertainty reduction are especially useful in the model development phase, since they can inform researchers about the general utility of a computational model. Accordingly, a researcher may experiment with different theoretically plausible parameterizations of a model and prune those which result in a poor information gain. Such an approach involves

[5] In cases where even approximating the (expected) Bayesian surprise is infeasible, an integral metric such as the maximum mean discrepancy (MMD, [61]) can be used to define Bayesian surprise as well.

performing repeated simulations from each model and then obtaining a posterior for each single simulation and model[6]. Notably, this approach can quickly become practically infeasible even for a few simulations if estimating the posterior is computationally demanding. Again, we will see in later sections why efficient amortized inference is particularly helpful for such undertakings.

3.3 Types of Uncertainty

The notion of uncertainty is central to scientific and daily life. However, in most research contexts dealing with probabilistic matters, it is often useful to introduce a *taxonomy of uncertainties*. A canonical approach in predictive modeling draws a distinction between *aleatoric* (sometimes called *ontic*) and *epistemic uncertainty* [74, 82].

Broadly speaking, aleatoric uncertainty is caused by intrinsic randomness, whereas epistemic uncertainty is brought about by ignorance. To illustrate this distinction with a (rather trivial) example [74], consider a coin-flipping game taking place in Bulgaria. A probabilistic model of the coin-flipping process could inform us about the likelihood of obtaining head or tail on any given toss. Further, assuming our probabilistic model is calibrated to reality, we can derive from it the expected value of the coin-flipping game and use decision theory to select certain actions (i.e., whether to bet my watch on the next toss). However, our model cannot foretell the *concrete* future outcome due to aleatoric reasons (matters of statistical physics aside), so there is uncertainty regarding the future possession of my watch. Differently, one might be equally uncertain about the meaning of the word "ezi" in Bulgarian. Yet the possible answers (and corresponding probabilities) are the same as before: head or tail. In this case, the ensuing uncertainty is caused purely by our lack of linguistic knowledge (i.e., epistemic uncertainty).

It has been argued that distinguishing between aleatoric and epistemic uncertainty constitutes a so-called *distinction without a difference* [154], that is, one without any practical consequences for decision makers. However, since ignorance (as opposed to intrinsic randomness) can in principle be reduced or even removed with additional information, it appears that pinpointing the source of uncertainty implies different handling of uncertainty. Thus, in our coin-flipping game, one can easily get rid of the epistemic uncertainty surrounding the word "ezi" by means of a dictionary. No need to set up a probabilistic model. Accordingly, epistemic uncertainty is generally treated as reducible; aleatoric uncertainty is generally treated as irreducible.

Note, however, that the irreducibility of aleatoric uncertainty is, in most cases, rather a practical decision than an ontological necessity. For instance, even in the coin-flipping game (the all-time favorite example of probability textbooks), knowledge of the initial conditions and of all forces acting on the coin at all times would, in principle, render the system deterministic. To further illustrate this point, consider the following (deterministic) linear model with two covariates, x_1, x_2, and one outcome y:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} \tag{3.7}$$

[6] Despite being self-consistent, such an approach does not guarantee the utility of a model when applied to real data. A potential *simulation-gap* between simulation and reality can be detected in later modeling stages with different tools.

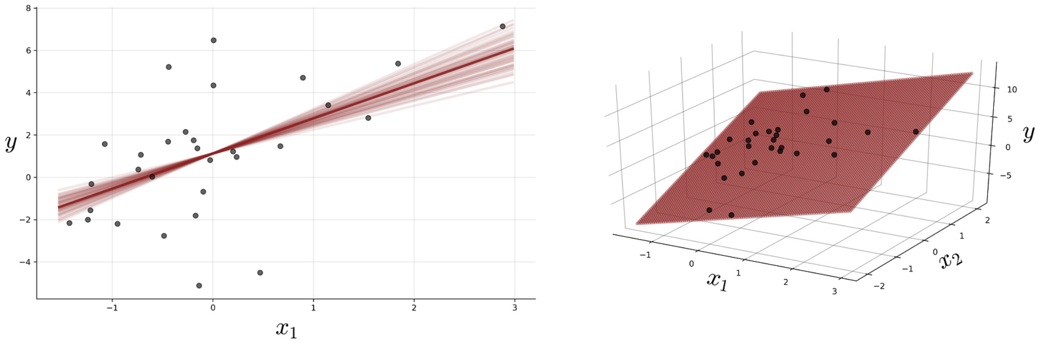

Figure 3.2: An illustrative example of the contextual irreducibility of aleatoric uncertainty. Left panel: having limited access to the relevant data (only x_1) induces aleatoric variability which cannot be reduced with further observations of the same kind. The deterministic model appears stochastic. Right panel: having access to the full data (both x_1 and x_2) makes the uncertainty disappear, since the true model is deterministic.

with all $x \sim \mathcal{N}(0,1)$. If one were to fit a proverbial simple linear regression on a reduced data set $\boldsymbol{D} = \{x_{1i}, y_i\}_{i=1}^{N=30}$ generated from the model, with x_2 treated as unknown to the modeler, the data would behave as if they had been generated via the well-known:

$$y_i = \beta_0 + \beta_1 x_{1i} + \xi_i \tag{3.8}$$

with $\xi \sim \mathcal{N}(0, \sigma)$, where σ is aleatoric Gaussian noise dependent on the unknown β_2 and x_2.

The data set and the resulting best-line fit are depicted in the left panel of Figure 3.2. In this simple regression scenario, epistemic and aleatoric uncertainty are seemingly separable. Due to the finite N, there is epistemic uncertainty about the precise values of β_0 and β_1, which might change if a different set of 30 observations was generated from the model. In a Bayesian setting this uncertainty would be captured via the posterior distribution $p(\beta_0, \beta_1 \mid \boldsymbol{D})$, which will get narrower as N increases. Note also, that the epistemic uncertainty in β_0 and β_1 results in uncertainty in the best-line fit, as depicted by the multiple shaded lines. However, regardless of how large N gets, the aleatoric factor ξ will cause the points to vary unsystematically around the line, and, correspondingly, there will be irreducible predictive uncertainty about the true outcome of an upcoming observation $x_{1i'}$. On the other hand, if we could augment our original data set with (the previously hidden) x_2, that is, if we use $\boldsymbol{D}' = \{x_{1i}, x_{2i}, y_i\}_{i=1}^{N=30}$ in a multiple regression model, all previous aleatoric uncertainty disappears, since the additional information of x_2 renders all points to perfectly lie on the best-fitting plane (cf. Figure 3.2, right panel). Thus, what we ultimately treat as irreducible uncertainty might depend on the particular modeling context instead of referring to an absolute notion.

When performing Bayesian inverse inference with Monte Carlo simulators, we typically want to mimic the real-world randomness, regardless of its source, via the stochastic component $p(\boldsymbol{\xi})$ involved in forward inference:

$$\boldsymbol{x}_n = g(\boldsymbol{\theta}, \boldsymbol{\xi}_n) \quad \text{with} \quad \boldsymbol{\xi}_n \sim p(\boldsymbol{\xi}) \tag{3.9}$$

Since we are dependent on computer-based random number generation, we typically need to assume a parametric form for $p(\boldsymbol{\xi})$ in order to make forward inference tractable in the first place. Even though in some applications (mostly experiments in physics), the precise source and form of the data-corruption process might be known in advance, most cognitive models are confined to a convenient form of uncertainty (i.e., one which we can easily simulate or write down mathematically). Thus, there is uncertainty about the chosen form of randomness, which, along with the uncertainty about the generative capabilities of the simulator, would fall under the rubric of *model uncertainty*, generally considered as another (sub-)type of epistemic uncertainty [74]. However, since the true data generator in areas reserved for the behavioral sciences is almost never transparent, model uncertainty is almost always practically irreducible. Accordingly, the reducible uncertainty encoded by the Bayesian posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N})$ is immersed into an ocean of potentially irreducible uncertainty and the sought-after Bayesian surprise is only guaranteed when the observed data are informative for the target parameters $\boldsymbol{\theta}$.

3.4 Exchangeable Observations

In the previous chapter, we briefly touched upon the idea of a memoryless probabilistic program. Running a memoryless simulation means that each run of the simulator is not affected by any of the previous runs and will, in turn, not affect any of the future runs. But on what grounds can we assume that such models can even remotely do justice to the noisy vicissitude of reality?

The answer is simplicity. Consider an observed sequence of N data points $\boldsymbol{x}_{1:N}$, that is, a data set obtained from an experiment or in an observational study, waiting to be analyzed by an eager scientist. Without a model in mind, the data set can be assumed to arise from an unknown random process $p^*(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$, also known to statisticians as the *data-generating* process. Generally, it will be beyond our intellectual reach to provide a complete description of how even a single observation in a given behavioral data set has come about[7]. Thus, we often need to leverage some probabilistic symmetry imposed on p^* which renders the data describable with compact and interpretable models. One such symmetry, tirelessly assumed in the Bayesian modeling literature, is *exchangeability*.

To illustrate exchangeability, imagine a scenario, in which researcher A, for whatever reasons, conspired to shuffle the observations in a data set collected by their colleague B. Under the assumption of exchangeability, the mischievous researcher A would be wasting their time - the ordering of the data points does not matter at all to researcher B. Formally speaking, researcher B views the sequence of data points as *invariant under all permutations of the data points* and plans to build a memoryless model of the data conforming to this view.

Exchangeable observations may come in various forms, for instance, patients entering a hospital, psychology students participating in a response time experiment, or raw response times recorded from a single participant. Exchangeable observations are not only conceptually simple to work with, but also admit a particularly useful probabilistic treatment. According to de Finetti's

[7] Just try to think of all possible factors affecting a single response time of a drunk participant in a reaction time study.

representation theorem [33, 116], an exchangeable data-generating distribution p^* has the following integral decomposition:

$$p^*(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) = \int_\Upsilon p^*(\boldsymbol{\upsilon}) \prod_{n=1}^{N} p^*(\boldsymbol{x}_n \mid \boldsymbol{\upsilon})\, d\boldsymbol{\upsilon} \tag{3.10}$$

with $\boldsymbol{\upsilon} \in \Upsilon$ being an (potentially infinitely dimensional) absolutely continuous random vector. Even though the representation theorem does not provide any clues on how to actually find the correct integral decomposition, it motivates the formulation of Bayesian models of the form

$$p(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n \mid \boldsymbol{\theta}) \tag{3.11}$$

as useful approximations of the unknown "parameter vector" $\boldsymbol{\upsilon}$ and densities $p^*(\boldsymbol{\upsilon})$ and $p^*(\boldsymbol{x} \mid \boldsymbol{\upsilon})$. Note also, that Equation 3.11 represents the numerator of Bayes' rule for multiple *i.i.d.* observations. Essentially, this equation represents a statement of *conditional independence* and specifies a generative recipe for simulating synthetic observations: first, obtain a random sample from the prior $p(\boldsymbol{\theta})$ and run the simulator N times with the corresponding sample to obtain a simulated tuple $(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$.

Exchangeable observations impose a symmetry on the posterior as well. As a consequence of the assumption, the resulting posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N})$ should also be invariant with respect to the ordering of the individual observations $\boldsymbol{x}_n$. In other words, if the function $\mathbb{S}_N(\boldsymbol{x}_{1:N}) = (\boldsymbol{x}_{\pi(1)}, \ldots, \boldsymbol{x}_{\pi(N)})$ represents an arbitrary permutation of N elements, the following should hold for the posterior:

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N}) = p(\boldsymbol{\theta} \mid \mathbb{S}_N(\boldsymbol{x}_{1:N})) \tag{3.12}$$

The same applies to any (sufficient) summary statistic of the data $h(\boldsymbol{x}_{1:N})$:

$$p(\boldsymbol{\theta} \mid h(\boldsymbol{x}_{1:N})) = p(\boldsymbol{\theta} \mid h(\mathbb{S}_N(\boldsymbol{x}_{1:N}))) \tag{3.13}$$

and, by extension, to any estimator of the posterior which is a function of the raw data. In later chapters, we will elaborate on why the preservation of posterior symmetry poses a challenge to standard neural network estimators. We will further discuss and describe how to employ specialized symmetry-preserving networks for addressing this challenge [13]. Albeit common, memoryless models are not ubiquitous throughout the behavioral sciences. Thus, our inference frameworks will incorporate different choices of neural architectures for stateful models and non-exchangeable distributions.

3.5 Bayesian Model Comparison

Our discussion on Bayesian inference so far has concentrated on a single abstract model. In fact, we have even treated the model as an invisible part of the unspecified context C implicit to all distributions involved in the computation of Bayes' rule (Equation 3.1). However, this simplified setting is rather rare in the behavioral sciences. In most model-rich research areas, such as decision making, attention, judgement formation, risk-taking, and memory, there is a multitude of models and model parameterizations competing to account for a given behavior. Moreover, in a Bayesian context, different models can be defined not only via different generative mechanisms, but also by different priors (e.g., as in prior sensitivity analysis, [163]), or by different data processing pipelines (e.g., as in multiverse analysis, [149]). Thus, researchers often find themselves in a scenario calling for formal model comparison, model selection, or model averaging.

In order to extend our discussion and formal notation to multiple models, consider a collection of J candidate models $\mathcal{M} = \{\mathcal{M}_1, ..., \mathcal{M}_J\}$ and corresponding parameter spaces $\mathcal{H} = \{\Theta_1, \ldots, \Theta_J\}$. We assume that each model is realizable via a generative algorithm and a simulation program, such that $g_j(\boldsymbol{\theta}_j, \boldsymbol{\xi}_j)$ realizes $\mathcal{M}_j$ for each j. Thus, in order to perform Bayesian parameter estimation for each model given an observation $\boldsymbol{x}$, we need to compute J posteriors of the form

$$p(\boldsymbol{\theta}_j \mid \boldsymbol{x}, \mathcal{M}_j) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta}_j, \mathcal{M}_j)\, p(\boldsymbol{\theta}_j \mid \mathcal{M}_j)}{p(\boldsymbol{x} \mid \mathcal{M}_j)} \tag{3.14}$$

with the prior, posterior, likelihood, and marginal likelihood explicitly written as dependent on the particular model $\mathcal{M}_j$.

How does one assign preferences to competing models using a probabilistic toolkit? Assuming that all quantities in Equation 3.14 are tractably computable, Bayesian methods for model selection revolve around two key concepts: *prior predictive measures* and *posterior predictive measures* [52]. Prior predictive and posterior predictive approaches to model comparison answer somewhat different questions, so asking "which one is better" for a specific modeling problem is rarely expedient. Moreover, their answers can occasionally diverge, so oftentimes, it is informative to explore *both* approaches in order to obtain a more nuanced picture of the candidate models' performance. The most commonly faced obstacle in practice is feasibility: both approaches are computationally expensive (sometimes intractable) for complex models and can thus benefit from our proposed frameworks. We now briefly discuss each approach.

3.5.1 The Prior Predictive and Occam's Razor

The canonical measure of prior predictive performance is the already encountered *evidence*, which is the denominator in Bayes' rule:

$$p(\boldsymbol{x} \mid \mathcal{M}_j) = \int_{\Theta_j} p(\boldsymbol{x} \mid \boldsymbol{\theta}_j, \mathcal{M}_j)\, p(\boldsymbol{\theta}_j \mid \mathcal{M}_j)\, d\boldsymbol{\theta}_j \tag{3.15}$$

Note, that the integral runs over the prior space of each model $\mathcal{M}_j$ and thus represents the expected likelihood with respect to each model's prior. The evidence thus penalizes prior complexity, that is, the inelegance, of a model, since the prior acts as a weight on the likelihood. It also induces a well-known and widely appreciated source of intractability in Bayesian inference, since it typically involves a multi-dimensional integral over potentially unbounded parameter spaces. For most non-trivial models, this integral cannot be computed in closed-form or approximated numerically. Sophisticated algorithms for efficiently approximating the evidence have been proposed in the Bayesian universe, such as *bridge sampling* and *path sampling* [54, 62]. However, these methods still depend on the ability to evaluate the likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta}_j, \mathcal{M}_j)$ for each candi-

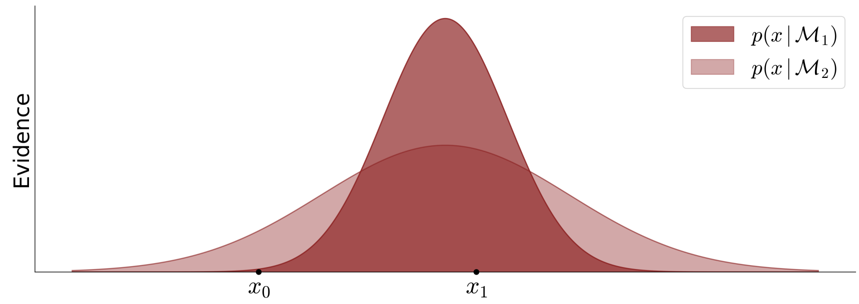

Figure 3.3: An illustrative example of Bayesian Occam's razor. The figure depicts the hypothetical evidence (marginal likelihood) of a simple model $\mathcal{M}_1$ vs. that of a more complex model $\mathcal{M}_2$. The complex model has a larger generative scope and thus accounts for a broader range of observations by spreading its marginal likelihood to cover the whole range. It does so at the cost of diminished sharpness. Even though observation x_1 is well within its generative scope, the simpler model $\mathcal{M}_1$ yields a higher evidence and is therefore favored. In contrast, observation x_0 has a higher evidence under model $\mathcal{M}_2$, as it is very unlikely to be generated by the simpler model.

date model. If the likelihood itself is intractable, as is the case with complex simulators, the task becomes increasingly hopeless even with the most efficient current methods.

Provided that the marginal likelihood can be reliably approximated, one can compute the ratio of marginal likelihoods for two models $\mathcal{M}_j$ and $\mathcal{M}_k$

$$\mathrm{BF}_{jk} = \frac{p(\boldsymbol{x} \mid \mathcal{M}_j)}{p(\boldsymbol{x} \mid \mathcal{M}_k)} \tag{3.16}$$

This famous ratio is called a Bayes factor (BF) and is used in Bayesian settings for quantifying relative model preference. Thus, a BF > 1 indicates preference for model j over model k, taking observation $\boldsymbol{x}$ into account. Alternatively, one can directly focus on the (marginal) posterior probability of a model $\mathcal{M}_j$

$$p(\mathcal{M}_j \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \mathcal{M}_j)\, p(\mathcal{M}_j) \tag{3.17}$$

which equips the model space itself with a multinomial prior distribution $p(\mathcal{M})$ encoding potential prior beliefs on the plausibility of each model before collecting any data. Such a prior might be useful if a model embodies extraordinary claims (e.g., psychokinesis) and thus requires extraordinary evidence supporting it. However, if no prior reasons can be given for favoring some models over others (i.e., one prefers not to prefer), a uniform model prior $p(\mathcal{M}_j) = 1/J$ can be assigned.

The ratio of posterior model probabilities is called the *posterior odds* and is connected to the Bayes factor via the corresponding model priors:

$$\frac{p(\mathcal{M}_j \mid \boldsymbol{x})}{p(\mathcal{M}_k \mid \boldsymbol{x})} = \frac{p(\boldsymbol{x} \mid \mathcal{M}_j)}{p(\boldsymbol{x} \mid \mathcal{M}_k)} \times \frac{p(\mathcal{M}_j)}{p(\mathcal{M}_k)} \tag{3.18}$$

If both models are deemed equally likely *a priori*, that is, $p(\mathcal{M}_j) = p(\mathcal{M}_k)$, the posterior odds are identical to the Bayes factor. In this case, if the Bayes factor, or, equivalently, the posterior odds equal one, the observed data provide no evidence for favoring one of the models over the other. Importantly, a relative evidence of one does not distinguish whether the data are equally likely or equally unlikely under both models, as this is a question of absolute evidence. It simply means that the observations are not informative to the question of model comparison[8].

It follows from our discussion so far, that, from a prior predictive perspective, model complexity is determined by (i) the prior of a model and (ii) the likelihood function of a model. Together, these two quantities establish the *generative scope* of a model and dictate how to select between two hypothetical cognitive models which both provide a reasonable account of some behavioral manifestations.

Thus, a complex model g_j with a large generative scope can generate a larger variety of behaviors, and so the density of $p(\boldsymbol{x} \mid \mathcal{M}_j)$ must spread over a larger portion of the observation space $\mathcal{X}$. On the other hand, a simpler model g_k with a smaller generative score can generate a limited range of behaviors, so its density will be restricted to a smaller portion of $\mathcal{X}$, and thus more peaked (cf. Figure 3.3). This is the reason why a marginal likelihood is said to automatically embody a fundamental trade-off between a model's complexity (an antonym of scientific elegance) and its ability to convincingly account for a wide variety of empirical phenomena. Indeed, this trade-off is sometimes embraced as a probabilistic version of the famous Occam's razor.

From a generative perspective, simpler simulators will tend to produce synthetic observations which are more similar to each other compared to those generated via more complex simulators. Indeed, this is the most important property of Bayesian models that we will later leverage in order to perform efficient model comparison between complex cognitive simulators. Essentially, by approximating Equation 3.17 directly, we will be circumventing two common sources of intractability: the marginal likelihood and the likelihood function itself.

3.5.2 The Posterior Predictive and Fortuna's Knife

Prior predictive measures such as the Bayes factor do not utilize the posterior when comparing models (cf. Equation 3.15). In contrast, posterior predictive methods quantify the ability of models to forecast *new observations* which have not been used for Bayesian updating. In other words, if the posterior of each model represents a modified state of knowledge upon integrating observation $\boldsymbol{x}$, one tests which form of modified knowledge can "best" predict an unseen observation $\boldsymbol{x}'$. The most straightforward way to perform such an operation consists in computing the *posterior predictive* distribution:

$$p(\boldsymbol{x}' \mid \boldsymbol{x}, \mathcal{M}_j) = \int_{\Theta_j} p(\boldsymbol{x}' \mid \boldsymbol{\theta}_j, \boldsymbol{x}, \mathcal{M}_j)\, p(\boldsymbol{\theta}_j \mid \boldsymbol{x}, \mathcal{M}_j)\, d\boldsymbol{\theta}_j \tag{3.19}$$

where the likelihood $p(\boldsymbol{x}' \mid \boldsymbol{\theta}_j, \boldsymbol{x}, \mathcal{M}_j)$ simplifies to $p(\boldsymbol{x}' \mid \boldsymbol{\theta}_j, \mathcal{M}_j)$ when working with memoryless models. The main difference now is that integration is performed with respect to the posterior distribution of each model. Intuitively, the posterior predictive uses all information gained

[8] Unless one assumes that the model set $\mathcal{M}$ provides an exhaustive description of reality, in which case relative evidence *is* absolute evidence.

through previous observation(s) to provide an uncertainty-aware forecast for the new query $\boldsymbol{x}'$, where the parameter posterior $p(\boldsymbol{\theta}_j \mid \boldsymbol{x}, \mathcal{M}_j)$ acts as a weight on the likelihood. Thus, the probability of an upcoming observation under a model $\mathcal{M}_j$ is a weighted average over its probabilities under all plausible parameter configurations $\boldsymbol{\theta}_j$. Thereby, epistemic uncertainty, as captured by the parameter posterior, is essentially "averaged out" in the posterior predictive.

The canonical approach for Bayesian posterior predictive comparisons are cross-validation (CV) methods [166]. Examples for widely applied methods that fall into this category are approximate cross-validation methods using Pareto-smoothed importance sampling (PSIS-CV) [17, 165], information criterion measures, such as the widely applicable information criterion (WAIC; [171]), or stacking of posterior predictive distributions [176]. All of these methods require not only the ability to evaluate the likelihood of each model for each observation during parameter estimation, but also for new observations during prediction.

What is more, if application of exact CV methods is required because approximations are insufficient or unavailable, models need to be estimated several times based on different data sets or subsets of the original data set. This renders such methods practically infeasible when working with complex simulators for which posterior inference is already computationally demanding. Thus, even a single intractable model in the candidate model set suffices to disproportionately increase the difficultly of performing posterior predictive model comparison.

Posterior predictive measures based on evaluating the likelihood of new data points provide information about the *relative* performance of models. For certain applications, one can replace the likelihood in Equation 3.19 with a scoring function $S : \mathcal{X} \to \mathbb{R}$ and arrive at a measure of model predictive performance with respect to the scoring function:

$$s_j = \int_{\Theta_j} \int_{\Xi_j} S(\boldsymbol{x}', g_j(\boldsymbol{\theta}_j, \boldsymbol{\xi}_j))\, p(\boldsymbol{\xi}_j \mid \boldsymbol{\theta}_j)\, p(\boldsymbol{\theta}_j \mid \boldsymbol{x}, \mathcal{M}_j)\, d\boldsymbol{\xi}_j\, d\boldsymbol{\theta}_j \tag{3.20}$$

where the integrals are typically approximated via Monte Carlo samples from the corresponding posterior and noise distributions. If the scoring function is well aligned with the particular goals of an inference task, it can serve as a useful proxy for quantifying both absolute and relative performance of the models under scrutiny. Moreover, it replaces the dependence on the likelihood, at least during prediction. Still, the estimation of each posterior and its repeated re-estimation for different data sets when using variants of exact LOO-CV remain as potential bottlenecks. These bottlenecks become even more challenging to overcome when doing simulation-based inference even with a single model of interest.

Note, that oftentimes researchers use the term *predict* when they are actually referring to a model's ability to *reproduce* the data used to inform the estimation of model parameters [177]. Formally, this amounts to replacing the new observation $\boldsymbol{x}'$ in the previous two equations with the observation $\boldsymbol{x}$ (or set of observations $\boldsymbol{x}_{1:N}$) used to condition the posterior. Such a procedure is indeed useful, since it can be indicative of a model's *generative performance* [120] and can thus help in diagnosing model misspecifaction or a simulation gap. However, generative performance is generally *not indicative* of a model's predictive performance [53, 177], so we find it important to highlight and keep the distinction as clear and explicit as possible.

At this point, there is an important distinction to be made when it comes to predicting *new* observations. Suppose that each observation in a data set was generated by the unknown process

p^* of which we formulate a parametric model $p_{\boldsymbol{\theta}}$. If we now make the model "blind" to certain observations in the original data set (as in CV) and use these observations to assess predictive performance, we are essentially testing the model's ability to perform *induction* about the statistical regularities of the process. In such a scenario, however, we are not assessing the model's ability to faithfully forecast the future, since the observations are new only from the relative perspective of model fitting. An attempt to forecast future behavior with a static cognitive model will only be meaningful, if the process p^* is stationary (i.e., its regularities are invariant with respect to time) or if the model somehow explicitly incorporates a potential time-dependent change inherent to the generator. Since such changes are extremely hard to know in advance (otherwise they would have been predicted and incorporated into the model's equations), a model which claims time-invariant performance should regularly be subjected to the falsification of time.

Finally, note that the probabilistic Occam's razor from prior predictive approaches does not automatically show up in posterior predictive approaches. Differently, from a posterior predictive perspective, a model's quality should be judged based on how well it *generalizes* to unseen instances it is supposed to accurately predict. In other words, a model has to withstand the sharp challenges of its future, and models which fail to do so, are discarded. However, one might still anticipate that overly complex models would *overfit* the data at hand and fail dramatically at predicting new data, again being implicitly subjected to some form of Occam's razor. Indeed, such an anticipation has been formally framed under the so called *bias-variance dilemma* [47] which bounds the generalization error of supervised learning algorithms. However, the practical consequences of such a dilemma have recently been called into question by the achievements of "black-box" neural networks having billions of parameters and still being able to generalize beyond their training data [111]. It remains therefore unclear to what extent Occam's razor is implicitly encoded in posterior predictive measures when applied to "white-box" models of cognition.

3.5.3 Bayesian Model Averaging and Wisdom of the Crowd

A rather disparate approach to model selection is that of *not selecting* a single model but instead *averaging* across the predictions of all candidate models. However, in a Bayesian model averaging (BMA) setting, models are not created equal, and thus not weighted with indifference to their performance or elegance. Instead, posterior model probabilities are used as weights in the aggregate prediction:

$$p(\boldsymbol{x}' \mid \boldsymbol{x}) = \sum_{j=1}^{J} p(\boldsymbol{x}' \mid \boldsymbol{x}, \mathcal{M}_j)\, p(\mathcal{M}_j \mid \boldsymbol{x}) \tag{3.21}$$

Crucially, BMA depends on the marginal likelihoods of the different models, as well as on their likelihood functions. Naturally, BMA can also be used for averaging over scoring functions instead of posterior predictive distributions, when a proxy of absolute performance is needed. BMA is particularly useful when predictive performance matters and predictions of a single model are expected to be unstable. In fact, model averaging generally yields superior predictive results in expectation compared to those obtained by model selection [126]. Broadly speaking, it can be regarded as the Bayesian embodiment of the well-known *wisdom of the crowd*, or *vox populi*, statistical phenomenon [50].

Clearly, BMA is of limited use when the goal of inference is to compare competing theories instantiated by formal models and subsequently choose the most plausible among all (in a probabilistic sense). However, BMA might still come in handy for selecting between *model classes*, where the performance of single model instances is not crucial or when competing models might admit different plausible parameterizations.

Consider, for example, a class of mathematical models $\mathcal{A} = \{\mathcal{M}_j^{\mathcal{A}}\}_{j=1}^{J}$, having property A in common, and another class of models $\mathcal{B} = \{\mathcal{M}_j^{\mathcal{B}}\}_{j=1}^{J}$, having property B in common. Applying Equation 3.21 to the models in each class, we can obtain two model-averaged posterior predictive distributions $p(\boldsymbol{x}' \mid \boldsymbol{x}, \mathcal{A})$ and $p(\boldsymbol{x}' \mid \boldsymbol{x}, \mathcal{B})$. Subsequently, we can use these to quantify and compare the bulk predictive performance of the two model classes.

Performing BMA for selecting between model classes featuring complex models (e.g., memoryless vs. stateful models) is, however, even more computationally demanding, which makes it a highly underutilized approach in the behavioral sciences. It also presents a challenge we consider worthy of future investigation in the context of frameworks for simulation-based Bayesian inference.

3.6 Bayesian Simulation-Based Inference

We have seen that all central objects in Bayesian inference, from the Bayesian parameter posterior to the Bayesian model-averaged predictive distribution, depend on the likelihood function. Thus, we reiterate, when the likelihood cannot be efficiently evaluated or is not available in closed-form, standard Bayesian methods relying on the central proportionality $p(\boldsymbol{\theta} \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ do not apply. This includes both the less efficient but asymptotically appealing MCMC and the more efficient but asymptotically less winsome variational methods. Moreover, a potential intractable likelihood can also appear on top of the well-known intractability of the *marginal likelihood*, which is necessary for model comparison in a prior predictive context. In fact, this is not just a theoretical hardship, but a real practical predicament faced by researchers working with various complex models [27]. Complexity is usually a direct consequence of the desire to build high-fidelity models of cognitive processes, sometimes coupled with a modicum of neurophysiological realism.

In such cases, not all is lost. Instead of simplifying the model *ad hoc*, one can still retain the advantages (and disadvantages) of Bayesian inference by “simply” performing repeated simulations and using them to guide the process of inference. Such an approach is known as *simulation-based inference*. When the resulting quantities still represent the reduction of prior uncertainty by conditioning on data, we are essentially doing the same Bayesian inference as before, only with the likelihood implicitly involved in the process. All Bayesian simulation-based methods repeatedly go through the following three steps, given informally by:

1. Obtain a random sample from the prior.

2. Simulate an artificial data set with the sampled parameters.

3. Do something with the simulated pair of parameters and data.

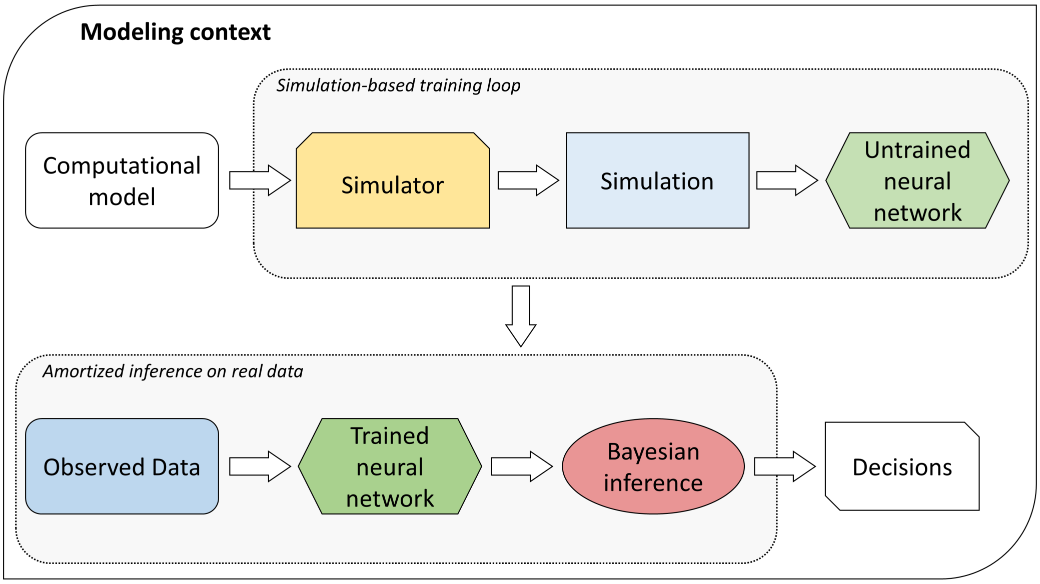

Figure 3.4: An abstract overview of the central idea underlying our proposed solutions to Bayesian inverse inference.

The essential difference between most Bayesian simulation-based methods lies in the particular implementation of step number three. Our frameworks are no exception to this pattern. A common theme will be to use the cognitive model as a trainer for a specialized neural network, which is driven through a number of simulations towards the Bayesian answer to an estimation or a model comparison problem (cf. Figure 3.4. The next chapter will provide a brief and very broad survey of the related work on simulation-based inference. Before we move on, however, we shortly discuss the idea of *samplers*, as it is central to our developed methods and probabilistic modeling as a whole.

3.7 Samplers and Neural Samplers

It is important to note, that the posterior distribution itself is hardly ever available as a known density function which can be analytically calculated. In the typical textbook cases where the likelihood belongs to a known family of probability distributions and the prior is chosen to be *conjugate* to the likelihood, then the posterior must belong to the same distribution family as the prior. However, such mathematical convenience has proven insufficient for addressing most unidealized real-world problems, as it leaves little room for flexible or high-fidelity modeling. Thus, researchers and statistical software developers have resorted to MCMC methods.

MCMC methods, such as the Metropolis-Hastings algorithm [66], Gibbs sampling [51], Hamiltonian Monte Carlo [112] or its extension to No-U-Turn sampling [71], approximate the posterior in the form of *random samples from the target posterior*. MCMC methods belong to a family of stateful algorithms which generate a sequence of correlated samples that converge in distribution to a target equilibrium distribution (i.e., the posterior, in a Bayesian setting).

The idea of approximating a complicated distribution via dependent random samples, albeit rather straightforward in hindsight, has gradually transformed and shaped the field of Bayesian inference. Moreover, it forms the main logic behind major probabilistic programming languages such as JAGS [127] or Stan [19], which return posterior estimates in the form of random samples. A *sampler* is thus a program which uses computer-generated randomness to "draw" samples from a distribution instead of deriving or estimating its algebraic form.

Recently, the idea of random sampling has entered the rapidly expanding field of deep learning under the umbrella term *deep generative modeling*. As a consequence, the concept of *neural samplers* has emerged [72, 87, 113, 114]. Neural samplers emulate sampling from a distribution via neural networks that transform a random input vector into a sample from a target probability distribution defined by the network weights. The random input vector is typically drawn from a simple distribution (e.g., uniform or Gaussian) which is computationally cheap and easy to sample from. The expressive transformation of simple inputs creates diversity and flexibility. Accordingly, the network weights defining the transformation are optimized in a way to ensure that the subsequently generated samples are actually representative of the target distribution.

Neural samplers have so far shown tremendous success in computer vision and natural language processing. Our inference frameworks also make extensive use of neural samplers. However, our neural networks are trained to approximate the Bayesian posterior induced by a particular model given a set of observations and thus generate posterior samples in a way similar to MCMC. Moreover, we will utilize the important fact that neural samplers distil global information about a particular model family into their trainable parameters (e.g., network weights). Thus, once trained, neural samplers are easy to store on a computer or a server (a couple of megabytes memory demand) and re-use across multiple applications of the same model family to many data sets of potentially variable size. This property gives rise to *amortized inference*, a concept we will repeatedly encounter in later chapters.

4 Related Work

As with any new and rapidly expanding field, it is nearly impossible to manage a comprehensive review of the existing literature, since new methods would have emerged upon the review's completion. Indeed, this is exactly what happened during the (rather expeditious) completion of this thesis. Thus, this chapter attempts to provide a cursory glance upon the landscape of simulation-based inference with a special focus on deep learning methods. For a more detailed review of recent developments, see, for example, [27]. For a collection of classical methods with a focus on cognitive science, see, for example, [119].

There are multiple ways to devise a taxonomy for members of the zoo of simulation-based inference. For the purpose of this thesis, we can classify methods as *amortized* vs. *non-amortized*, with different degrees of amortization possible (see Chapters 5 and 7). The main difference between the two methodological endpoints is this. Non-amortized methods require a repetition of the same computations for each model application, that is, estimation starts from scratch for each observed data set $\boldsymbol{x}_{1:N}^{(obs)}$. In contrast, amortized methods involve an upfront optimization/training phase which ensures that subsequent applications of the model to any observed data set are very cheap (i.e., the cost of the optimization phase *amortizes* over multiple inferences).

The standard non-amortized solution to intractable modeling problems is offered by *approximate Bayesian computation* (ABC) methods [29, 150]. ABC methods approximate the posterior by repeatedly sampling parameters from a proposal (prior) distribution $\boldsymbol{\theta}^{(l)} \sim p(\boldsymbol{\theta})$ and then simulating a synthetic data set by running forward inference, $\boldsymbol{x}_n \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}^{(l)})$ for $n = 1, \ldots, N$, with the sampled parameters. If the simulated data set is sufficiently similar to an actually observed data set (as measured by a user-defined distance function), the corresponding parameter configuration $\boldsymbol{\theta}^{(l)}$ is retained as a random draw from the desired posterior, otherwise rejected. However, in practice, ABC methods are notoriously inefficient and suffer from various problems, such as the *curse of dimensionality* or *curse of inefficiency* [104], to name the most severe.

More efficient and sophisticated methods, such as sequential Monte Carlo algorithms (ABC-SMC) or Markov chain Monte Carlo with implicit likelihoods (ABC-MCMC), employ different creative techniques to optimize sampling or correct potential biases [65, 90, 107, 119, 123, 157, 158, 160]. Currently, the gold-standard in cognitive science and mathematical psychology appears to be non-amortized ABC-MCMC with kernel density estimation (KDE) [44, 160, 161]. This method has the advantage of doing away with hand-crafted summary statistics and distance functions, since each (simulated or actual) observation $\boldsymbol{x}_n$ enters the KDE computation at each MCMC step. However, the application of KDE-based MCMC to stateful models yielding non-*i.i.d.* observations is not at all straightforward, since KDE methods typically assume *i.i.d.* observations. Furthermore, every non-amortized method becomes infeasible in data-rich settings which require the estimation of the same model hundreds or even thousands of times (or even more than one million times, as was the case in [94]), unless a researcher has access to a high-end computing cluster with countless cores.

Recently, machine learning and deep learning innovations have permeated the field of simulation-based inference with the goal of scaling up standard ABC methods. Most of these innovations yield *amortized methods*, since they involve an expressive machine learning approximator (e.g., random forests or neural networks) trained on simulations from the Monte Carlo engine which emulates the behavior of the analytically intractable model. These approximators are either only able to return summaries of the full posterior (e.g., posterior means, variances, or quantiles) or capable of performing fully Bayesian inference. We now discuss each of the two approaches in turn.

Perhaps the most straightforward inference method has been to cast the problem of parameter estimation as a supervised regression task [14, 79, 136, 140]. In this setting, the simulator is run repeatedly to create a large dataset of the form $\mathcal{D} = \{(h(\boldsymbol{x}_{1:N}^{(m)}), \boldsymbol{\theta}^{(m)})\}_{m=1}^{M}$, also referred to as a *reference table* in the ABC literature [150]. Typically, the dimensionality of the simulated data is reduced by computing summary statistics with a fixed summary function $h(\boldsymbol{x}_{1:N})$ (but see [136]). Then, the reference table $\mathcal{D}$ is used as training data for a supervised learning algorithm (e.g., random forest [140], or a convolutional neural network [136]). The learning algorithm is trained to output an estimate of the true data-generating parameters and an optional uncertainty estimate (e.g., the posterior variance [14, 136] or quantiles [140]). Thus, supervised methods attempt to approximate the intractable inverse model directly and globally via non-linear regression $\widehat{\boldsymbol{\theta}} = f(h(\boldsymbol{x}_{1:N}))$. Importantly, the trained algorithms can be cheaply stored and re-used for estimation on an arbitrary number of observed data sets or integrated into an ABC routine [79].

A severe shortcoming of supervised approaches is that they provide only limited information about the full posterior or impose overly restrictive assumptions on its distributional form (e.g., Gaussian). This is especially problematic when the true posterior is acutely skewed or multi-modal, in which case the mean or the variance are not particularly representative of its relevant characteristics.

To address this shortcoming, neural density estimation (NDE) methods employ specialized neural networks capable of performing fully Bayesian inference (i.e., returning full posteriors). NDE methods approximate different components of the intractable joint Bayesian model, that is, $p(\boldsymbol{\theta}, \boldsymbol{x}_{1:N}) = p(\boldsymbol{\theta})p(\boldsymbol{x}_{1:N} \,|\, \boldsymbol{\theta})$.

Sequential neural posterior estimation (SNPE) methods iteratively refine a proposal distribution via specialized neural networks (e.g., mixture density networks, autoregressive or normalizing flows) to generate parameter samples which closely match a particular observed data set [60, 101, 121]. Even though these methods also entail a relatively expensive learning phase and a cheap inference phase, they are capable of amortized inference only when operating in a non-sequential manner (i.e., the prior is used as a proposal throughout every optimization step). Otherwise, a separate neural density estimator has to be trained for each observed data set, which quickly becomes infeasible when working with many data sets. The main feature of SNPE methods is that they avoid MCMC sampling altogether and are able to sample from the *true posterior* given perfect convergence. This is in contrast to variational methods which optimize a lower-bound on the posterior [87, 89], and oftentimes need to assume Gaussian approximate posteriors through the reconstruction error.

The sequential neural likelihood (SNL, [122]) method and the method of emulated likelihoods [100] propose to learn the likelihood instead of the posterior. In this way, the trained neural approximators can be integrated into standard Bayesian pipelines and can be re-used across changes in the prior model $p(\boldsymbol{\theta})$, as long as the likelihood remains invariant. Sequential neural ratio (SNR) methods [40, 68], on the other hand, propose to train a classifier to approximate density ratios. These density ratios can then be used to sample from the posterior via standard MCMC methods. Of course, the computational time of SNL and SNR methods will still be dominated by the non-amortized components (e.g., MCMC or alternative sampling schemes) when faced with more than a few data sets. The same will be true for *inference compilation* approaches, which calibrate specialized neural networks through simulations in order to improve proposal distributions within (non-amortized) sequential Monte Carlo [117].

An interesting approach for amortized inference which does not rely on neural networks is the pre-paid estimation method without likelihoods [108]. This method memorizes a large database of pre-computed summary statistics for fast nearest-neighbor inference, aided by advanced interpolation methods. Even though the pre-paid method is very powerful and applicable to all kinds of models, it still crucially depends on the ability to (heuristically) select good summary statistics. Thus, in a future work, it seems worthwhile to explore the possibility of combining the pre-paid method with a neural network capable of learning maximally informative summary vectors (as proposed in our frameworks).

Ideas for direct posterior estimation via NDE are closely related to the concept of *optimal transport maps* and its application in Bayesian inference [11, 23, 36, 125]. A transport map defines a transformation between (probability) measures which can be constructed in a way to *warp* a simple probability distribution into a more complex one. In the context of Bayesian inference, transport maps have been applied to accelerate MCMC sampling [125], to perform sequential inference [36], and to solve inference problems via direct optimization [11]. In fact, our BayesFlow framework can be loosely viewed as a parameterization of invertible transport maps via invertible neural networks. An important distinction from this line of research is that NDE methods do not require an explicit likelihood function for approximating the target posteriors and are capable of amortized inference.

Curiously, throughout the advancement of amortized NDE methods, the task of simulation-based Bayesian model comparison appears to have taken a backseat. With certain caveats, neural density estimators can be adapted for posterior/prior predictive Bayesian model comparison by post-processing the samples from an approximate posterior/likelihood over each model's parameters. However, such an approach will involve training a separate neural estimator for each model in the candidate set $\mathcal{M}$ and has not yet been systematically investigated. In addition, most NDE methods also rely on fixed summary statistics [121, 122]) as inputs to the networks and few applications using raw data directly exist [60].

Alongside advancements in simulation-based inference, there has been an upsurge in the development of methods for uncertainty quantification in deep learning applications [74]. For instance, much work has been done on the efficient estimation of Bayesian neural networks [69, 96, 99] since the pioneering work of [102]. Parallel to the establishment of novel variational methods [85, 86], the drive for representing uncertainty has paved the way towards more interpretable and trustworthy neural network applications. Moreover, the need for distinguishing between different sources of uncertainty and the overconfidence of deep neural networks in both classification and regression tasks has been demonstrated quite effectively in recent works [82, 145].

Our frameworks are inspired by recent methods for deep probabilistic modeling [38, 88] and uncertainty representation in classification tasks [145, 156]. However, our goal is to efficiently approximate Bayesian posteriors and posterior odds between competing mechanistic models using *non-Bayesian neural networks*, not to estimate neural network parameters (e.g., weights) via Bayesian methods. Indeed, the incorporation of Bayesian neural networks into our frameworks appears to be an interesting avenue for future research. Moreover, our frameworks combine some of the latest ideas from simulation-based inference and uncertainty quantification for training efficient and uncertainty-aware estimators capable of amortized Bayesian parameter estimation and model comparison.

Accordingly, we propose to solve each task *globally*, that is, over the entire range of plausible parameters, data sets, and models. For parameter estimation, we will employ invertible neural networks (INN, [3, 4, 37, 38]). Previously, INNs have been successfully employed to tackle inverse problems in astrophysics and medicine [3]. We will adapt these flow-based INN architectures to suit the task of Bayesian parameter estimation in the context of various intractable model and data types. As for model comparison, we will employ *evidential neural networks*, which have previously been convincingly employed for uncertainty-aware classification [145].

Finally, by introducing our frameworks, we will further discuss many open questions, such as end-to-end estimation of various model classes (e.g., memoryless models, stateful models, joint models) via algorithmic alignment, model validation, Bayes factor approximation, and ideas for meta-amortized inference.

5 Amortized Parameter Estimation with BayesFlow

Let reality be reality. Let things flow naturally forward in whatever way they like.
— Laozi

Estimating the parameters of cognitive models is a crucial task in behavioral and cognitive modeling. However, the task can prove notably difficult or even impossible when a model can faithfully simulate behavior but the probabilistic form (i.e., the likelihood) of its outputs cannot be described analytically. In this chapter, we introduce our framework for amortized Bayesian parameter estimation which we coined *BayesFlow*. It comprises a new Bayesian solution to the simulation-only setting in terms of *invertible neural networks* and the theory of *normalizing flows*. The main idea behind BayesFlow is to split Bayesian analysis into a potentially expensive upfront training phase, followed by a much cheaper inference phase. The goal of the upfront training phase is to train a neural sampler that yields well-calibrated posteriors for *any* observed data set from the generative scope of a model. Subsequently, applying the neural sampler to specific observations during inference is very fast and can easily be performed in parallel, so that the training effort amortizes over repeated evaluations. The following chapter describes the mathematical details behind BayesFlow and discusses its strengths, limitations and future enhancements.

5.1 Desiderata

We have seen in the previous chapter that simulation-based methods need to optimize multiple, often conflicting, requirements concerning their performance. We therefore commence this chapter by stating the concrete desiderata for the utility of our framework:

1. Fully Bayesian estimation without framework-imposed restrictions on the type of priors, simulators, and posteriors amenable for inverse inference
2. Automatic extraction of maximally informative data representations instead of reliance on potentially suboptimal hand-crafted summary statistics
3. Scalability to high-dimensional problems (regarding both the data space $\mathcal{X}$ and the parameter space Θ) via algorithmic alignment and considerations on probabilistic symmetry
4. Full amortization over multiple empirical data sets and data set sizes
5. Parallel computing and GPU acceleration applicable to forward inference (simulations), training, and inverse inference

6. Low memory demands, both during training (i.e., through online learning and on-the-fly simulations) and after training (i.e., no need to store large grids, reference tables or parameter databases)

7. A theoretical guarantee for convergence of the approximate posterior to the true posterior under certain optimal conditions

Throughout this chapter, we will gradually introduce our BayesFlow framework and discuss how it addresses each desideratum. Along the way, we will point out unexplored conceptual or empirical territories and lay out ideas for potential future improvements and applications.

5.2 Background

5.2.1 Deep Generative Modeling

As previously mentioned, BayesFlow draws on major advances in modern deep generative modeling, also referred to as deep probabilistic modeling. The core idea behind deep generative modeling is to represent a complicated target distribution as a non-linear transformation of some simpler latent distribution (e.g., Gaussian or uniform), a so called pushforward. Density estimation of the target distribution, a very complex task, is thus reduced to learning a non-linear transformation, a task that is ideally suited for gradient-based neural network training via standard backpropagation. Typically, deep generative models approximate the target distribution by sampling from the simpler latent distribution and applying the (inverse) transformation learned during gradient-based optimization. Consequently, we will train our neural networks to sample from intractable posterior distribution over the parameters of complex (behavioral) simulators.

Deep generative methods have demonstrated tremendous successes in applications dealing with very high-dimensional data, such as images, texts, or videos [88, 170, 175]. To draw an equivalent between these applications and simulation-based inference, consider a prototypical generative task in computer vision. In this context, the target distribution runs over the pixels of an image, and estimating a generative model of this distribution poses a major challenge. Conditional image generation, an even more challenging task, involves modeling the distribution over pixels contingent on additional information, such as image categories and descriptions (captions). Notably, the same ideas can be seamlessly transferred to model-based Bayesian inference, where the associated challenge is estimating the distribution over model parameters contingent on some observed data. Moreover, recent work demonstrating excellent generative performance with high-resolution images [88] suggests that deep generative models might be excellent candidates for overcoming the curse of dimensionality from which standard simulation-based methods notoriously suffer [27].

5.2.2 Forward Inference

BayesFlow depends on the ability to (efficiently) perform forward inference, that is, simulate artificial data sets given possible parameter configurations. Moreover, in order to use the simulator as a calibrator for the neural networks involved in BayesFlow, we need to simulate multiple data sets from the Bayesian joint distribution $p(\boldsymbol{\theta}, \boldsymbol{x}_{1:N}) = p(\boldsymbol{\theta})\, p(\boldsymbol{x}_{1:N} \mid \boldsymbol{\theta})$, which, for memoryless models, decomposes further into $p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta})$. As already mentioned, the decomposition of the joint distribution is essentially a generative receipt, which can be carried out with or without an explicit likelihood (due to Equation 2.4).

Corresponding to the previous description, Algorithm 1 describes the steps for generating a batch of synthetic data sets using simulations from a memoryless model and a randomly drawn data size N for each batch. The data-set size will typically be drawn from a discrete uniform distribution, $p(N) = \mathcal{U}(N_{\min}, N_{\max})$, but fixed sizes or different distributions are possible (and reasonable for certain applications). Note also, that the algorithm is trivially extendable to stateful models by including memory variables or an explicit dependence of the simulator on previous outputs.

Algorithm 1 Monte Carlo generation of B synthetic data sets

Require: $g(\boldsymbol{\theta}, \boldsymbol{\xi})$ - stochastic model simulator, $p(\boldsymbol{\theta})$ - prior over model parameters, $p(\boldsymbol{\xi})$ - noise distribution, $p(N)$ - distribution over data set sizes, B - number of data sets to generate.

```
Draw data set size: N ~ p(N).
for b = 1, ..., B do
    Draw model parameters from prior: θ^(b) ~ p(θ).
    for n = 1, ..., N do
        Sample noise instance: ξ_n ~ p(ξ).
        Run simulator to obtain n-th synthetic observation: x_n = g(θ^(b), ξ_n).
    end for
    Store pair (θ^(b), x^(b)_{1:N}).
end for
Return mini-batch D_N^(B) := {θ^(b), x^(b)_{1:N}}_{b=1}^B.
```

Importantly, the efficiency of Algorithm 1 depends highly on its actual implementation. The naive complexity of the data-generation algorithm is at least $\mathcal{O}(N * B * G)$, where G denotes the cost of executing the simulator once to obtain a single synthetic observation $\boldsymbol{x}_n$. Thus, the algorithm can benefit from three levels of parallelism: i) over the number of data sets (B); ii) over the number of observations in each data set (N); iii) and over the computational steps of the simulator itself (G). In the ideal case where all levels can be executed in parallel, the computational complexity reduces to $\mathcal{O}(1)$. For some applications, even parallelizing the most costly level can bring about a significant speedup in practice. Notably, the parallelization of data generation with stateful models is generally not immediately obvious due to the sequential dependence of each output on previous outputs (the loop over N), but multiple data sets from a stateful model can still be readily generated in parallel (the loop over B).

5.2.3 Normalizing Flows

In order to perform neural density estimation, we will implement a *normalizing flow* via an invertible neural network (INN, [37, 38]). A normalizing flow represents a transformation of a simple probability density (e.g., Gaussian) into a more complex (unknown) density by a sequence of invertible and differentiable mappings [38]. In contrast to variational methods [12], flow-based methods can perform asymptotically exact inference by using lossless compression. Additionally, they scale favourably from simple low-dimensional problems to (potentially intractable) high-dimensional distributions with complex dependencies [4, 88].

To set the stage, let $\boldsymbol{z} \in \mathbb{R}^D$ be a random variable with a known (simple) probability density and $\boldsymbol{\theta} \in \mathbb{R}^D$ a random variable with an unknown (complicated) probability density. Let $f : \mathbb{R}^D \to \mathbb{R}^D$ be an invertible, differentiable function such that $\boldsymbol{z} = f(\boldsymbol{\theta})$ and $\boldsymbol{\theta} = f^{-1}(\boldsymbol{z})$. By using the change of variable rule of probability theory, the density of the variable $\boldsymbol{\theta}$ can be computed as:

$$p(\boldsymbol{\theta}) = p(\boldsymbol{z} = f(\boldsymbol{\theta})) \left| \det \left(\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right) \right| \tag{5.1}$$

In our framework, we use a unit Gaussian as a base distribution, $p(\boldsymbol{z}) = \mathcal{N}_D(\boldsymbol{z} \,|\, \mathbf{0}, \mathbb{I})$, and the pushforward density will be the posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x})$ over model parameters $\boldsymbol{\theta}$ given a single (for now) observation $\boldsymbol{x}$. Thus, our aim is to learn an approximate posterior q which matches the pushforward posterior. Accordingly, we reparameterize the approximate posterior in terms of a conditional invertible neural network (cINN) estimator $f_{\boldsymbol{\phi}}$ which implements a normalizing flow between $\boldsymbol{\theta}$ and $\boldsymbol{z}$ given observation $\boldsymbol{x}$:

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}) = p(\boldsymbol{z} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})) \left| \det \left(\frac{\partial f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})}{\partial \boldsymbol{\theta}} \right) \right| \tag{5.2}$$

Accordingly, sampling from the approximate posterior involves sampling from the base density and transforming the sample via the inverse operation of the cINN into the pushforward posterior:

$$\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}) \iff \boldsymbol{\theta} = f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{z}; \boldsymbol{x}) \text{ with } \boldsymbol{z} \sim \mathcal{N}_D(\boldsymbol{z} \,|\, \mathbf{0}, \mathbb{I}) \tag{5.3}$$

Thus, our solution to the task of simulation-based parameter estimation is to train a cINN which approximates the true posterior for all possible observations $\boldsymbol{x}$ arising from a given model $\mathcal{M}_j$ as accurately as possible. There are many ways to implement a cINN in practice, and we will illustrate our preferred architecture below based on coupling flows. Note, however, that the field of deep generative modeling is rapidly expanding, with novel architectures emerging almost on a daily basis. Thus, it is likely that the concrete cINN architecture proposed in this work might be supplanted by a better candidate in the not-so-distant future. In any case, the problem of inverse inference in science is here to stay, so the BayesFlow framework could easily be adapted to incorporate a different neural sampler $f_{\boldsymbol{\phi}}$.

5.2.4 Coupling Flows

Coupling flows are one of the most widely used invertible architectures [92] because they are i) conceptually simple; ii) easily invertible; and iii) able to represent highly expressive transformation with tractable Jacobian determinants [88]. A coupling flow $C_{\boldsymbol{\phi}} : \mathbb{R}^D \to \mathbb{R}^D$ with trainable parameters $\boldsymbol{\phi}$ between $\boldsymbol{\theta} \in \mathbb{R}^D$ and $\boldsymbol{z} \in \mathbb{R}^D$ can be realized as follows. Consider a disjoint partition of the input parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ into two subspaces: $(\boldsymbol{\theta}^{\mathcal{A}}, \boldsymbol{\theta}^{\mathcal{B}}) \in \mathbb{R}^d \times \mathbb{R}^{D-d}$. Input partitioning is required by most coupling flows to ensure invertibility [92]. The invertible forward transformation of a coupling flow can be defined as:

$$\boldsymbol{z}^{\mathcal{A}} = C_1(\boldsymbol{\theta}^{\mathcal{A}}; \Omega_1(\boldsymbol{\theta}^{\mathcal{B}})) \tag{5.4}$$
$$\boldsymbol{z}^{\mathcal{B}} = C_2(\boldsymbol{\theta}^{\mathcal{B}}; \Omega_2(\boldsymbol{z}^{\mathcal{A}})) \tag{5.5}$$
$$\boldsymbol{z} = (\boldsymbol{z}^{\mathcal{A}}, \boldsymbol{z}^{\mathcal{B}}) \tag{5.6}$$

where C_1 and C_2 are invertible functions and Ω_1 and Ω_2 are called *conditioners*. The conditioners can be realized via arbitrarily complex functions (e.g., deep neural networks) which themselves need not be invertible, as long as C_1 and C_2 are (easily) invertible. Correspondingly, the inverse transformation $C_{\boldsymbol{\phi}}^{-1}$ of a coupling flow is defined as:

$$\boldsymbol{\theta}^{\mathcal{B}} = C_2^{-1}(\boldsymbol{z}^{\mathcal{B}}; \Omega_2(\boldsymbol{z}^{\mathcal{A}})) \tag{5.7}$$
$$\boldsymbol{\theta}^{\mathcal{A}} = C_1^{-1}(\boldsymbol{z}^{\mathcal{A}}; \Omega_1(\boldsymbol{\theta}^{\mathcal{B}})) \tag{5.8}$$
$$\boldsymbol{\theta} = (\boldsymbol{\theta}^{\mathcal{A}}, \boldsymbol{\theta}^{\mathcal{B}}) \tag{5.9}$$

Note, that there are many viable ways to parameterize a coupling flow [92]. Our BayesFlow method uses a composition of conditional affine coupling layers (cACLs). A single cACL performs the following bijective mapping on its split input

$$\boldsymbol{z}^{\mathcal{A}} = \boldsymbol{\theta}^{\mathcal{A}} \odot \exp(S_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x})) + T_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x}) \tag{5.10}$$
$$\boldsymbol{z}^{\mathcal{B}} = \boldsymbol{\theta}^{\mathcal{B}} \odot \exp(S_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x})) + T_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x}) \tag{5.11}$$
$$\boldsymbol{z} = (\boldsymbol{z}^{\mathcal{A}}, \boldsymbol{z}^{\mathcal{B}}) \tag{5.12}$$

where $\odot$ denotes element-wise multiplication and the functions S_1, S_2, T_1, T_2 are implemented as fully connected (FC) neural networks with $\boldsymbol{x}$ passed through an additional input head. By construction, this bijection works independently of the form of the functions s and t, which themselves are never inverted. The inverse transformation of the cACL is thus given by:

$$\boldsymbol{\theta}^{\mathcal{B}} = (\boldsymbol{z}^{\mathcal{B}} - T_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x})) \odot \exp(-S_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x})) \tag{5.13}$$
$$\boldsymbol{\theta}^{\mathcal{A}} = (\boldsymbol{z}^{\mathcal{A}} - T_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x})) \odot \exp(-S_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x})) \tag{5.14}$$
$$\boldsymbol{\theta} = (\boldsymbol{\theta}^{\mathcal{A}}, \boldsymbol{\theta}^{\mathcal{B}}) \tag{5.15}$$

The Jacobian of the forward coupling transformation is a product of two triangular matrices

$$\frac{\partial C_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \mathrm{diag}(\exp(S_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x})) & \text{finite} \\ 0 & \mathbb{1} \end{bmatrix} \begin{bmatrix} \mathbb{1} & 0 \\ \text{finite} & \mathrm{diag}(\exp(S_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x})) \end{bmatrix}, \tag{5.16}$$

so its determinant is easy to compute and given by

$$\det \boldsymbol{J}_{C_{\boldsymbol{\phi}}} = \exp\left(\sum_{i=1}^{d} S_1(\boldsymbol{\theta}^{\mathcal{B}}; \boldsymbol{x})_i) + \sum_{i=1}^{D-d} S_2(\boldsymbol{z}^{\mathcal{A}}; \boldsymbol{x})_i \right), \tag{5.17}$$

where we have abbreviated the Jacobian of the cACL as $\boldsymbol{J}_{C_{\boldsymbol{\phi}}}$. In section 5.3.1, we will show how to compose multiple cACLs into an invertible network and discuss some additional features for improving the basic design introduced here. Before we introduce our complete Bayesian framework for amortized inference, we briefly peruse the concepts of distribution matching and amortized inference.

5.2.5 Distribution Matching and Amortization

The term amortized inference refers to an approach which reduces the cost of inference by casting some or all inferential phases as an optimization task. In particular, for a given simulator, one can approximate an unknown ground-truth posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ via a parameterized distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})$ by minimizing some f-divergence between the two distributions:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} D_f(p(\boldsymbol{\theta} \mid \boldsymbol{x}) \,||\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})) \tag{5.18}$$

$$= \arg\min_{\boldsymbol{\phi}} \int_{\Theta} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})\, f\left(\frac{p(\boldsymbol{\theta} \mid \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})}\right) d\boldsymbol{\theta} \tag{5.19}$$

where f is a convex function. Usually, the Kullback-Leibler ($\mathbb{KL}$) divergence is chosen, so the objective becomes:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \int_{\Theta} p(\boldsymbol{\theta} \mid \boldsymbol{x}) \log \frac{p(\boldsymbol{\theta} \mid \boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})}\, d\boldsymbol{\theta} \tag{5.20}$$

$$= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})] - \mathbb{H}[p(\boldsymbol{\theta} \mid \boldsymbol{x})] \tag{5.21}$$

$$= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})] \tag{5.22}$$

where $\mathbb{H}[p(\boldsymbol{\theta} \mid \boldsymbol{x})]$ in Equation 5.21 is the Shannon entropy of the true posterior and can be dropped from the optimization objective since it does not depend on $\boldsymbol{\phi}$. We will now differentiate between three types of amortized inference, which all leverage the fact that we can generate synthetic datasets via a scientific simulator. We believe that such an explicit distinction is important, given the current abundance of neural network methods for Bayesian inference.

Case-wise amortization In case-wise amortized inference, we perform an optimization loop for each individual observation $\boldsymbol{x}$ (e.g., as in sequential neural posterior estimation, [40, 60]). Thus, in expectation over all possible observations, the criterion can be expressed as:

$$\mathbb{E}_{p^*(\boldsymbol{x})}\left[\min_{\boldsymbol{\phi}_{\boldsymbol{x}}} \mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})]\right] \tag{5.23}$$

and inference is only amortized in the context of individual observations and models. This implies that different neural network parameters $\boldsymbol{\phi}^*_{\boldsymbol{x}}$ are found for each approximate posterior defined by a particular observation $\boldsymbol{x}$ and a particular model $\mathcal{M}_j$. Crucially, the fact that the minimization objective is inside the outer expectation can render inference infeasible when multiple observations and models are available.

Model-wise amortization In the model-wise amortized scenario, optimization is performed globally for the entire range of plausible observations, which involves pulling the minimum operator (min) out of the outer expectation:

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})} \left[\mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})]\right] \tag{5.24}$$

In this case, the training effort amortizes over the entire data range for a given model. This is the main approach taken in the *BayesFlow* method. The training (optimization) phase in the model-wise amortized setting is considerably longer than the training phase in the case-wise amortized setting. However, once optimization has converged to an approximator of $\boldsymbol{\phi}^*$, the resulting neural estimator $f_{\boldsymbol{\phi}^*}$ can be reused for arbitrarily many observations assumed to arise from a given model $\mathcal{M}_j$. In some cases, the break-even in terms of efficiency between case-wise and model-wise amortized inference occurs even after a few observations, without noticeable accuracy degradation [133]. However, when multiple candidate models should be estimated and compared, the training effort can become prohibitively large, since a separate set of neural network parameters needs to be learned for each model.

Meta-amortization In the meta-amortized setting, optimization is performed over all possible models simultaneously, introducing one more expectation into the objective:

$$\min_{\boldsymbol{\phi}_{\mathcal{M}}} \mathbb{E}_{p(\mathcal{M})} \left[\mathbb{E}_{p^*(\boldsymbol{x})} \left[\mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{M})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{M})]\right]\right] \tag{5.25}$$

In this way, an estimator with parameters $\boldsymbol{\phi}^*_{\mathcal{M}}$ (the subscript denoting amortization over the entire model set $\mathcal{M}$) can be reused for inference on multiple observations with an arbitrary number of competing models from a particular research domain or model class. Importantly, such setting can only be useful if the latent parameter spaces are allowed to vary across the models, as competing models in various domains can have widely different parameterizations. Note also, that both the model-wise and meta-amortized setting can employ amortization over different dataset sizes N (to be discussed shortly) as long as the dimensionality of the data summary statistic stays the same, a property which can be of great utility in practice.

Nevertheless, each further amortization step might introduce an *amortization gap*. The issue has been discussed in the context of variational inference [25] and refers to a potential drop in performance as a consequence of optimizing neural network parameters in expectation as opposed to optimizing for each individual observation. However, a potential amortization gap has not been investigated outside the context of variational inference and warrants an empirical assessment in a meta-amortized context.

In a later chapter, we will show how to make the meta-amortized objective tractable by using ideas from the literature on normalizing flows and multi-task learning. For now, however, we focus on model-wise amortization with BayesFlow.

5.3 BayesFlow: Building Amortized Neural Samplers

At a high level, *BayesFlow* [133] incorporates a summary network h and an inference network f to jointly invert a generative Bayesian model. The *summary network* $h(\boldsymbol{x}_{1:N})$ reduces data sets

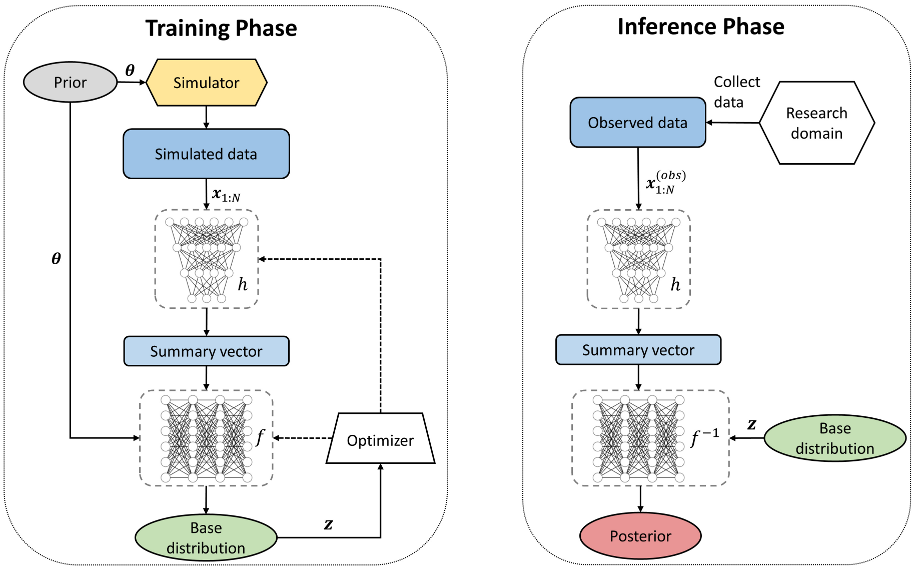

Figure 5.1: Both phases of the BayesFlow framework. Left panel: During the training phase of our BayesFlow method, a summary (h) and an inference network (f) are trained jointly with random draws from the prior and synthetic data from the simulator; Right panel: During the inference phase, BayesFlow works entirely in a feed-forward manner, that is, no training or optimization happens in this phase. The upfront training effort amortizes over arbitrary many observations and data sets from a research domain working on the same model family.

of arbitrary size to fixed-size vector representations. The *inference network* samples from an approximate posterior q via a conditional invertible neural network (cINN) f which implements a normalizing flow between $\boldsymbol{\theta}$ and a normally distributed $\boldsymbol{z}$ given the outputs of the summary network:

$$q(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N}) = p(\boldsymbol{z} = f(\boldsymbol{\theta}; h(\boldsymbol{x}_{1:N}))) \left| \det\left(\frac{\partial f(\boldsymbol{\theta}; h(\boldsymbol{x}_{1:N}))}{\partial \boldsymbol{\theta}} \right) \right| \tag{5.26}$$

where the dependence on all neural network parameters is implicit and has been omitted for clarity. The introduction of a summary network whose structure is aligned to the structure of the simulator (i.e., stateless vs. stateful) frees our framework from a restriction to a particular model class or data type. Moreover, the summary network itself does not have to be invertible, since its output is concatenated with $\boldsymbol{\theta}$ and fed to each coupling layer, but not directly mapped to $\boldsymbol{z}$. Figure 5.1 illustrates the different components and phases of our BayesFlow framework.

5.3.1 Composing Invertible Networks

In this section, we describe how to stack multiple coupling layers to obtain a deep invertible network. For now, consider the case when raw simulated data $\boldsymbol{x}_{1:N}$ of size $N = 1$ is entered directly into the invertible network without using a summary network. In order to ensure that our architecture is expressive enough to encode complex posterior distributions, we chain multiple ACLs, so that the output of each ACL becomes the input to the next one. In this way, the whole network remains invertible from the first input to the last output and can be viewed as a single bijective function. Such chaining of operations is possible, since a composition of invertible functions is itself invertible and its Jacobian determinant is the product of the Jacobian determinants of the individual coupling blocks. Therefore, we refer to a cINN as a composition of K conditional ACLs.

$$\boldsymbol{z} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x}) \equiv C_{\boldsymbol{\phi}_K} \circ C_{\boldsymbol{\phi}_{K-1}} \circ \cdots \circ C_{\boldsymbol{\phi}_1}(\boldsymbol{\theta}; \boldsymbol{x}) \tag{5.27}$$

with trainable parameters $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K)$ and inverse:

$$\boldsymbol{\theta} = f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{z}; \boldsymbol{x}) \equiv C_{\boldsymbol{\phi}_1}^{-1} \circ C_{\boldsymbol{\phi}_2}^{-1} \circ \cdots \circ C_{\boldsymbol{\phi}_K}^{-1}(\boldsymbol{z}; \boldsymbol{x}) \tag{5.28}$$

Note, that the observation $\boldsymbol{x}$ (or a transformation thereof) is fed unchanged to each coupling layer $C_{\boldsymbol{\phi}_k}$. In our applications, the input to the first ACL is the parameter vector $\boldsymbol{\theta}$, and the output of the final ACL is a D-dimensional vector $\boldsymbol{z}$ representing the non-linear transformation of the parameters into $\boldsymbol{z}$-space. Shortly, we will show how to ensure that $\boldsymbol{z}$ follows a unit Gaussian distribution through optimization, that is, we will enforce $p(\boldsymbol{z}) = \mathcal{N}_D(\boldsymbol{z} \,|\, 0, \mathbb{I})$. We also use fixed permutation matrices before each ACL to ensure that each axis of the transformed parameter space $\boldsymbol{z}$ encodes information from all components of $\boldsymbol{\theta}$, in order to capture posterior dependencies (e.g., posterior covariance). In addition, we apply soft clamping of the exponential outputs in each ACL for numerical stability.

Intuitively, our cINN realizes the following process: the forward pass maps data-generating parameters $\boldsymbol{\theta}$ to $\boldsymbol{z}$-space using conditional information from the observation $\boldsymbol{x}$, while the inverse pass maps data points from $\boldsymbol{z}$-space to the data-generating parameters of interest using the same conditional information.

5.3.2 Summary Networks

Since the number of observations might vary in practical scenarios (e.g., different number of trials or time points) or measurements might arrive in streams, we need to perform some form of dimensionality reduction on the data before feeding it to the cINN. As previously mentioned, we want to avoid information loss through restrictive hand-crafted summary statistics and, instead, learn the most informative finite summary vectors directly from data. Therefore, instead of feeding the raw simulated or observed data to the cINN, we pass the data through an auxiliary summary network to obtain a fixed-sized vector representation $\widetilde{\boldsymbol{x}} = h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N})$.

As already alluded to in previous sections, the architecture of the summary network should match the probabilistic symmetry of the observed data (which, in turn, is dictated by the simulator), a property we refer to as *algorithmic alignment* [174]. In other words, different network architectures are needed for exchangeable (generated by memoryless models) and non-exchangeable (generated by stateful models) data. In the following, we illustrate three common scenarios in model-based inference.

Memoryless Models Memoryless models typically generate *i.i.d.* observations, which imply exchangeability and induce permutation invariant posteriors. In other words, changing (permuting) the order of individual elements should not change the associated likelihood or posterior (see

Section 3.4). Memoryless models abide in the cognitive sciences [39, 43, 138, 162], mainly due to their convenient simplicity, but also due to the computational limitations of existing methods for Bayesian estimation. Following [13], we encode probabilistic permutation invariance through functional permutation invariance realized by a deep invariant network. Such a network is capable of learning expressive permutation invariant functions through a combination of *equivariant* and *invariant* transformations.

First, we can obtain a permutation invariant function via an *invariant module* Σ_I which performs an equivariant non-linear transformation h_1 followed by a pooling operator (e.g., sum or max) and another non-linear transformation h_2:

$$\widetilde{x} = \Sigma_I(\boldsymbol{x}_{1:N}) = h_1\left(\sum_{n=1}^{N} h_2(\boldsymbol{x}_n)\right) \tag{5.29}$$

where h_1 and h_2 can be arbitrarily complex neural networks (cf. Figure 5.2, lower left panel). Second, in order to increase the capacity of the invariant transformation, we can stack together multiple *equivariant modules* Σ_E. Each equivariant module implements a learnable equivariant transformation by performing the following operations for each input element $\boldsymbol{x}_n$:

$$\boldsymbol{y}_n = \Sigma_E(\boldsymbol{x}_n, \widetilde{\boldsymbol{x}}) = h_3(\boldsymbol{x}_n, \widetilde{\boldsymbol{x}}) \quad \text{for} \quad n = 1, \ldots, N, \tag{5.30}$$

so that Σ_E is a combination of element-wise (equivariant) and invariant transforms (cf. Figure 5.1, lower middle panel). Again, the internal function h_3 can be parameterized via an arbitrary feed-forward neural network. Importantly, each equivariant module also contains a separate invariant model whose output is concatenated with each observation in order to increase the expressiveness of the learned transformation.

Finally, we can stack multiple equivariant modules followed by an invariant module, in order to obtain a deep invariant summary network $h_{\boldsymbol{\psi}} : \mathcal{X}^N \to \mathbb{R}^S$:

$$\widetilde{\boldsymbol{x}} = h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}) = (\Sigma_I \circ \Sigma_E^{(K)} \circ \Sigma_E^{(K-1)} \circ \cdots \circ \Sigma_E^{(1)})(\boldsymbol{x}_{1:N}), \tag{5.31}$$

where $\boldsymbol{\psi}$ denotes the vector of all learnable neural network parameters and S denotes the dimensionality of the output layer of the last invariant module Σ_I. The complete inference phase of BayesFlow using a deep invariant summary network is depicted in Figure 5.1.

Stateful Models Stateful models incorporate some form of memory and are thus capable of generating observations with complex dependencies. A prime example are dynamic models, which typically describe the evolution trajectory of a system or a process, such as an infectious disease, over time [81]. Observations generated from such models are usually the solution of a stochastic differential equation (SDE) and imply a more complex probabilistic symmetry than those generated from memoryless models.

In a recent application of the BayesFlow framework for estimating key epidemiological parameters [135], we have proposed a set-up specifically designed to tackle dynamic models with simulation-based inference. Our BayesFlow architecture comprises three sub-networks: (i) a convolutional *filtering* network performing noise reduction and feature extraction on the raw time-series data; (ii) a recurrent *summary* network reducing pre-processed time-series of *varying* length

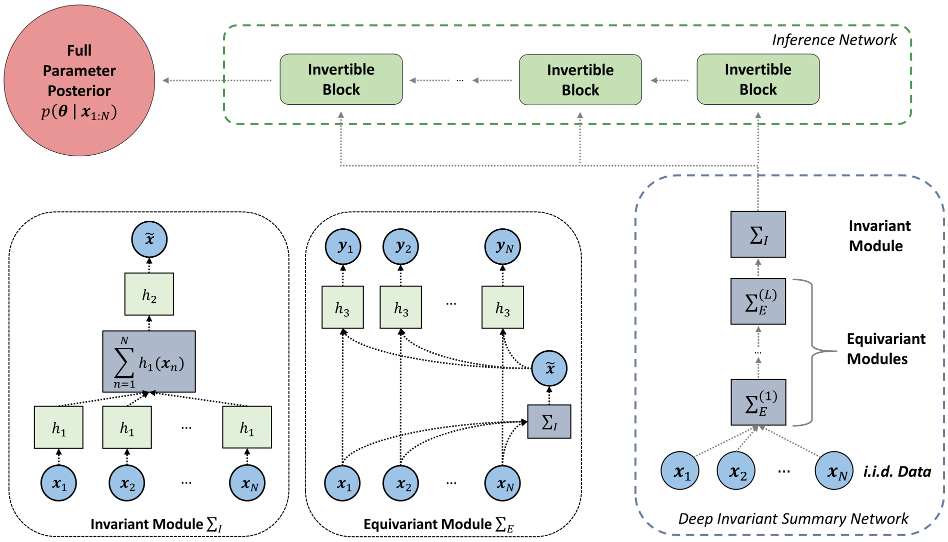

Figure 5.2: Inference with BayesFlow on *i.i.d.* data from a memoryless model using a deep permutation invariant summary network. The summary network is composed of a sequence of flexible equivariant neural modules followed by an invariant neural module. In this way, *i.i.d.* data (sets) of varying length are embedded into fixed-size vector representations which carry maximal information for posterior inference with a memoryless model.

to statistical summaries of *fixed* size; (iii) a cINN inference network performing Bayesian parameter inference given the learned summary vectors of the observations. Figure 5.2 depicts the architecture of this composite network.

The design of the convolutional network is inspired by that of the *Inception* neural architecture which has demonstrated tremendous success in a wide variety of computer vision tasks [152]. In particular, our network is implemented as a deep fully convolutional network which applies adjustable one-dimensional filters of different size at each level (cf. Figure 5.3, lower left panel). The intuition behind this design is that filters of different size might capture patterns at different temporal scales. For instance, if $t = 1, \ldots, T$ is measured in days, a filter of size one will capture daily fluctuations whereas a filter of size seven will capture weekly dynamics. This, in turn, should ease the task of extracting informative temporal features for Bayesian updating.

The output of the convolutional network is a multivariate sequence containing the filtered time-series $\boldsymbol{x}_{1:T}$. In order to reduce the filtered sequence to a fixed-size vector, we pass it through a long-short term memory (LSTM) recurrent network [57]. Importantly, the LSTM network (see Figure 5.3, lower right panel) can deal with sequences of varying length, which enables online learning (i.e., Bayesian updating when new observations become available) and makes the same inference network applicable to settings with different degrees of data availability. Compared to a fixed pooling operation (e.g., mean or max), our many-to-one recurrent network performs a learnable pooling operation which respects the sequential probabilistic symmetry of the data. In this way, the composite summary network learns to filter and extract the most informative fea-

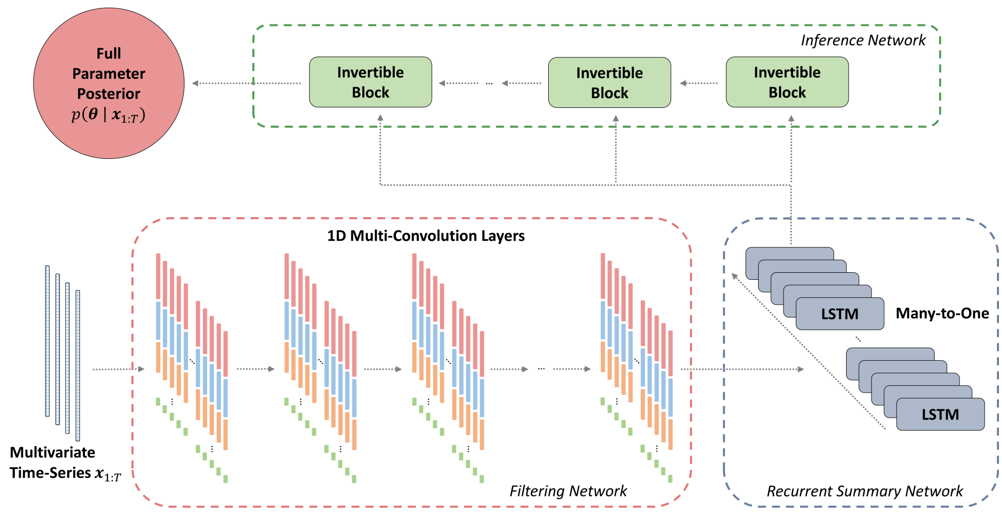

Figure 5.3: Inference on multivariate time-series data arising from a stateful model using a composite summary network architecture. The summary network is composed of an inception-like 1D fully convolutional network, followed by a many-to-one recurrent LSTM network. In this way, time-series of varying length are embedded into fixed-size vector representations which carry maximal information for posterior inference with a stateful model.

tures from the noisy observations in an end-to-end manner, such that no manual (and potentially suboptimal) selection of hand-crafted data features is required from the user at any point.

More formally, let us denote the functions represented by the filtering and summary networks as h_c and h_r. Then, the convolutional filtering network yields a filtered time-series $\widetilde{\boldsymbol{x}}_{1:T'} = h_c(\boldsymbol{x}_{1:T})$ from observed data $\boldsymbol{x}_{1:T}$, where the number of time steps T may vary according to data availability. The recurrent summary network turns the outputs of the filtering network into fixed-size vectors $\widetilde{\boldsymbol{x}} = h_r(\widetilde{\boldsymbol{x}}_{1:T'})$. The cINN thus generates samples $\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \widetilde{\boldsymbol{x}})$ from the parameter posterior by computing $\boldsymbol{\theta} = f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{z}, \widetilde{\boldsymbol{x}})$ with normally distributed random vectors $\boldsymbol{z} \sim \mathcal{N}_D(0, \mathbb{I})$. The complete inference phase of BayesFlow using a deep sequence network is depicted in Figure 5.3.

Joint Models Joint models are an attempt to account for different processes (e.g., neural and cognitive) within a single composite model [30, 118, 159]. Thus, joint models integrate different sources and types of data and require more complex summary architectures. A hypothetical scenario with three data sources (e.g., behavioral data, neural data, and eye-tracking data) is depicted in Figure 5.4. In this case, a separate processor network, each aligned to the particular data type, reduces a separate set of observations from a given source. The outputs of the individual processor networks are then concatenated and fed through an *integrator network*, which combines the information from all processor networks into a single vector representation. In this way, the main cINN architecture can remain the same as in the previous examples and utilize the modularity of neural network. Since the application of integrative joint models is still in its infancy, such composite BayesFlow architectures are yet to prove their usefulness.

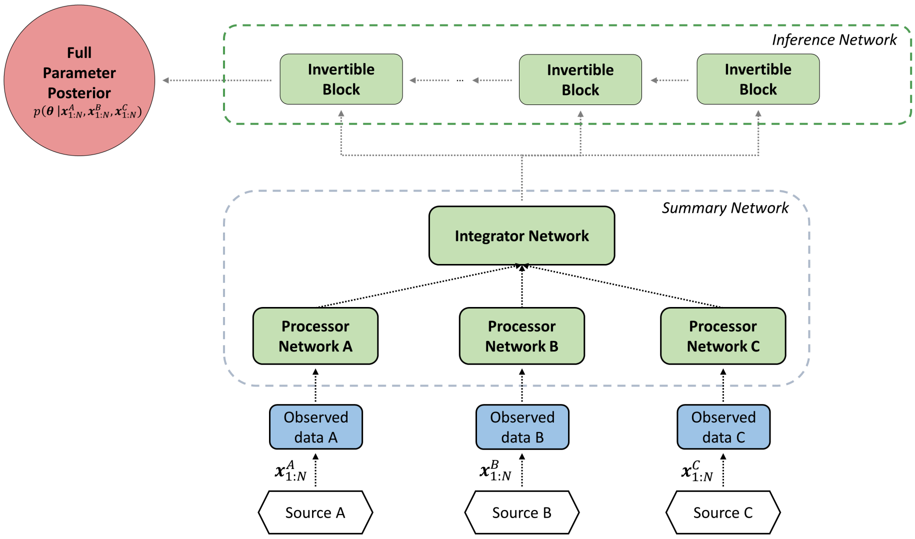

Figure 5.4: Inference on different data sources from a joint model using an ensemble of summary networks and an integrator network. The different data sets, potentially of varying size and structure, are processed by separate algorithmically aligned summary networks. The outputs of all summary networks are then combined into a fixed-size vector representations by the integrator network, which informs the inference network about the joint posterior over all model parameters.

Regardless of the summary network's concrete design, its parameters $\boldsymbol{\psi}$ are optimized jointly with those of the cINN via backpropagation. Thus, the training phase remains completely end-to-end, and BayesFlow learns to generalize to data sets of different sizes by suitably varying N during training (see Algorithm 1).

5.3.3 Optimization Objective

For *any* given (simulated or observed) dataset $\boldsymbol{x}_{1:N}$, our framework needs to ensure that the inverse transformation of the trained cINN, $\boldsymbol{\theta} = f_{\boldsymbol{\phi}}^{-1}(\boldsymbol{z}; h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}))$ with $\boldsymbol{z} \sim \mathcal{N}_D(\boldsymbol{z} \mid 0, \mathbb{I})$, yields samples from the true posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N})$. To achieve this, we resort to the concept of distribution matching introduced earlier and minimize the expected KL divergence between the true and the approximate posterior for all possible observations within the generative scope of a

Monte Carlo simulator. For clarity, we first derive our optimization objective with $N = 1$ and no summary network,

$$
\begin{aligned}
\boldsymbol{\phi}^* &= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}[\mathbb{KL}(p(\boldsymbol{\theta} \mid \boldsymbol{x}) \,||\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x}))] && (5.32)\\
&= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}\left[\mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[\log p(\boldsymbol{\theta} \mid \boldsymbol{x}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})]\right] && (5.33)\\
&= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{p^*(\boldsymbol{x})}\left[\mathbb{E}_{p(\boldsymbol{\theta} \mid \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})]\right] && (5.34)\\
&= \arg\min_{\boldsymbol{\phi}} -\int_{\mathcal{X}} p^*(\boldsymbol{x}) \int_{\Theta} p(\boldsymbol{\theta} \mid \boldsymbol{x}) \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x}) d\boldsymbol{\theta}\, d\boldsymbol{x} && (5.35)
\end{aligned}
$$

which corresponds to model-wise amortization, as defined earlier. To render optimization of this criterion tractable, we first apply the change of variable rule to $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \boldsymbol{x})$ as given in Equation 5.2 to obtain:

$$
\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} -\int_{\mathcal{X}} \int_{\Theta} p^*(\boldsymbol{x}) p(\boldsymbol{\theta} \mid \boldsymbol{x}) \big(\log p(f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})) + \log\left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}\right| \big) d\boldsymbol{\theta}\, d\boldsymbol{x} \tag{5.36}
$$

where we have abbreviated $\partial f_{\boldsymbol{\phi}}(\boldsymbol{\theta}; \boldsymbol{x})/\partial \boldsymbol{\theta}$ (the Jacobian of the entire cINN $f_{\boldsymbol{\phi}}$ evaluated at $\boldsymbol{\theta}$ and $\boldsymbol{x}$) as $\boldsymbol{J}_{f_{\boldsymbol{\phi}}}$ and moved $p^*(\boldsymbol{x})$ inside the inner expectation, as it does not depend on $\boldsymbol{\theta}$. Since Equation 5.36 defines an expectation over the true and unknown data-generating distribution, we replace it with the Bayesian joint model $p(\boldsymbol{\theta}, \boldsymbol{x})$ from which we can obtain Monte Carlo samples (e.g., by using Algorithm 1). Accordingly, for a batch of B parameters and corresponding synthetic data sets $\mathcal{D}^{(B)} = \{(\boldsymbol{\theta}^{(b)}, \boldsymbol{x}^{(b)})\}_{b=1}^{B}$, we can define the following loss function

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\phi}) &= \frac{1}{B} \sum_{b=1}^{B} \left(-\log p\Big(f_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(b)}; \boldsymbol{x}^{(b)})\Big) - \log\left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}^{(b)}\right| \right) && (5.37)\\
&= \frac{1}{B} \sum_{b=1}^{B} \left(\frac{\left\| f_{\boldsymbol{\phi}}\Big(\boldsymbol{\theta}^{(b)}; \boldsymbol{x}^{(b)}\Big) \right\|_2^2}{2} - \sum_{k=1}^{K} \log\left|\det \boldsymbol{J}_{C_{\boldsymbol{\phi}_k}}^{(b)}\right| \right), && (5.38)
\end{aligned}
$$

which we minimize using standard backpropagation to arrive at an unbiased estimate $\widehat{\boldsymbol{\phi}}$ of $\boldsymbol{\phi}^*$. The first term follows from Equation 5.36 due to the fact that we have prescribed a unit Gaussian distribution to $\boldsymbol{z}$. It represents the negative log of $\mathcal{N}_D(\boldsymbol{z} \mid 0, \mathbb{I}) \propto \exp(\left\|-\frac{1}{2}\boldsymbol{z}\right\|_2^2)$. The second term follows from Equation 5.27 and controls the rate of volume change induced by the non-linear transformation from $\boldsymbol{\theta}$ to $\boldsymbol{z}$ learned by $f_{\boldsymbol{\phi}}$. Thus, minimizing Equation 5.38 ensures that $\boldsymbol{z}$ follows the prescribed unit Gaussian and that $f_{\boldsymbol{\phi}^*}$ is a model-wise amortized neural sampler which yields independent samples from the true posterior under perfect convergence [133].

When the number of observations varies during inference, we need to vary it during training as well, in order to achieve amortization over data sets $\boldsymbol{x}_{1:N}$ with different sizes (if required by the application). Thus, we introduce a suitable summary network $h_{\boldsymbol{\psi}}$ which renders the cINN independent of N and learns to extract maximally informative statistics from the (raw) simulated data in an end-to-end manner. Our modified criterion then becomes:

$$\boldsymbol{\phi}^*, \boldsymbol{\psi}^* = \underset{\boldsymbol{\phi}, \boldsymbol{\psi}}{\arg\min}\, \mathbb{E}_{p(N, \boldsymbol{\theta}, \boldsymbol{x})}[-\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}))] \tag{5.39}$$

Accordingly, our modified loss function for a batch $\mathcal{D}_N^{(B)} = \{(\boldsymbol{\theta}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^B$ simulated from the Bayesian model $p(N, \boldsymbol{\theta}, \boldsymbol{x})$ becomes:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \frac{1}{B} \sum_{b=1}^{B} \left(\frac{\left\| f_{\boldsymbol{\phi}}\left(\boldsymbol{\theta}^{(b)}; h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)})\right) \right\|_2^2}{2} - \sum_{k=1}^{K} \log \left| \det \boldsymbol{J}_{C_{\boldsymbol{\phi}_k}}^{(b)} \right| \right), \tag{5.40}$$

which corresponds to a trivial change that simply sets the conditioning vector of the cINN to the output of the summary network. Again, we can use backpropagation with any gradient-based optimization method to obtain unbiased estimates $\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}}$ of the optimal neural network parameters $\boldsymbol{\phi}^*, \boldsymbol{\psi}^*$ from Equation 5.39. Note, that minimizing the above loss function leads to a self-consistent criterion which recovers the true posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N})$ over all $\boldsymbol{x}$ and N under perfect convergence of both networks [133]. However, perfect convergence is often a chimera in practice, so, in a later section, we will discuss the potential sources of errors and respective remedies in detail. Having formulated our optimization criterion, we now describe the different training regimes of BayesFlow.

5.4 Training Phase

The training phase of the BayesFlow framework (left panel of Figure 5.1) can be implemented in different ways, depending on the modeling scenario and the modelers' computational budget. The starting point of all Bayesian analysis is the observed data itself. If a single observed data set $\boldsymbol{x}_{1:N}$ should be analyzed with a complex model that is custom-tailored for this and only this data set, it is worth considering a case-wise amortized approach, such as SNPE [60, 122]. The speed break-even point between case-wise and model-wise amortized inference is application-dependent and currently being investigated [133]. However, at present, a systematic quantitative comparison between different model classes and network architectures is missing from the literature, so modelers need to base their decisions on empirical considerations or pilot simulation studies. Be that as it may, we now present and discuss three viable simulation-based training approaches in the context of model-wise amortization with BayesFlow.

Offline learning The starting point of traditional simulation-based approaches has been the so-called *reference table* $\mathcal{D}^{(S)}$, which is simply a large data structure containing S pairs of parameters and summary statistics of synthetic observations [29, 150]. Indeed, initial machine learning approaches have already recognized the potential of using the reference table as training data for learning algorithms, such as quantile random forests [140] or deep neural networks [79], [136]. In this way, the problem of inverse inference becomes a supervised learning task which can easily be tackled with expressive learning algorithms. With BayesFlow, we can take a similar approach, as outlined in Algorithm 2.

Algorithm 2 BayesFlow training phase using offline learning

Require: $f_{\boldsymbol{\phi}}$ - invertible inference network, $h_{\boldsymbol{\psi}}$ - algorithmically aligned summary network, S - total number of simulations, B - number of simulations per batch (batch size).

1: Generate a large reference table $\mathcal{D}_N^{(S)} := \{\boldsymbol{\theta}^{(s)}, \boldsymbol{x}_{1:N}^{(s)})\}_{s=1}^S$ using Algorithm 1.
2: **repeat**
3: Sample a mini-batch: $\mathcal{D}_N^{(B)} \sim \mathcal{D}_N^{(S)}$.
4: Pass each synthetic data set through the summary network: $\widetilde{\boldsymbol{x}}^{(b)} = h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)})$.
5: Pass each pair $(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$ through the inference network: $\boldsymbol{z}^{(b)} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$.
6: Compute loss according to Equation 5.40 from the training batch.
7: Update neural network parameters $\boldsymbol{\phi}, \boldsymbol{\psi}$ via backpropagation.
8: **until** convergence to $\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}}$
9: **Return** trained inference and summary networks $f_{\widehat{\boldsymbol{\phi}}}, h_{\widehat{\boldsymbol{\psi}}}$.

A few points regarding Algorithm 2 are worth mentioning. First, it involves a single call to Algorithm 1 to generate the entire reference table (step 1), which will return data sets with the same size N if called only once[1]. Thus, if we want to vary N during offline learning, we need to create the reference table via multiple calls to Algorithm 1 and make sure that we have an efficient data structure to store entries with different sizes. Second, steps $3 - 7$ can be executed with GPU parallelization leading to a considerable speed-up in convergence. Third, the convergence criterion can be chosen as in standard deep learning application. For instance, we can establish a pre-defined number of epochs (i.e., loops through the entire training data) or an early stopping condition (i.e., if the loss does not improve in some number of consecutive epochs).

The offline learning regime is particularly useful when active calls to the simulator are computationally expensive, since data generation and training are clearly separated. It also has the advantage of reusing the simulated data and being closest to standard applications of deep learning. Obvious drawbacks of the offline learning regime are the memory demands for storing potentially large and heterogeneous data structures as well as the need to address potential overfitting.

Online learning An alternative to the offline learning regime utilizes the possibility to generate a theoretically limitless number of synthetic data sets on-the-fly. In this way, the networks never "experience" the same inputs (simulated parameters and data sets) twice, since simulations are discarded after each backpropagation update. Moreover, since classical overfitting is nearly impossible in an online learning regime, training can continue as long as the networks keep improving (i.e., the loss keeps decreasing).

Algorithm 3 outlines the online learning regime with BayesFlow. Note, that the key difference to offline training is the fact that learning and data generation are tightly intertwined when performing online learning. The most prominent advantage of online learning is also its most notable disadvantage: since simulations are not reused, the simulator needs to work actively and presents a potential bottleneck. Note also, that this detail presents less of a problem if simulations are computationally cheap or implemented efficiently (e.g., by utilizing different forms of parallelism as discussed in section 5.2.2).

[1] Mathematically, the fixed N scenario is simply a special case where $p(N)$ reduces to a point mass distribution.

Algorithm 3 BayesFlow training phase using online learning

Require: $f_{\boldsymbol{\phi}}$ - invertible inference network, $h_{\boldsymbol{\psi}}$ - algorithmically aligned summary network, B - number of simulations per iteration (batch size).

1: **repeat**
2: Generate a mini-batch $\mathcal{D}_N^{(B)} := \{\boldsymbol{\theta}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^{B}$ using Algorithm 1.
3: Pass each simulated data set through the summary network: $\widetilde{\boldsymbol{x}}^{(b)} = h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)})$.
4: Pass each pair $(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$ through the inference network: $\boldsymbol{z}^{(b)} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$.
5: Compute loss according to Equation 5.40 from the training batch.
6: Update neural network parameters $\boldsymbol{\phi}, \boldsymbol{\psi}$ via backpropagation.
7: **until** convergence to $\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}}$
8: **Return** trained inference and summary networks $f_{\widehat{\boldsymbol{\phi}}}, h_{\widehat{\boldsymbol{\psi}}}$.

Hybrid learning Offline and online learning represent the two endpoints on a hypothetical continuum of training strategies. However, various hybrid learning approaches appear viable for optimizing the total simulation budget available for a given modeling problem. For instance, we can use a technique used widely in reinforcement learning called *experience replay* [98, 148]. Experience replay is a hybrid learning approach aimed at balancing data usage and computational efficiency. It uses a data structure called a circular buffer which keeps past simulations in main memory and discards the oldest once its capacity has been exceeded. We outline this type of hybrid learning in Algorithm 4.

Algorithm 4 BayesFlow training phase using hybrid learning with experience replay

Require: $f_{\boldsymbol{\phi}}$ - invertible inference network, $h_{\boldsymbol{\psi}}$ - algorithmically aligned summary network, S - memory capacity, $\mathcal{F}$ - replay memory buffer, B - number of simulations per iteration (batch size).

1: Initialize replay memory buffer $\mathcal{F}$ with capacity S.
2: **repeat**
3: Generate a mini-batch $\mathcal{D}_N^{(B)} := \{\boldsymbol{\theta}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^{B}$ using Algorithm 1.
4: Store mini-batch $\mathcal{D}_N^{(B)}$ in memory buffer $\mathcal{F}$.
5: Sample a mini-batch $\widetilde{\mathcal{D}}_N^{(B)}$ randomly from $\mathcal{F}$.
6: Pass each pair $(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$ through the inference network: $\boldsymbol{z}^{(b)} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$.
7: Compute loss according to Equation 5.40 from the sampled batch.
8: Update neural network parameters $\boldsymbol{\phi}, \boldsymbol{\psi}$ via backpropagation.
9: **until** convergence to $\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}}$
10: **Return** trained inference and summary networks $f_{\widehat{\boldsymbol{\phi}}}, h_{\widehat{\boldsymbol{\psi}}}$.

To further increase the efficiency when using experience replay, we can introduce a dummy parameter $\alpha \in [0, 1]$ which controls the probability of creating new simulations by executing Algorithm 1. In other words, if $\alpha = 0.5$, new parameters and synthetic observations will be generated in roughly every other pass through lines $3 - 8$, thus reducing the overall number of simulations by a half.

Another hybrid learning approach utilizes a round-based strategy inspired from SNPE methods [40, 60]. Accordingly, the training phase moves through a progression of rounds and each round introduces its own simulation phase. In this way, we keep a reference table in main memory and augment it in a step-wise manner for a pre-defined number of rounds R. Thereby, each round becomes potentially longer but also reuses simulations from all previous rounds. This approach appears preferable to pure offline learning, especially when it is difficult to estimate the required number of simulations in advance. Moreover, an early stopping criterion can be grafted in-between rounds, in case further training is not conductive to the networks' performance. Algorithm 5 lays out the essential steps of the round-based approach.

Algorithm 5 BayesFlow training phase using round-based hybrid learning

Require: $f_{\boldsymbol{\phi}}$ - invertible inference network, $h_{\boldsymbol{\psi}}$ - algorithmically aligned summary network, R - number of rounds, S - number of simulations per round, B - batch size.

Initialize reference table $\mathcal{D}^{(R \times S)} := \{\}$.
for $r = 1, ..., R$ **do**
    Generate synthetic data $\mathcal{D}_r^{(S)} := \{\boldsymbol{\theta}^{(s)}, \boldsymbol{x}_{1:N}^{(s)}\}_{s=1}^{S}$ using Algorithm 1.
    Aggregate data: $\mathcal{D}^{(R \times S)} := \mathcal{D}^{(R \times S)} \cup \mathcal{D}_r^{(S)}$.
    repeat
        Sample a mini-batch: $\mathcal{D}_N^{(B)} \sim \mathcal{D}^{(R \times S)}$.
        Pass each synthetic data set through the summary network: $\widetilde{\boldsymbol{x}}^{(b)} = h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)})$.
        Pass each $(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$ through the inference network: $\boldsymbol{z}^{(b)} = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}^{(b)}, \widetilde{\boldsymbol{x}}^{(b)})$.
        Compute loss according to Equation 5.40 from the sampled batch.
        Update neural network parameters $\boldsymbol{\phi}, \boldsymbol{\psi}$ via backpropagation.
    until convergence to $\widehat{\boldsymbol{\phi}}_r, \widehat{\boldsymbol{\psi}}_r$
end for
Return trained inference and summary networks $f_{\widehat{\boldsymbol{\phi}}_R}, h_{\widehat{\boldsymbol{\psi}}_R}$.

To sum up, one should keep an open mind regarding alternative training regimes which go beyond the ones discussed in this section. The field of neural Bayesian inference is new and, despite being an area of active research, systematic analyses of key practical issues are currently missing from the literature. As we saw in this section, there are various ways to implement the training phase of BayesFlow in practice, each coming with its own advantages and disadvantages. Ideally, the training phase should be structured so as to maximize the performance of the networks while minimizing the number of simulations. Albeit not always easy to achieve in practice, the attainment of this (informal) criterion can greatly benefit from prior considerations on computational resources and domain knowledge of the modeling problem.

5.5 Inference Phase

Once the training phase has completed, the converged BayesFlow networks can be stored on any computer and used for efficient amortized inference on any upcoming data set from the generative scope of the simulator. In other words, the summary and the inference networks have become "domain experts" for Bayesian inference with a particular model family. Moreover, since the price of inference has been pre-paid during the upfront training phase, uncertainty-aware model inversion is now extremely efficient using the pre-trained networks. Indeed, we have extensively demonstrated the efficiency benefits of amortized inference with BayesFlow in our main paper [133]. Throughout the examples considered there, we have shown that we can obtain thousands of samples on hundreds of data sets for a couple of seconds. Algorithm 6 describes the inference phase of BayesFlow (see also Figure 5.1, right panel) on a list of I observed data sets.

Algorithm 6 BayesFlow inference phase with pre-trained networks

Require: $f_{\widehat{\boldsymbol{\phi}}}$ - pre-trained invertible inference network, $h_{\widehat{\boldsymbol{\psi}}}$ - pre-trained summary network, $\{\boldsymbol{x}_{1:N_i}^{(i,obs)}\}_{i=1}^{I}$ - list of observed data sets for inference, L - number of posterior samples.

for $i = 1, \ldots, I$ **do**
  Pass the i-th data set through the summary network: $\widetilde{\boldsymbol{x}}^{(i,obs)} = h_{\widehat{\boldsymbol{\psi}}}(\boldsymbol{x}_{1:N_i}^{(i,obs)})$.
  for $l = 1, \ldots, L$ **do**
    Sample a latent variable instance: $\boldsymbol{z}_l^{(i)} \sim \mathcal{N}_D(\boldsymbol{z} \,|\, 0, \mathbb{I})$.
    Evaluate the inference network in reverse: $\boldsymbol{\theta}_l^{(i)} = f_{\widehat{\boldsymbol{\phi}}}^{-1}(\boldsymbol{z}_l^{(i)}; \widetilde{\boldsymbol{x}}^{(i,obs)})$.
  end for
  Store $\{\boldsymbol{\theta}_l^{(i)}\}_{l=1}^{L}$ as samples from the i-th posterior $p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N} = \boldsymbol{x}_{1:N_i}^{(i,obs)})$.
end for

Note, that all components of Algorithm 6 can also benefit from a tremendous speed-up with the aid of GPU acceleration. In particular, both loops over I and L can be performed in parallel using a GPU. Thus, it seems evident that every step of a BayesFlow analysis pipeline is amenable to modern parallel computing, from Monte Carlo simulations to inference on real data (and also validation, as we will discuss shortly). The correctness of Algorithm 6 is guaranteed under the conditions of perfect convergence and self-consistency, that is:

$$f_{\boldsymbol{\phi}^*}^{-1}(\boldsymbol{z}; h_{\boldsymbol{\psi}^*}(\boldsymbol{x}_{1:N})) \sim p(\boldsymbol{\theta} \mid \boldsymbol{x}_{1:N}) \quad \text{with} \quad \boldsymbol{z} \sim \mathcal{N}_D(\boldsymbol{z} \,|\, 0, \mathbb{I})) \tag{5.41}$$

where $\boldsymbol{\phi}^*, \boldsymbol{\psi}^*$ are global minimizers of the modified criterion (Equation 5.39) and the simulation gap induced by modeling $p^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ via $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \int_\Theta p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ is negligible (see [133] for a detailed proof). Moreover, the samples obtained by perfectly converged BayesFlow networks are fully independent, in contrast to MCMC and other stateful Bayesian methods which sometimes induce severe auto-correlation among successive samples. In practice, however, it is important to be aware of potential deficiencies in computational faithfulness, to which we turn next.

5.6 Sources of Error

Computational faithfulness refers to the adequacy or the ability of a Bayesian method to recover the correct target posterior in a particular (simulated or real-world) modeling scenario. Thus, computational faithfulness is not just a nice-to-have extra, but a crucial prerequisite for trustworthy model-based inference. No Bayesian method is exempt from the privilege of occasionally leading modelers and decision makers astray. Therefore, even though each method will eventually err in some (inevitably unexpected) situation, it seems important to at least have a handy catalogue of errors, listed together with their potential causes and fixes. Such a catalogue does not have to be static, but can dynamically grow as a particular method is continuously used in novel applications or integrated in existing analysis pipelines. In the following, we discuss five prominent sources of error which can potentially compromise faithful Bayesian inference with BayesFlow.

The first source of error is the *simulation gap* which can occur under model misspecification or when the observed data are contaminated in ways not covered by the stochastic component $\boldsymbol{\xi}$ of the simulator. Despite being an issue which needs to be addressed via *prior predictive checks*, that is, *before doing inference*, errors due to model misspecification will result in incorrect posteriors that might be hard to detect in practice. In some cases, model misspecification might manifest itself in posteriors which are incompatible with the prior (e.g., posterior samples having 0 density under the prior), but more complex misbehavior is also possible. In other cases, researchers might anticipate how data will be contaminated (e.g., inattention by participants in an experiment or guesswork during a performance test) and explicitly model the contaminants[2]. However, in most cases, model misspecification will be far from obvious (otherwise one would have taken steps to eliminate it), so its potential to bias subsequent inference remains a real issue. This underlines the importance of domain expertise consistency when setting up a model and highlights the fact, that all steps in a Bayesian workflow are inter-dependent, with errors inherent in initial phases tacitly propagating to further phases of data analysis.

The second source is the Monte Carlo error introduced by necessarily using a finite number of simulations from the joint model $p(N, \boldsymbol{\theta}, \boldsymbol{x})$ to approximate the expectation in Equation 5.39. This source is also referred to as *approximation error* and is a widely accepted concomitant of all Monte-Carlo methods. It is also relatively easy to mitigate in an online learning regime, since, in principle, we can run the simulator as long as we can afford and thus generate a potentially infinite amount of training data. In this respect, neural simulation-based inference is in a better position to exploit the capacity of data-hungry deep neural networks than more prototypical deep learning applications operating a limited-data regime.

The third source is the *amortization gap* which refers to a potential deficiency in the inference phase due to the use of a single set of summary and inference networks parameters $(\widehat{\boldsymbol{\psi}}, \widehat{\boldsymbol{\phi}})$ to perform inverse inference globally (i.e., to obtain model-wise amortization). An amortization gap can be elusive and non-trivial to detect with certainty in practical scenarios unless one performs case-wise inference alongside (which would be wasteful in practice) and quantifies the quality of both analyses. Sometimes, an amortization gap can be detected via probabilistic calibration methods (e.g., simulation-based calibration, SBC, [155]), although the reasons for miscalibrated inference might be obscure at first. The more severe problem with this approach, however, is that miscalibrated inference might have different and overlapping causes, and thus not be directly attributable to an amortization gap. To make matters worse, perfectly calibrated inference on the basis of simulations might still be perfectly miscalibrated when transferred to real data, so, detrimentally, an amortization gap can manifest effects similar to a simulation gap. Thus, proper posterior model checking is needed to ensure a model's generative and predictive performance meet the modeler's needs. In the case of a determinable amortization gap, moving to case-wise amortization might be

[2] In fact, this is what we did in our application of BayesFlow to Covid-19 modeling [135]

a viable option when dealing with complex models. Alternatively, increasing the expressiveness of the summary and inference networks could also ameliorate amortization-related problems (to be discussed shortly).

The fourth source is due to a summary network which may not fully capture the relevant information in the data or when sufficient summary statistics do not exist. All things being equal, not capitalizing on the information contained in the data will result in incorrect inference, usually in the form of overdispersed or otherwise miscalibrated posteriors [133]. Thus, the choice of summary network is a crucial proviso for the overall performance of a BayesFlow application. Indeed, the design and architecture of optimal summary networks is a subject of ongoing research. And even though concrete guidelines for optimal summary network design are currently lacking, there are at least two wells of guidance. On the one hand, recent work on *probabilistic symmetry* [13] and *algorithmic alignment* [174] can provide theoretical ideas on how to select a suitable summary architecture for a particular problem. On the other hand, recent simulation-based applications using the BayesFlow framework to tackle complex stochastic models in different research domains can provide viable empirical hints for aligning the summary network to the data at hand. Currently, the BayesFlow method has been employed to perform inference on complex models from psychology [172], cognitive science [137], computational psychiatry [31], epidemiology [135], mathematical finance [147], and physics [10]. Nevertheless, more theoretical and empirical work is needed for definite recommendations at the current stage of development.

The fifth source is due to an inference network which does not accurately transform the true posterior into the prescribed (Gaussian) latent space. This error can be easily detected by passing multiple simulations through the networks and exploring the structure of the latent space $p(\boldsymbol{z})$. Industrious modelers might even consider computing a formal metric between the desired latent space (e.g., Gaussian) and the one obtained by the networks. In the presence of a mismatch, increasing the capacity of the inference network should be the first step to take before further investigations into the problem. Accordingly, both the depth (number of coupling layers) of the cINN as well as the design of the coupling layers themselves could be tuned to increase the expressiveness of the learned transformation from $\boldsymbol{\theta}$-space to $\boldsymbol{z}$-space. The benefits of neural network depth have been confirmed both in theory and in practice [5, 97], so one should expect better performance in complex settings with increasing network depth. However, one should also bear in mind, that an underexpressive summary network could also be responsible for a deficient transformation, since summary and inference network are optimized jointly during the training phase of BayesFlow. Thus, an exclusive focus on the inference network might not be conductive to solving all possible transformation errors. In any case, visualizing the learned latent space and inspecting it for deviations from the desired one (i.e., as prescribed by the optimization criterion) is integral to any application of BayesFlow.

To sum up, as in any Bayesian framework, care should be taken to ensure computational faithfulness as a basis for reliable amortized inference with BayesFlow. Fortunately, we can address model misspecification (error 1) with standard Bayesian prior/posterior predictive checks [52, 56, 144]. Moreover, we can establish deficiencies in self-consistency (errors 2-5) by simply visualizing the latent space obtained in any application of BayesFlow, which provides us with a self-diagnostic method. Naturally, using this method does not help us pinpoint the exact source of error, but only indicates its potential presence. As previously discussed, certain heuristics can be applied for a more detailed error checking in particular applications. In addition, future research should take steps towards a more fine-grained theoretical error analysis, elucidating the consequences of imperfect convergence and investigating error bounds.

5.7 A Bayesian Workflow with BayesFlow

We will now briefly discuss the place of BayesFlow in a principled Bayesian workflow with a focus on cognitive modeling [144]. In the context of a single cognitive model, a principled Bayesian workflow proposed by [144] goes through the following steps:

1. Prior predictive checks
2. Computational faithfulness checks
3. Model sensitivity checks
4. Posterior predictive checks

Prior predictive checks are designed to test whether a model is consistent with the relevant domain expertise. Computational faithfulness refers to the accuracy of the estimation method. Model sensitivity asks whether the parameters of a model can be recovered given the model's prior specification, generative scope, and algorithmic from. Finally, posterior predictive checks assess whether the model captures the relevant structure of the assumed true data generating process. Needless to say, these steps are all computationally intensive and associated with their own specific challenges. In the following, we describe the significant role of amortized inference with BayesFlow at each step of the Bayesian workflow.

5.7.1 Prior Consistency

Since no inference happens at the (pre-data) stage of ensuring consistency with domain expertise, there is little room for amortized inference either. However, prior predictive checks should be an integral part of any Bayesian (simulation-based) analysis. Inconsistent models can require either a modification of the prior $p(\boldsymbol{\theta})$ or/and the simulator $g(\boldsymbol{\theta}, \boldsymbol{\xi})$, in order to resolve conflicts with self-evident domain expertise. Ideally, cognitive models should (re-)produce meaningful patterns of human behavior and not harness pathological patterns in their generative scope (e.g., superhuman reaction times or flawless memory). The easiest way to control for inconsistent model behavior is to constrain the priors to meaningful domains (numerical ranges). In other cases, incorporating certain constraints into the simulator and extensive exploration of the data space (e.g., via *prior pushforward checks*) might be necessary.

5.7.2 Computational Faithfulness

Computational faithfulness is best ensured when performing Bayesian inference with methods capable of self-diagnosis. For example, convergence issues in MCMC sampling methods in general can be detected by inspecting the Gelman-Rubin ($\widehat{R}$) metric [55] or specific problems with Hamiltonian Monte Carlo (HMC) can be indicated by divergent transitions [8].

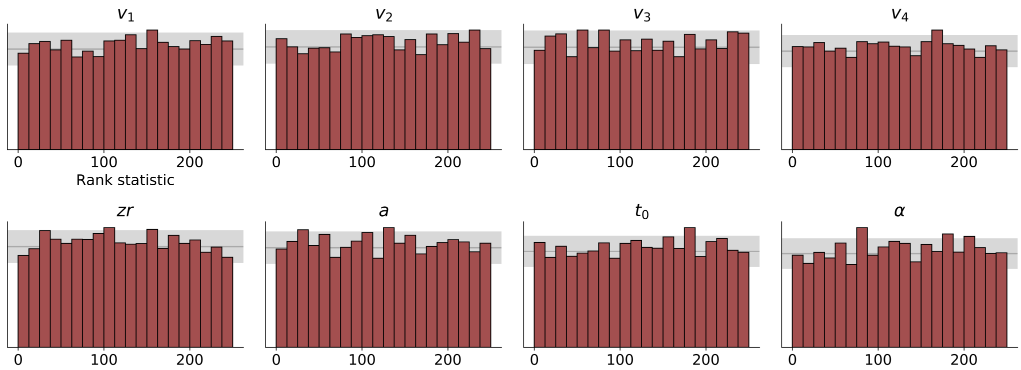

Figure 5.5: Simulation-based calibration (SBC) results for a Lévy flight model with 8 parameters at $N = 800$ trials as a validation check for computational faithfulness. The histograms indicate no systematic deviations from uniformity across marginal posteriors.

A natural self-diagnostic of BayesFlow can be derived by inspecting its the ability to correctly transform $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ into $p(\boldsymbol{z})$ for any $\boldsymbol{x}$. To ensure this, one can simulate a set of pairs $(\boldsymbol{\theta}, \boldsymbol{x})$, pass them through a converged BayesFlow configuration and inspect the resulting latent space for deviations from the prescribed latent space (a spherical Gaussian in our case). This can be done either visually, or numerically, by computing, for instance, the maximum mean discrepancy (MMD, [61]). Note, that this procedure is very fast, since it requires only simulations and forward evaluations of the network, which can all be performed in parallel and furthered through GPU acceleration.

Alternatively, one can resort to calibration algorithms, which can reveal systematic biases in the approximate posteriors. One such approach is simulation-based calibration (SBC, [155]), which is a variant of probabilistic calibration [58] specifically tailored for generative Bayesian models. SBC can be used to validate the inferential correctness of a Bayesian sampling method without knowing the true posterior distribution, which makes it a very powerful diagnostic tool.

However, SBC is extremely time-intensive with standard Bayesian methods, since the computational model needs to be estimated repeatedly, potentially hundreds of times, on different simulated data sets. In addition, the obtained posterior samples should be independent for SBC to yield reliable results, which further increases the required computing time to eliminate auto-correlation via thinning while still retaining enough posterior samples afterwards [155]. These requirements often render SBC practically infeasible for non-amortized Bayesian methods.

Within the BayesFlow framework, SBC can be performed with extreme efficiency once the training phase is over. It simply requires running Algorithm 6 repeatedly with simulated data sets instead of actual observations. Amortized inference ensures that these runs are very efficient. In addition, a perfectly converged BayesFlow configuration yields independent samples from the posterior. Using GPU acceleration, SBC with BayesFlow typically takes a couple of seconds, assuming that the synthetic observations have already been simulated. Thus, we advise the routine and automated use of SBC when doing amortized Bayesian inference.

5.7.3 Model Sensitivity

Model sensitivity, or model adequacy, refers to the feasibility of inverse inference. In other word, it asks about the amount of information gained through Bayesian updating, assuming computational faithfulness of the inferential method and self-consistency of the Bayesian simulator. A straightforward way to obtain a measure of model sensitivity is to compute the expected Bayesian surprise (see Section 3.2), which can also be used for model comparison in a pre-data stage. However, since Bayesian surprise could be hard to interpret in practical terms and without reference to information-theoretic notions, one can resort to other proxies of information gain, such as posterior contraction or posterior z-score [144].

Posterior contraction is a measure of sharpness achieved by Bayesian updating and can be computed for both marginal as well as joint distributions (see Section 3.2). Higher values indicate a high degree of uncertainty reduction and, equivalently, a noticeable posterior sharpness. Likewise, the posterior z-score is a measure of accuracy computed as the difference between the posterior mean (expected value) and the true parameter configuration of a simulated data set, standardized by the posterior variance. Accordingly, smaller values suggest that the posterior concentrates strongly around the true parameter (i.e., the posterior mean is a reasonable representation of the full posterior) while larger values suggest a posterior that concentrates in other parts of the prior domain.

In order to avail themselves of posterior contraction and posterior z-score as useful measures of model sensitivity, modelers need to simulate multiple data sets from the generative model, perform inverse inference on all of them, and compute the corresponding metrics. Similarly to SBC, the feasibility of this procedure depends heavily on the efficiency of the Bayesian estimation method. Thus, evaluating model sensitivity with non-amortized methods might turn out to be prohibitively slow, whereas it becomes trivial when doing amortized inference with a pre-trained BayesFlow configuration. The same would be true for *any* measure of model sensitivity requiring repeated inverse inference on multiple simulated data sets, so model sensitivity is another step of a principled Bayesian workflow which can massively profit from amortized inference.

5.7.4 Posterior Predictive Checks

Posterior predictive checks are vital for evaluating a computational model on actually observed data with respect to the model's generative and predictive performance. Moreover, posterior predictive metrics, such as cross-validation or Bayesian information criteria, can be used for subsequent model comparison and selection in a multi-model setting. As already discussed in Section 3.5, posterior predictive checks comprise a serious computational bottleneck in Bayesian pipelines, even more so when dealing with intractable models.

For instance, k-fold or leave-one-out (LOO) cross-validation (CV) require re-estimating the same model on multiple sub-sets of the original data set in order to approximate out-of-sample predictive performance. When multiple data sets are to be modeled, the computational load increases in a multiplicative manner with the number of data sets B, so extensive posterior predictive checks with standard Bayesian methods quickly become too costly to perform. Once again, amortized inference with BayesFlow offers considerable efficiency gains, since repeated applications of the same model simply involve running the pre-trained networks in a feed-forward mode with different (sub)-sets of the full data.

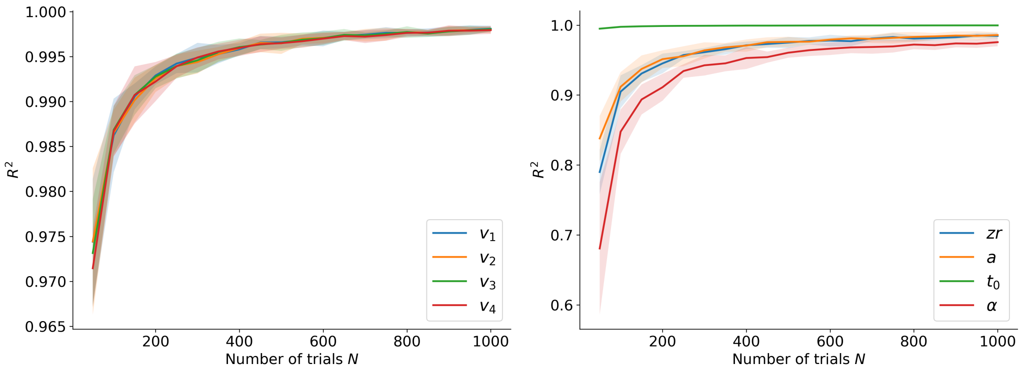

Figure 5.6: The left panel depicts parameter recovery of the four drift rate parameters as a function of trial numbers N using the R^2 metric between true and estimated values. The right panel depicts recovery of the other four parameters. Posterior means are used as summaries of the full posteriors and shaded regions represent bootstrap 95% confidence intervals.

5.8 A Quick Demonstration

As an illustrative example, we present an application of BayesFlow to a recent intractable evidence accumulation model (EAM). Further applications to models from different research domains are described in Chapter 8 or in applied works [10, 31, 135, 147]. EAMs are a popular class of mechanistic models in psychology and cognitive science, since they enable a principled model-based analysis of human response time (RT) data obtainable in controlled experimental environments.

For this example, we focus on a Lévy flight model (LFM) with a non-Gaussian noise assumption [169, 172]. The Lévy flight process is driven by the following stochastic ordinary differential equation (ODE):

$$dx_c = vdt_c + \xi dt^{1/\alpha} \tag{5.42}$$

$$\xi \sim \text{AlphaStable}(\alpha, 0, 1, 0) \tag{5.43}$$

where dx_c denotes accumulated cognitive evidence in condition $c \in \{1, 2, 3, 4\}$, v_c denotes the average speed of information processing (drift) in condition c, and α controls the heaviness of the noise distribution's tails (i.e., smaller values increase the probability of outliers in the accumulation process).

Consider first a simple question of optimal experimental design. A behavioral researcher wants to conduct a response times (RT) experiment with four conditions and model performance via the Lévy flight model. How many trials are needed for accurate parameter recovery? To answer these questions, we can simulate multiple experiments with varying number of trials N per synthetic participant and then compute some practically relevant discrepancy between ground-truth parameters and their estimates. Afterwards, we can quantify computational faithfulness and model sensitivity with the particular number of trials N collected in the experiment. Note, that the mandatory prior predictive and posterior predictive checks are left out for conciseness of exposition.

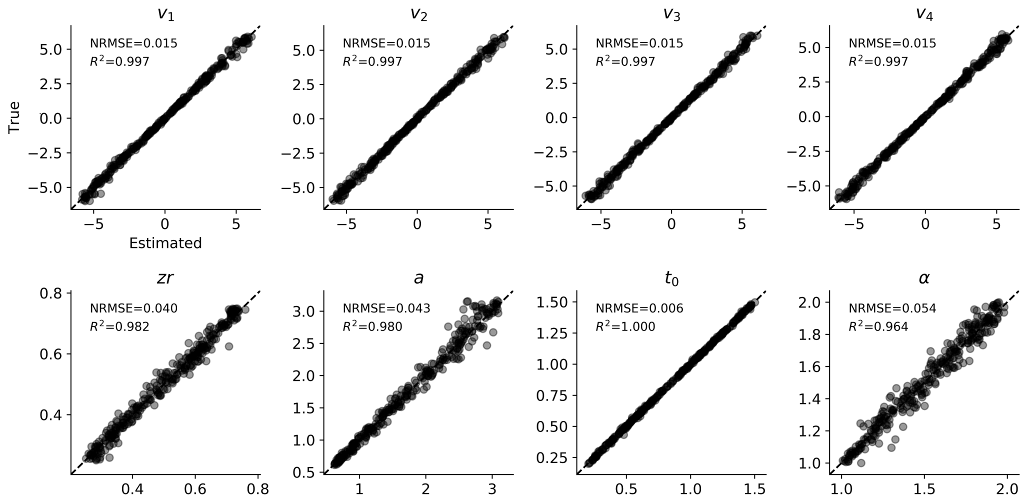

Figure 5.7: Parameter recovery (true vs. estimated) values for $N = 800$ simulated trials. Normalized root-mean-square error ($NRMSE$) and the coefficient of determination (R^2) are used to quantify discrepancy between posterior means and true parameter values.

Since the Lévy flight model is analytically intractable, such a simulation scenario is not feasible with non-amortized methods, which would need weeks on standard machines [169]. However, using a BayesFlow architecture, we can obtain an amortized neural sampler capable of working with variable number of trials (i.e., by using a permutation invariant summary network). The online training phase with Algorithm 3 took approximately one day on a standard laptop equipped with an NVIDIA® GTX1060 graphics card. Subsequent inference is then extremely efficient, as amortized Bayesian estimation on 500 simulated participants takes less than two seconds [137].

We visualize the results by plotting the average R^2 metric obtained from estimating the Lévy flight model on 300 simulated participants with N varying between 50 and 1000 (cf. Figure 5.6. Notably, recovery of the ground-truth parameters via posterior means is nearly perfect at higher trial numbers, and resembles a logarithmic function of N[3]. A similar plot can be created for posterior contraction as a function of N (see Ricker example in [133]).

Further, we can now apply the same network from the previous simulation example for executing fully Bayesian inference on real data. For this illustrative example, we estimate the Lévy flight model from eleven participants performing a long lexical decision task with $N = 800$ trials per condition [137]. Since the task had a 2×2 design, with a factor for *difficulty* (hard vs. easy), and a factor for *stimulus type* (word vs. non-word), we assume a different drift rate v_c for each design cell $c \in \{1, 2, 3, 4\}$.

Before performing inference on actually observed data, we compute SBC and evaluate parameter recovery using simulations with $N = 800$ trials per condition (aligned to the particular experimental design) in order to become a rough sense of computational faithfulness and model sensitivity. Importantly, these checks were performed within seconds with amortized inference and

[3]Strictly speaking, one should also ensure that inference is calibrated for each N, a step which is no more computationally expensive with BayesFlow and which we omit here for brevity.

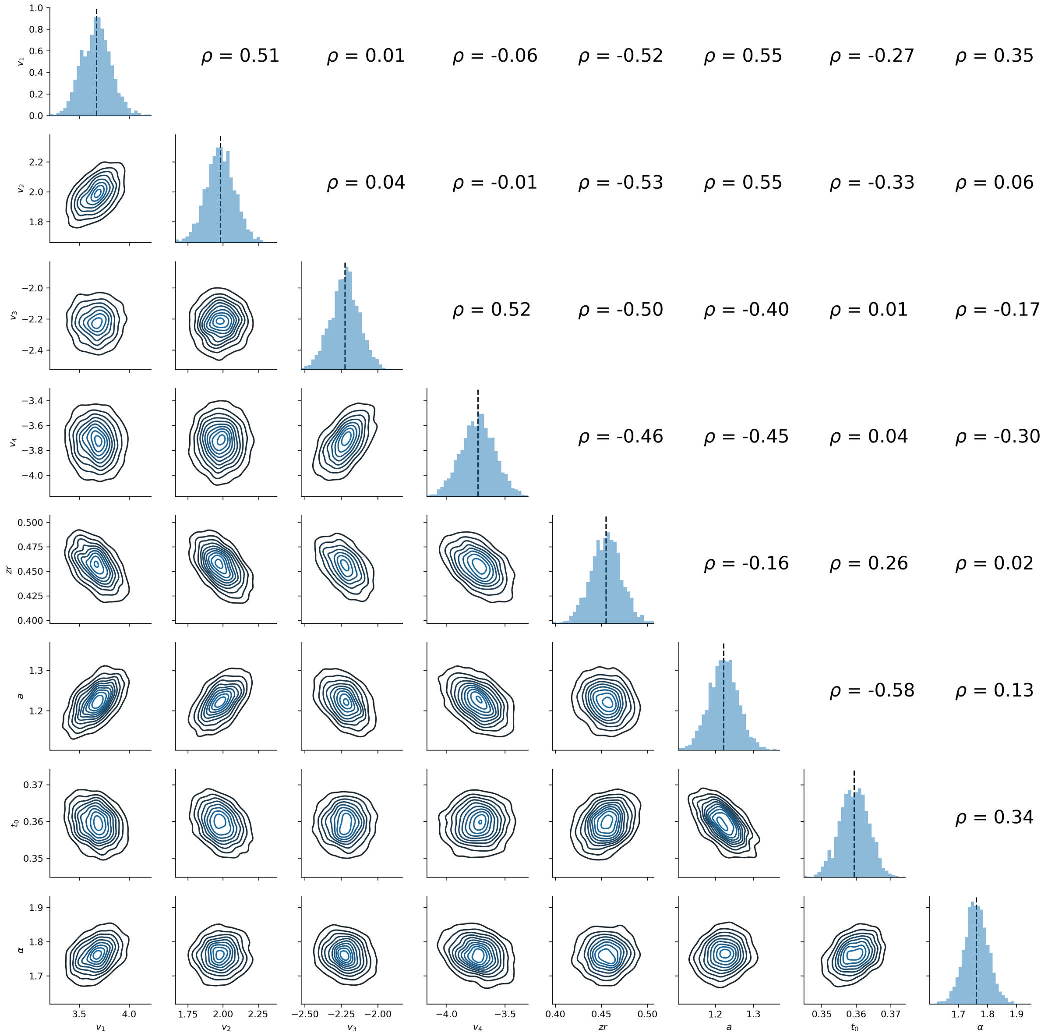

Figure 5.8: Individual bivariate posteriors obtained from data of one example participant in the lexical decision task.

would have been intractable with standard methods. Accordingly, marginal SBC and parameter recovery plots are depicted in Figures Figure 5.5 and Figure 5.7, respectively. The SBC histograms suggest no systematic biases across the approximate marginal posteriors for each parameters (e.g., no under- or overdispersion of the true posterior). Likewise, the recovery plots indicate excellent parameter recovery using posterior means as summaries of the full posteriors, a result which is also evident from the earlier Figure 5.6.

Thus, we can interpret the results from these pre-data checks as hints of intact self-consistency and proceed to applying BayesFlow to real data. A typical output from applying BayesFlow to a single data set is depicted in Figure 5.8, which presents marginal and bivariate posteriors. The latter allows us to visually inspect posterior correlations as indicators of disentanglement (linear independence) between the individual model parameters [137].

5.9 Concluding Remarks

This chapter introduced the building blocks of our BayesFlow framework and discussed its mathematical and algorithmic formulation at a relatively high level. More details regarding perfect convergence, training, and hyperparameter choice (e.g., learning rate, optimizer settings) can be found in our methodological work [133] as well as in the applied works [10, 31, 135, 147]. Details regarding implementation as well as templates for parameter estimation are also available at the corresponding code repository (`https://github.com/stefanradev93/BayesFlow`). Whereas a multitude of features and potential improvements remain to be explored in future research, our results from initial simulations and applications appear highly promising. Thus, we hope that our framework will accelerate model-based inference in a variety of scientific fields and prove its utility beyond the current applications.

6 Amortized Model Comparison

When you have eliminated the impossible, whatever remains, however improbable, must be the truth.
— Arthur Conan Doyle

Researchers from various scientific fields face the task of selecting the most plausible theory for an empirical phenomenon among multiple competing theories. Theories in the cognitive and behavioral sciences are also not exempt from being subject to a relentless selection process. As already discussed, rigorous theories are often instantiated as formal models which describe how observable quantities arise from unobservable parameters in the language of mathematics. Focusing on the level of mathematical models, the problem of theory selection then becomes one of *model selection*.

For instance, neuroscientists might be interested in comparing different models describing the spiking patterns revealed by *in vivo* recordings of neural activity [76]. Epidemiologists, on the other hand, might consider different dynamic models for predicting the transmission rate or other characteristics of an unfolding infectious disease [167]. Crucially, the preference for one model over alternative models in these examples can have important consequences for research projects or social policies.

Accounting for complex natural phenomena often requires specifying complex models which entail some degree of randomness. Inherent stochasticity, incomplete description, or epistemic ignorance all call for some form of uncertainty awareness. As a further complication, empirical data on which models are fit are necessarily finite and can only be acquired with finite precision. Finally, the plausibility of many non-trivial models throughout various branches of science can be assessed only approximately, through rather costly simulation-based methods [26, 35, 76, 104, 139, 161].

Our evidential method aims to amortize Bayesian model comparison by combining latest ideas from simulation-based inference and uncertainty quantification for building efficient and uncertainty-aware neural classifiers. As such, it is intended to complement the toolbox of simulation-based methods for parameter estimation with crucial model comparison capabilities. Moreover, it incorporates a unique feature for estimation of higher-order uncertainty, which goes beyond the scope of standard ABC methods.

6.1 Desiderata

In the previous chapter, we introduced the nuts and bolts of our BayesFlow method for simulation-based Bayesian parameter estimation. This chapter will present our complementary framework for simulation-based Bayesian model comparison. The next chapter will discuss the potential and challenges inherent in combining both frameworks into a single meta-framework. As with parameter estimation, we begin by stating our desiderata for building a useful model comparison method:

1. Estimated model probabilities should be, at least in theory, calibrated to the true model probabilities induced by an empirical problem
2. Estimated model probabilities should be accurate even for finite or small sample sizes
3. Preference for simpler models (i.e., the probabilistic Occam's razor) should be encoded by the estimated model probabilities
4. The method should be applicable to complex models with implicit likelihoods within reasonable time limits
5. The method should enable full amortization over arbitrarily many models, data sets, and varying data set sizes
6. The method should automatically extract maximal information from the raw data and avoid information loss through insufficient summary statistics of the data

Evidently, the desiderata for model comparison are somewhat overlapping with those stated earlier for parameter estimation. Indeed, in this chapter, we will reuse many of the previous concepts for building algorithmically aligned *summary networks* in the BayesFlow framework.

6.2 Background

The following section will briefly rehearse some of the core concepts related to Bayesian model comparison (see also Chapter 3), thereby setting the stage for the derivation of our evidential framework.

6.2.1 Bayesian Model Comparison

In Bayesian modeling, we typically start with a collection of J competing generative models, which we denoted as $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_J\}$. Each abstract model index $\mathcal{M}_j$ is associated with a generative mechanism g_j, typically realized as a Monte Carlo simulation program, and a corresponding parameter space Θ_j equipped with a prior distribution $p(\boldsymbol{\theta}_j \mid \mathcal{M}_j)$. Ideally, each g_j represents a theoretically plausible stochastic mechanism by which observable behavior $\boldsymbol{x}$ arises from hidden time-invariant parameters $\boldsymbol{\theta}_j$ and independent noise $\boldsymbol{\xi}$:

$$\boldsymbol{x}_n = g_j(\boldsymbol{\theta}_j, \boldsymbol{\xi}_n) \text{ with } \boldsymbol{\theta}_j \in \Theta_j \tag{6.1}$$

where Θ_j is the corresponding parameter space of model g_j and the subscript j explicates that each model might be specified over a different parameter space[1]. We assume that the functional or algorithmic form of each g_j is known and that we have a sample (data set) $\{\boldsymbol{x}_i\}_{i=1}^N := \boldsymbol{x}_{1:N}$ of N

[1] Also the noise distribution $p(\boldsymbol{\xi})$ and noise space Ξ might differ for each model, but we will keep this possibility implicit.

(multivariate) observations generated from an unknown process p^*. The task of Bayesian *model comparison* is to assign a plausibility score (e.g., a posterior probability) to each of the models in $\mathcal{M}$. The task of Bayesian *model selection* is then to choose the model in $\mathcal{M}$ that best describes the observed data by balancing simplicity (sparsity) and predictive performance.

As already discussed in Chapter 3, Bayesian methods for model comparison can be categorized as either posterior predictive or prior predictive approaches [52], with our method falling into the latter category. Posterior predictive approaches are concerned with predicting upcoming observations using models extracted from the available data. In prior predictive approaches, models are conditioned only on prior information but not on the available data. Accordingly, all available data counts as new data for the purpose of prior predictive methods.

To recapitulate, the canonical measure of prior predictive performance is the *marginal likelihood*:

$$p(\boldsymbol{x}_{1:N} \mid \mathcal{M}_j) = \int_{\Theta_j} p(\boldsymbol{x}_{1:N} \mid \boldsymbol{\theta}_j, \mathcal{M}_j)\, p(\boldsymbol{\theta}_j \mid \mathcal{M}_j)\, d\boldsymbol{\theta}_j \tag{6.2}$$

which forms the basis for the computation of Bayes factors and posterior odds between pairs of competing models. If two models are equally likely *a priori*, the posterior odds equal the Bayes factor. Furthermore, if the Bayes factor, or, equivalently, the posterior odds equal one, the observed data provide no decisive evidence for one of the models over the other. However, a relative evidence of one does not distinguish whether the data are equally likely or equally unlikely under both models, as this is a question of absolute evidence. Needless to say, the distinction between relative and absolute evidence is of paramount importance for model comparison, so we address it in the next section on model comparison frameworks.

6.2.2 $\mathcal{M}$-Frameworks

Closely related to the distinction between relative and absolute evidence is the distinction between $\mathcal{M}$-closed and $\mathcal{M}$-complete frameworks [176]. Under an $\mathcal{M}$-closed framework, the true model is assumed to be in the predefined set of competing models $\mathcal{M}$, so relative evidence *is* identical to absolute evidence. Under an $\mathcal{M}$-complete framework, a true model is assumed to exist but is not necessarily assumed to be a member of $\mathcal{M}$. However, one still focuses on the models in $\mathcal{M}$ due to computational or conceptual limitations[2].

Deciding on the particular $\mathcal{M}$-framework under which a model comparison problem is tackled is often a matter of prior theoretical considerations. However, since in most non-trivial research scenarios $\mathcal{M}$ is a finite set and candidate models in $\mathcal{M}$ are often simpler approximations to the true model, there will be *uncertainty* as to whether the observed data could have been generated by one of these models. In the following, we will refer to this uncertainty as *epistemic uncertainty*. Our method utilizes a data-driven way to calibrate its epistemic uncertainty in addition to the model probabilities through simulations under an $\mathcal{M}$-closed framework.

Consequently, given real observed data, a researcher can obtain a measure of uncertainty with regard to whether the generative model of the data is likely to be in $\mathcal{M}$ or not. From this perspective, our method lies somewhere in the middle ground between $\mathcal{M}$-closed and an $\mathcal{M}$-complete framework as it provides information from both viewpoints.

[2] See also [176] for discussion of an $\mathcal{M}$-open framework, in which no true model is assumed to exist.

6.2.3 Model Selection as Classification

In line with previous simulation-based approaches to model comparison (e.g., ABC), we will utilize the fact that we can simulate arbitrary amounts of data from each simulator g_j (to be described shortly). Following previous machine learning approaches to model selection [104, 132], we reinterpret the problem of model comparison as a probabilistic classification task. In other words, we seek to learn a mapping $f : \mathcal{X}^N \to \Delta^J$ from an arbitrary data space $\mathcal{X}^N$ to a probability simplex Δ^J containing the multinomial posterior model probability $p(\mathcal{M} \mid \boldsymbol{x}_{1:N})$. Previously, different learning algorithms, such as random forests have been employed to tackle model comparison as classification [104]. Reusing the ideas from algorithmic alignment and probabilistic symmetry incorporated into the BayesFlow framework, our method parameterizes $f_{\boldsymbol{\eta}}$ via a specialized neural network with trainable parameters $\boldsymbol{\eta}$ which is aligned to the probabilistic structure of the generative models (i.e., a permutation invariant network for memoryless models or a recurrent network for stateful models).

In addition, our method differs from previous classification approaches to model comparison in the following aspects. First, it requires no hand-crafted summary statistics, since the most informative summary statistics are learned directly from data. Second, it can make use of online learning (i.e., on-the-fly simulations) which requires no storage of large reference tables or data grids. Third, the addition of new competing models does not require changing the architecture or re-training the network from scratch, since the underlying data domain remains the same. In line with the transfer learning literature, only the last layer of a pre-trained network needs to be changed and training can be resumed from where it had stopped. Last, our method is uncertainty-aware, as it returns a higher-order distribution over posterior model probabilities. From this distribution, one can extract both absolute and relative evidences, as well as quantify the model selection uncertainty implied by the observed data (more on this distinction later).

Intuitively, a converged evidential network encodes the probabilistic relationship between data and models through the network's weights. Thus, once trained, the evidential network can be reused to perform instant model comparison on multiple real observations. As mentioned above, the addition of new models requires simply adjusting the pre-trained network, which requires much less time than re-training the network from scratch.

6.2.4 Multi-Model Forward Inference

Our evidential methods requires the ability to implement each candidate model as a simulator and efficiently generate synthetic observations from each model. This process amounts to performing forward inference in a multi-model context and is described in detail in Algorithm 7. Since we only need the simulated data sets and the corresponding model indices, we can run Algorithm 7 repeatedly to construct training batches of the form $\mathcal{D}_N^{(B)} := \{(\boldsymbol{m}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^{B}$ with B simulated data sets of size N and B corresponding one-hot encoded model indices. We can then feed each batch to a specialized neural network which takes as input simulated data with variable sizes and returns a distribution over posterior model probabilities. Note, that similar considerations regarding computational efficiency and parallelism apply as previously discussed in the context of parameter estimation with BayesFlow.

Algorithm 7 Monte Carlo generation of synthetic data sets for model comparison

Require: $p(\mathcal{M})$ - prior over models, $\{p(\boldsymbol{\theta} \mid \mathcal{M}_j)\}$ - list of priors over model parameters, $\{g_j\}$ - list of stochastic simulators, $p(\boldsymbol{\xi})$ - noise distribution, $p(N)$ - distribution over data set sizes, B - number of data sets to generate per iteration (batch size).

Draw data set size: $N \sim p(N)$.
for $b = 1, ..., B$ **do**
  Draw model index from model prior: $\mathcal{M}_j^{(b)} \sim p(\mathcal{M})$.
  Draw model parameters from prior: $\boldsymbol{\theta}_j^{(b)} \sim p(\boldsymbol{\theta}_j \mid \mathcal{M}_j^{(b)})$.
  for $n = 1, ..., N$ **do**
    Sample noise instance: $\boldsymbol{\xi}_n \sim p(\boldsymbol{\xi})$.
    Run simulator j to obtain n-th synthetic observation: $\boldsymbol{x}_n = g_j(\boldsymbol{\theta}_j^{(b)}, \boldsymbol{\xi}_n)$.
  end for
  Encode model index as a one-hot-encoded vector: $\boldsymbol{m}^{(b)} = \text{OneHotEncode}(\mathcal{M}_j^{(b)})$.
  Store pair $(\boldsymbol{m}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})$ in data structure $\mathcal{D}_N^{(B)}$.
end for
Return mini-batch $\mathcal{D}_N^{(B)} := \{\boldsymbol{m}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^{B}$.

6.3 Training Amortized Evidence Approximators

In our model comparison framework, evidential neural networks learn a higher-order uncertainty representation over the model posterior $p(\mathcal{M} \mid \boldsymbol{x}_{1:N})$ in a simulation-based manner. As we did in the BayesFlow framework, we split model comparison into two phases: an expensive training/simulation phase, in which neural network parameters are optimized via standard backpropagation; and a cheap inference phase, in which a pre-trained evidential network is applied to an arbitrary amount of real data sets $\boldsymbol{x}_{1:N}^{(obs)}$. The output of an evidential network is a higher-order distribution, from which we can obtain a vector of probabilities $p_\eta(\boldsymbol{m} \mid \boldsymbol{x}_{1:N}^{(obs)})$ which approximates the true model posterior $p(\mathcal{M} \mid \boldsymbol{x}_{1:N}^{(obs)})$ for any observable $\boldsymbol{x}_{1:N}^{(obs)}$. In addition, we can obtain local uncertainty information which serves as a proxy for absolute evidence.

6.3.1 Evidence Representation

How can we obtain a measure of absolute evidence by considering only a finite number of competing models? Indeed, such an undertaking has the appearance of an ill-posed problem from the very offset. Our approach will be to re-frame the problem as teaching a neural network to respond with *I don't know* when faced with data which could not have been generated by one of the models (i.e., has not been experienced during the simulation-based training phase). In general, however, a purely probabilistic approach is not well-suited for representing a lack of knowledge [74], since even the uniform distribution encodes the belief in *equally likely events*. In contrast, meta-probabilistic approaches propose to use second-order probabilities [80, 146] for representing the absence of any definite knowledge. In a Bayesian setting, we typically lack knowledge regarding the misspecification degree of the candidate models. Thus, our framework can also be viewed as an approach to quantifying model misspecificatin via higher-order uncertainty. In this way,

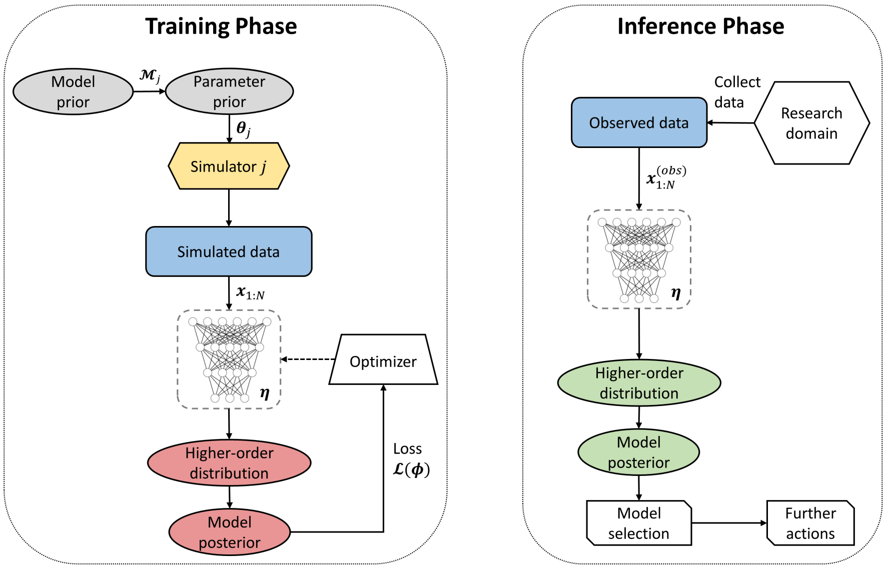

Figure 6.1: Both phases of our evidential model comparison framework. Left panel: During the training phase, an algorithmically aligned evidential network is trained jointly with random draws from the joint prior $p(\mathcal{M}, \boldsymbol{\theta})$ and synthetic data from the simulator; Right panel: During the inference phase, no training or optimization happens in this phase. The upfront training effort amortizes over an arbitrary number of models, observations and data sets from a research domain working on the same model class.

our approach differs from *likelihood-tempering* methods, which require an explicit evaluation of a *tilted* likelihood (raised to a power $0 < t < 1$) in order to prevent overconfident Bayesian updating [63].

In terms of the theory of subjective logic (SL, [80]), we can model second-order probabilities by placing a Dirichlet distribution over the estimated posterior model probabilities [145]. These second-order probabilities represent an uncertainty measure over quantities which are themselves probabilities. We use the second-order probabilities to capture epistemic uncertainty about whether the observed data has been generated by one of the candidate models considered during training.

The probability density function (PDF) of a Dirichlet distribution is given by:

$$\mathrm{Dir}(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=J}^{J} \pi_j^{\alpha_j - 1} \tag{6.3}$$

where $\boldsymbol{\pi}$ belongs to the unit $J-1$ simplex (i.e., $\boldsymbol{\pi} \in \Delta^J := \{\boldsymbol{\pi} \mid \sum_{j=1}^J \pi_j = 1\}$ and $B(\boldsymbol{\alpha})$ is the multivariate beta function [131]. The Dirichlet density is parameterized by a vector of *concentration parameters* $\boldsymbol{\alpha} \in \mathbb{R}_+^J$ which can be interpreted as evidences in the ST framework [80]. The sum of the individual evidence components $\alpha_0 = \sum_{j=1}^J \alpha_j$ is referred to as the Dirichlet strength, and it affects the precision of the higher-order distribution in terms of its variance. Intuitively, the Dirichlet strength governs the *peakedness* of the distribution, with larger values leading to more peaked densities (i.e., most of the density being concentrated in a smaller region of the

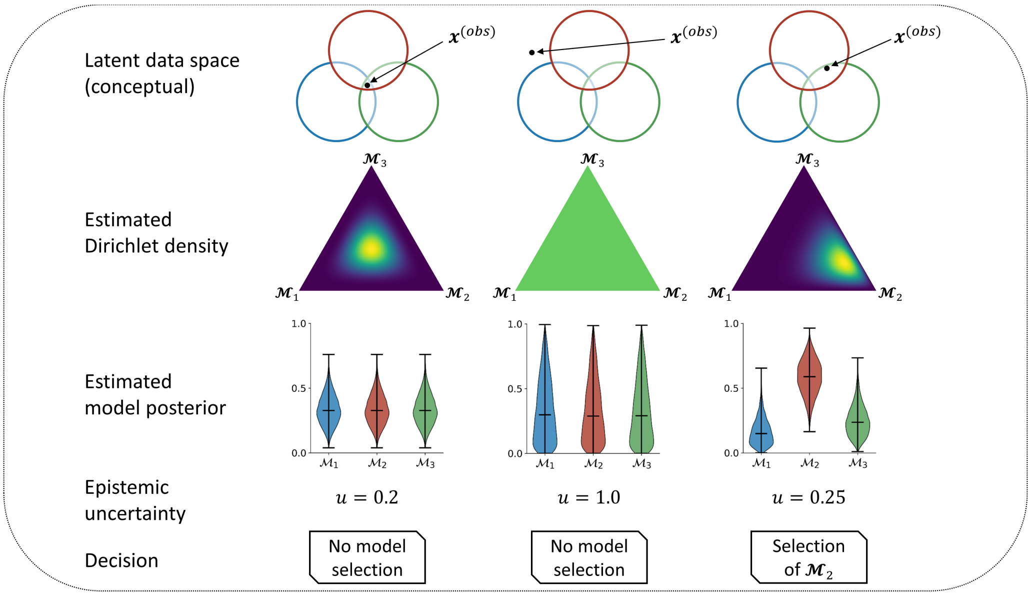

Figure 6.2: Three different hypothetical model comparison scenarios with different observations. The first column depicts observing a data set which is equally probable under all models. The second column depicts a data set which is beyond the generative scope of all models. The third column illustrates an observed data set which is most probable under model 2.

simplex). We can use the mean of the Dirichlet distribution, which is a vector of probabilities given by:

$$\mathbb{E}_{\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\alpha})}[\boldsymbol{\pi}] = \frac{\boldsymbol{\alpha}}{\alpha_0} \tag{6.4}$$

to approximate the posterior model probabilities $p(\boldsymbol{m} \mid \boldsymbol{x}_{1:N})$, as will become clearer later in this section. A crucial advantage of such a Dirichlet representation is that it allows to look beyond model probabilities by inspecting the vector of computed evidences. For instance, imagine a scenario with three possible models. If $\boldsymbol{\alpha} = (5, 5, 5)$, the data provides equally strong evidence for all models (Figure 6.2, first column) – all models explain the data well. If, on the other hand, $\boldsymbol{\alpha} = (1, 1, 1)$, then the Dirichlet distribution reduces to a uniform on the simplex indicating no evidence for any of the models (Figure 6.2, second column) – no model explains the observations well. Note that in either case one cannot select a model on the basis of the data, because posterior model probabilities are equal, yet the interpretation of the two outcomes is very different: The second-order Dirichlet distribution allows one to distinguish between *equally likely* (first case) and *equally unlikely* (second case) models. The last column of Figure 6.2 illustrates a scenario with $\boldsymbol{\alpha} = (2, 7, 3)$ in which case one can distinguish between all models. Later, we will also demonstrate a scenario with data simulated from an actual model.

We can further quantify this distinction by computing an uncertainty score given by:

$$u = \frac{J}{\alpha_0} \tag{6.5}$$

where J is the number of candidate models. This uncertainty score ranges between 0 (total certainty) and 1 (total uncertainty) and has a straightforward interpretation. Accordingly, total uncertainty is given when $\alpha_0 = J$, which would mean that the data provide no evidence for any of the J candidate models. On the other hand, $u << 1$ implies a large Dirichlet strength $\alpha_0 >> J$, which would read that the data provide plenty of evidence for one or more models in question. The uncertainty score corresponds to the concept of *vacuity* (i.e., epistemic uncertainty) in the terminology of SL [80]. We argue that epistemic uncertainty should be a crucial aspect in model selection, as it quantifies the strength of evidence, and, consequently, the strength of the theoretical conclusions we can draw given the observed data.

Consequently, model comparison in our framework consists in inferring the concentration parameters of a Dirichlet distribution given an observed or simulated data set. The problem of inferring posterior model probabilities can thus be reparameterized as:

$$p(\mathcal{M} \,|\, \boldsymbol{x}_{1:N}) \approx q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x}_{1:N}) = \mathbb{E}_{\boldsymbol{\pi} \sim \text{Dir}(f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}))}[\boldsymbol{\pi}] \tag{6.6}$$

where $f_{\boldsymbol{\eta}}$ is a neural network with positive outputs greater than one, that is, $f_{\boldsymbol{\eta}} : \mathcal{X}^N \rightarrow [1, \infty]^J$. Additionally, we can also obtain a measure of absolute model evidence by considering the uncertainty encoded by the full Dirichlet distribution (Eq.6.5). Before elaborating on the latter point, we discuss the main concepts for learning relative evidence, since they form the backbone for further developments.

6.3.2 Learning Evidence in an $\mathcal{M}$-Closed Framework

How do we ensure that the outputs of the neural network match the true unknown model posterior probabilities? As per Algorithm 7, we have unlimited access to samples (simulations) from the joint model $p(\mathcal{M}, \boldsymbol{x}) = \int p(\mathcal{M}, \boldsymbol{\theta}, \boldsymbol{x}) d\,\boldsymbol{\theta}$. Consider, for ease of exposition, a data set with a single observation, that is $N = 1$ such that $\boldsymbol{x}_{1:N} = \boldsymbol{x}$. We use the mean of the Dirichlet distribution $q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x})$ parameterized by an evidential neural network with parameters $\boldsymbol{\eta}$ to approximate $p(\mathcal{M} \,|\, \boldsymbol{x})$. To optimize the parameters of the neural network, we can minimize some loss $\mathcal{L}$ in expectation over all possible data sets:

$$\boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \mathbb{E}_{p(\mathcal{M}, \boldsymbol{x})}[\mathcal{L}(q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x}), \boldsymbol{m})] \tag{6.7}$$

$$= \arg\min_{\boldsymbol{\eta}} \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{E}_{p(\mathcal{M} \,|\, \boldsymbol{x})}[\mathcal{L}(q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x}), \boldsymbol{m})]\right] \tag{6.8}$$

where $\boldsymbol{m}$ is a one-hot encoded vector of the true model index $\mathcal{M}_j$. We also require that $\mathcal{L}$ be a *strictly proper loss* [59]. According to [59], a loss function in the context of simulation-based model comparison is strictly proper if and only if it attains its minimum when $q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x}) = p(\mathcal{M} \,|\, \boldsymbol{x})$.

When we choose the Shannon entropy $\mathbb{H}(q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x})) = -\sum_j q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x})_j \log q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x})_j$ for $\mathcal{L}$, we obtain the strictly proper logarithmic loss:

$$\mathcal{L}(q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x}), \boldsymbol{m}) = -\sum_{j=1}^{J} m_j \log q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x})_j \tag{6.9}$$

$$= -\sum_{j=1}^{J} m_j \log\left(\frac{f_{\boldsymbol{\eta}}(\boldsymbol{x})_j}{\sum_{j'=1}^{J} f_{\boldsymbol{\eta}}(\boldsymbol{x})_{j'}}\right) \tag{6.10}$$

where $m_j = 1$ when j is the true model index and 0 otherwise (i.e., standard one-hot encoding). Thus, in order to estimate ϕ, we can minimize the expected logarithmic loss over all simulated data sets where $f_{\boldsymbol{\eta}}(\boldsymbol{x})_j$ denotes the j-th component of the Dirichlet density given by the evidential neural network. Since we use a strictly proper loss, the evidential network yields the true model posterior probabilities over all possible data sets when perfectly converged.

Intuitively, the logarithmic loss encourages high evidence for the true model and low evidences for the alternative models. Correspondingly, if a data set with certain characteristics can be generated by different models, evidence for these models will jointly increase. Additionally, the model which generates these characteristics most frequently will accumulate the most evidence and thus be preferred. However, we also require low evidence, or, equivalently, high epistemic uncertainty, for data sets which are implausible under all models. We address this problem in the next section.

6.3.3 Learning Absolute Evidence through Regularization

We now propose a way to address the scenario in which no model explains the observed data well. In this case, we want the evidential network to estimate low evidence for all models in the candidate set. In order to attenuate evidence for data sets which are implausible under all models considered, we incorporate a Kullback-Leibler (KL) divergence into the criterion in Eq.6.9. We compute the KL divergence between the Dirichlet density generated by the neural network and a uniform Dirichlet density implying total uncertainty. Thus, the KL shrinks evidences which do not contribute to correct model assignments during training, so an implausible data set encountered in the inference phase will lead to low evidence under all models. This type of regularization has been used for capturing out-of-distribution (OOD) uncertainty in image classification tasks [145]. Curiously, the task of OOD detection closely resembles that of diagnosing model misspecification, so future developments in one of the areas would most likely benefit the other and *vice versa*.

Adding the KL regularization penalty, our modified optimization criterion becomes:

$$\boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \mathbb{E}_{p(\mathcal{M},\boldsymbol{x})}[\mathcal{L}(q_{\boldsymbol{\eta}}(\boldsymbol{m}\,|\,\boldsymbol{x}), \boldsymbol{m}) + \lambda\Omega(\tilde{\boldsymbol{\alpha}})] \tag{6.11}$$

with $\Omega(\tilde{\boldsymbol{\alpha}}) = \mathbb{KL}[\mathrm{Dir}(\tilde{\boldsymbol{\alpha}})\,||\,\mathrm{Dir}(\mathbf{1})]$. The term $\tilde{\boldsymbol{\alpha}} = \boldsymbol{m} + (1 - \boldsymbol{m}) \odot \boldsymbol{\alpha}$ represents the estimated evidence vector after removing the evidence for the true model. This is possible, because we know the true model index sampled from the model prior $p(\mathcal{M})$ during the simulation-based training phase. During the inference phase, knowing the ground truth is not required anymore, since $\widehat{\boldsymbol{\eta}}$ has already been obtained at this point.

The KL regularizer penalizes evidences for the false models and drives these evidences towards unity. Equivalently, it acts as a ground-truth preserving prior on the higher-order Dirichlet distribution which preserves evidence for the true model and attenuates misleading evidences for the false models. The hyperparameter λ controls the regularization weight and encodes the tolerance of the algorithm to accept implausible (out-of-distribution) data sets during inference. With large values of λ, it becomes possible to detect cases where all models are deficient (i.e., misspecified); with $\lambda = 0$, only relative evidence can be generated. Note, that in the latter case, we recover our original proper criterion without penalization. The KL weight λ should be selected through prior empirical considerations on how well the simulations cover the plausible set of real-world data sets.

Importantly, the introduction of the KL regularizer renders the loss no longer *strictly proper*. Therefore, a large regularization weight λ would lead to poorer calibration of the approximate model posteriors, as the regularized loss is no longer minimized by the true model posterior. However, since the KL prior is ground-truth preserving, the accuracy of recovering the true model should not be affected. Indeed, we observe this behavior across a number of simulated experiments. More analytical research is needed on the rate of miscalibration induced by a particular choice of λ.

To make optimization of Equation 6.11 tractable in practice, we utilize the fact that we can easily simulate batches of the form $\mathcal{D}_N^{(B)} = \{(\boldsymbol{m}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^B$ via Algorithm 7 and approximate Eq.6.11 via standard backpropagation by minimizing the following loss:

$$\mathcal{L}(\boldsymbol{\eta}) = \frac{1}{B}\sum_{b=1}^{B}\left[-\sum_{j=1}^{J} m_j^{(b)} \log\left(\frac{f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}^{(b)})_j}{\sum_{j'=1}^{J} f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}^{(b)})_{j'}}\right) + \lambda\Omega(\tilde{\boldsymbol{\alpha}}^{(b)})\right] \tag{6.12}$$

over multiple batches to converge at a Monte Carlo estimator $\widehat{\boldsymbol{\eta}}$ of the optimal neural network parameters $\boldsymbol{\eta}^*$. In practice, convergence can be determined as the point at which the loss stops decreasing, a criterion similar to *early stopping*. Alternatively, the network can be trained for a pre-defined number of epochs. Note, that, at least in principle, we can train the network arbitrarily long, since we assume that we can access the full joint Bayesian distribution $p(\mathcal{M}, \boldsymbol{x}, N)$ through simulation (cf. Figure 6.1, left panel). In practice, early stopping seems to work reasonably well, since it requires no prior considerations on the (most likely unknown) optimal number of simulations or interventions during training.

6.3.4 Implicit Preference for Simpler Models

Perfect convergence of the evidential network for a given model comparison problem implies $q_{\boldsymbol{\eta}}(\boldsymbol{m} \mid \boldsymbol{x}_{1:N}) \propto p(\boldsymbol{x}_{1:N} \mid \mathcal{M})p(\mathcal{M})$. Thus, a perfectly converged evidential network automatically encodes a preference for simpler models (Bayesian Occam's razor). This is due to the fact that we are approximating an expectation over all possible data sets, parameters, and models (i.e., the full Bayesian distribution). Accordingly, a simple model has a narrow generative scope, so data sets generated by a simpler model will tend to be more similar compared to those from a more complex competitor. Therefore, during training, certain data sets which are plausible under multiple models will be generated most often by the simplest model. Thus, a perfectly converged evidential network will capture this behavior by assigning higher posterior probability to the simplest model (assuming equal prior probabilities). Therefore, at least in theory, our method captures complexity differences arising purely from the generative behavior of the models and does not presuppose an *ad hoc* measure of complexity (e.g., number of parameters).

6.3.5 Training and Inference

Algorithm 8 (Online) Training phase and inference phase for amortized Bayesian model comparison with regularization-based uncertainty estimation.

Require: $f_{\boldsymbol{\eta}}$ - evidential neural network, $\{\boldsymbol{x}_{1:N_i}^{(obs)}\}_{i=1}^{I}$ - list of observed data sets for inference, λ - regularization weight, B - number of simulations at each iteration (batch size).

1: *Simulation-based training phase:*
2: **repeat**
3: Generate a training batch $\mathcal{D}_N^{(B)} = \{(\boldsymbol{m}^{(b)}, \boldsymbol{x}_{1:N}^{(b)})\}_{b=1}^{B}$ via Algorithm 7.
4: Compute evidences for each simulated data set in $\mathcal{D}_N^{(B)}$: $\boldsymbol{\alpha}^{(b)} = f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}^{(b)})$.
5: Compute loss according to Equation 6.12.
6: Update neural network parameters $\boldsymbol{\eta}$ via backpropagation.
7: **until** convergence to $\widehat{\boldsymbol{\eta}}$
8: *Amortized inference phase:*
9: **for** $i = 1, ..., I$ **do**
10: Compute model evidences $\boldsymbol{\alpha}_i^{(obs)} = f_{\widehat{\boldsymbol{\eta}}}(\boldsymbol{x}_{1:N_i}^{(obs)})$.
11: Compute uncertainty $u_i = J / \sum_{j=1}^{J} \alpha_{i,j}^{(obs)}$.
12: Approximate true model posterior probabilities $p(\mathcal{M} \,|\, \boldsymbol{x}_{1:N_i}^{(obs)})$ via $q_{\boldsymbol{\eta}}(\boldsymbol{m} \,|\, \boldsymbol{x}_{1:N_i}) = \boldsymbol{\alpha}_i^{(obs)} / \sum_{j=1}^{J} \alpha_{i,j}^{(obs)}$.
13: **end for**
14: Choose further actions.

The training phase in our evidential framework can be carried out using the same ideas and considerations explored in the context of BayesFlow. Accordingly, one can choose between online, offline, or a hybrid learning regime, depending on the computational resources available, the complexity of the candidate models and the simulation budget allocated for performing model comparison. Thus, in order to avoid repetition, Algorithm 8 summarizes both the training and inference phase with our evidential method using online learning during training. Note, that steps 2-7 and 9-13 can be executed in parallel and with GPU support in order to dramatically accelerate convergence and inference. Importantly, if the priors over model parameters change or additional models need to be considered, the parameters $\boldsymbol{\eta}$ of a pre-trained network can be augmented to $\boldsymbol{\eta}'$ by adding additional output nodes for the new models. Training can then be resumed from where it had previously stopped without optimizing $\boldsymbol{\eta}'$ from scratch.

6.3.6 Sources of Error

Since our evidential framework embodies some of the principles implemented in the BayesFlow framework (simulation-based training, amortized inference), it also inherits some of the error sources described in Section 5.6, namely, simulation gap, amortization gap, and approximation error. Most notably, our regularization approach to model misspecification is precisely designed to detect the presence of simulation gaps. Since our evidential networks distils global knowledge about the models' generative scopes, it is supposed to assign low evidences to models which encounter a simulation gap during inference. However, whenever an evidential network is trained to minimize Equation 6.12 with $\lambda = 0$ (i.e., no regularization is applied), simulations gaps remain an undetectable issue, at least until posterior predictive checks are performed. In any case, errors due to an amortization gap (i.e., learning a global neural estimator) and Monte Carlo approximation (i.e., estimating an expectation) remain something to be aware of when performing amortized neural model comparison.

Another source of error is an underexpressive evidential network which is unable to properly encode the probabilistic relationship between data and models. In this case, the evidential network will be poorly calibrated, that is, its outputs would not represent the true posterior distribution $p(\mathcal{M} \mid \boldsymbol{x})$. Fortunately, due to amortized inference, we can easily estimate and visualize the expected calibration error (ECE, [64]) of an evidential network over multiple simulations from $p(\mathcal{M}, \boldsymbol{\theta}, \boldsymbol{x})$. Accordingly, ECE values close to 0 indicate proper calibration of the network. Some model comparison scenarios may prioritize different metrics, such as accuracy or precision/recall ratios, common to classification tasks in machine learning applications.

6.4 A Simulated Experiment

As a brief illustrative example (described in detail in [134]), we applied our evidential method to distinguish between complex nested spiking neuron models describing the properties of biological cells in the nervous system. The purpose of this experiment was twofold. On the one hand, we wanted to assess the ability of our method to classify models deploying a variety of neural patterns accounting for different cortical and sub-cortical neuronal activity. On the other hand, we wanted to investigate the network's ability to detect biologically implausible data patterns, as indexed by our measure of epistemic uncertainty. To this aim, we rely on a renowned computational model of biological neural dynamics.

6.4.1 Model Comparison Setting

In computational neuroscience, mathematical models of neuronal electrical dynamics serve as a basis to explain the functional organization of the brain from both single neuron and large-scale neuronal networks processing perspectives [1, 16, 70, 75]. A multitude of different neuron models have been proposed during the last decades, ranging from completely abstract to biologically plausible models. The former offer a simplified mathematical representation which takes the main functional properties of spiking neurons into account. The latter provide a detailed analogy between models' state variables and ion channels in biological neurons [129]. Importantly, these

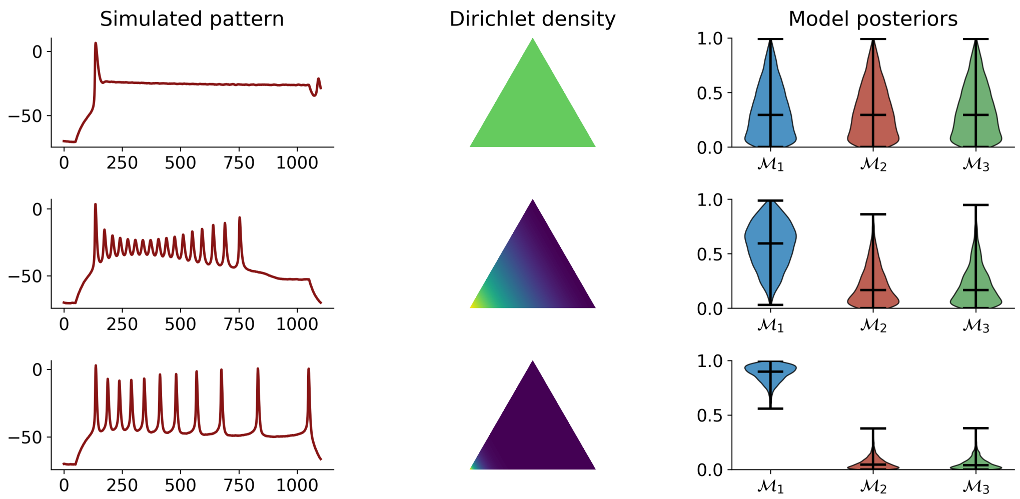

Figure 6.3: Three simulated firing patterns, corresponding estimated Dirichlet densities and model posteriors [134]. Each row represents a different value of the parameter $\bar{g}_K$, $\bar{g}_K = 0.1$, $\bar{g}_K = 0.5$, and $\bar{g}_K = 0.75$, respectively. An increase in the parameter $\bar{g}_K$ is accompanied by a decrease in epistemic uncertainty (as measured via Eq.6.5). An implausible value of $\bar{g}_k$ (first row) results in a flat Dirichlet density as an index of total epistemic uncertainty (uniform green areas). As the parameter value surpasses the plausibility boundary (second and third rows), the Dirichlet simplex becomes peaked towards the lower left edge encoding $\mathcal{M}_1$.

computational models differ in their capability to reproduce firing patterns observed in real cortical neurons [76].

For this simulated experiment, we consider a Hodgkin-Huxley stateful model of cortical and thalamic neurons [70, 130]. The forward model is formulated as a set of five ordinary differential equations (ODEs) describing how the neuron membrane potential $V(t)$ changes over time as a function of the injected current $I_{inj}(t)$ and of various ion channels properties (see [134] for more details regarding the forward model).

To set up the model comparison problem, we treat different types of conductance, g_L, $\bar{g}_{Na}$, $\bar{g}_K$ and $\bar{g}_M$, as free parameters, and define different neural models based on different parameter specifications. In particular, we formulate three models $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}$ defined by the parameter sets $\boldsymbol{\theta}_1 = (\bar{g}_{Na}, \bar{g}_K)$, $\boldsymbol{\theta}_2 = (\bar{g}_{Na}, \bar{g}_K, \bar{g}_M)$, and $\boldsymbol{\theta}_1 = (\bar{g}_{Na}, \bar{g}_K, \bar{g}_M, g_L)$, respectively.

In order to evaluate performance, we train an unregularized recurrent evidential network for 60 epochs resulting in 60000 backpropagation updates. At each iteration, we draw a random input current duration $T \sim \mathcal{U}_D(100, 400)$ (in units of milliseconds), keeping a constant input current, I_{inj}. T reflects the physical time window in which biological spiking patterns can occur. Since the sampling rate of membrane potential is fixed ($dt = 0.2$), T affects both the span of observable spiking behavior and the number of simulated data points.

6.4.2 Validation Results

The entire training phase using online learning (Algorithm 8) took approximately 2.5 hours of wall-clock time. On the other hand, model comparison on 5000 neural time-series simulated for validation took approximately 0.7 seconds, which is a remarkable efficiency gain.

Regarding model selection performance, the network exhibited accuracies above 0.92 across all Ts, with no gains in accuracy for increasing T. This result highlights the fact that even short input currents are sufficient for reliably distinguishing between these complex models. Further, we observed good calibration for all three models, with all ECEs less than 0.1. Notably, we observed no overconfidence for all three models.

In order to assess how well we can detect biologically implausible patterns, we train an identical recurrent evidential network with a gradually increasing regularization weight up to $\lambda = 1.0$. We then fix the parameter $\bar{g}_{Na} = 4.0$ of model $\mathcal{M}_1$ and gradually increase its second parameter $\bar{g}_K$ from 0.1 to 2.0. Since spiking patterns observed with low values of $\bar{g}_K$ are quite implausible and have not been encountered during training, we expect uncertainty to gradually decrease. Indeed, Figure 6.3 shows this pattern. On the other hand, changing the sign of the output membrane potential, which also results in biologically implausible patterns, leads to a trivial selection of $\mathcal{M}_3$. This is contrary to expectations, and shows that absolute evidence is also relative to the model knowledge the evidential network has learned during training. Future research should therefore focus on making the latent space of the evidential network interpretable, in order to make the conceptual visualization from Figure 6.2 tractable.

6.5 Concluding Remarks

This chapter introduced the building blocks of our evidential framework for Bayesian model comparison and discussed its mathematical and algorithmic formulation. In contrast to BayesFlow, applications of our amortization approach to real-world model comparison/selection problems are still underway. One reason for this is that recent developments in the field of simulation-based statistical inference have focused predominantly on parameter estimation and model comparison has often played a secondary role (or has been too costly to perform). Thus, we hope that our framework (or underlying ideas) can enhance and enrich model-based analysis and inference in many fields dealing with competing computational models of complex natural processes. We leave it to future research to investigate whether there are more elegant ways to quantify absolute evidence or detect model misspecification from a simulation-based perspective. Details regarding training, hyperparameter choice, and validation metrics can be found in our methodological paper [134]. Further details regarding implementation as well as templates for model comparison are also available at the code repository (`https://github.com/stefanradev93/BayesFlow`).

7 Meta-Amortized Inference

What is now proved was once only imagined.
— William Blake

In this chapter, we explore the idea of a universal neural Bayesian inference architecture for performing simultaneous parameter estimation and model comparison in a purely simulation-based way. We build on our previous methods by using ideas from *multi-task* learning [20] and meta-amortized variational inference [25]. In this way, we propose to enable and amortize all steps of a Bayesian workflow within a unified framework involving a single training/optimization phase. Instead of presenting a ready-made solution, this chapter merely intends to point out towards a speculative future development aimed at scaling up an entire Bayesian workflow to complex models.

7.1 The Bayesian Hardships

A Bayesian analysis consists of more than just parameter estimation and model comparison. The big picture of Bayesian inference involves a rather significant allocation of creative, computational, financial, and decision making resources (cf. Figure 7.1 for an illustrative overview). Most recently, attempts have been made to systematize Bayesian analysis into a principled, step-by-step workflow reminiscent of a cooking recipe [49, 56, 144]. Naturally, it is beyond the scope of this chapter to review these comprehensive works. Thus, we will attempt to extract the most basic elements of a Bayesian workflow which can directly benefit from the notion of amortized inference.

The starting point of our Bayesian workflow of interest is a collection of observed data sets $\mathcal{D}^{(obs)} = \{\boldsymbol{x}_{1:N_i}^{(obs)}\}_{i=1}^{I}$, with $I = 1$ in the case of a single data set and $N = 1$ in the case of a single observation, and a collection of J competing models $\mathcal{M} = \{\mathcal{M}_j\}_{j=1}^{J}$. Most Bayesian pipelines would go through multiple steps involving, among other things, a considerable computational burden. These steps are represented by the red-shaded boxes in Figure 7.1 and summarized as follows:

1. Parameter estimation
2. Evaluation of computational faithfulness
3. Evaluation of model adequacy/sensitivity
4. Posterior predictive checks
5. Model comparison/model aggregation

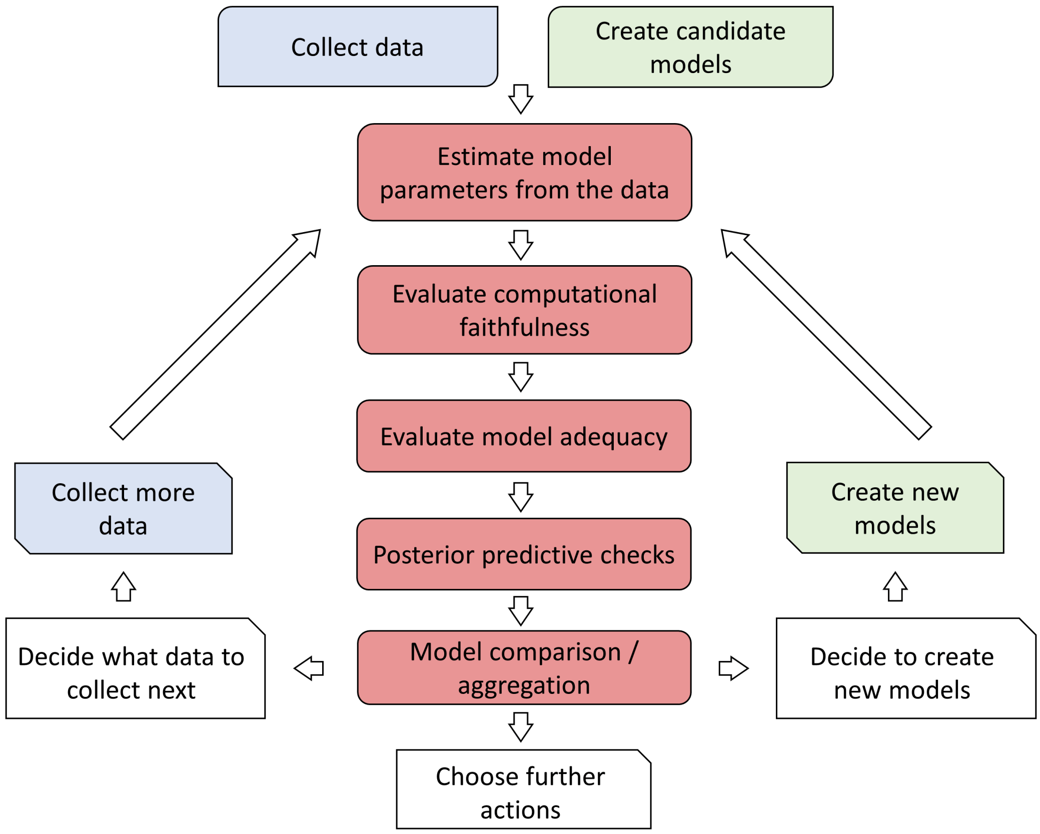

Figure 7.1: The basic conceptual steps of a Bayesian analysis pipeline (workflow) associated with allocation of different types of resources. White indicates allocation of decisional resources. Blue indicates allocation of financial resources. Green indicates allocation of creative resources. Red indicates allocation of computational resources. Especially the latter can profit to a great extent from amortization.

As we saw in the previous two chapters, we can tackle parameter estimation as well as checks of computational faithfulness, model sensitivity, and posterior predictions with our BayesFlow framework. Furthermore, we can circumvent fitting all models explicitly to each data set in $\mathcal{D}^{(obs)}$ and perform efficient model comparison with our evidential framework. However, as it currently stands, these frameworks seem rather disconnected and conceptualized to work on their own.

For instance, in order to perform parameter estimation for all models in $\mathcal{M}$, one has to train and store J neural density estimators. When using BayesFlow for parameter estimation, one might re-use the same (pre-trained) summary network over all models, thus effectively pooling some of the resources. However, separate inference networks would be still be needed for each of the models. Needless to say, such an approach does not scale well when J is large and needs the resources of a computational cluster to be feasible in practice.

In addition, in order to perform (prior predictive) model comparison, a disjoint training phase for an evidential network needs to be introduced. Researchers then need to ensure that simulations are shared between the parameter estimation steps and the model comparison step, otherwise a considerable portion of the simulation budget would be wasted by discarding simulations. Thus, the need for a framework which amortizes all of the above steps in a Bayesian workflow becomes immediately obvious. For such a framework to be useful in practice, it needs to enable, at least in theory, amortization over an arbitrary number of models, data sets, and observations (data set sizes).

7.2 Amortized Inference Revisited

In the following, we continue our discussion on different *levels* of amortization. In Chapter 5, we introduced three different types of amortized Bayesian inference bootstrapped by neural density estimation: case-wise, model-wise, and meta-amortized. We further assume, for the sake of our discussion, that the neural networks employed in these approaches are all capable of fully Bayesian inference (i.e., return a full posterior distribution) and use the raw simulated or observed data directly (i.e., do not rely on manual selection of summary statistics). Note, that the different types of amortization have not been explicitly distinguished in the literature on simulation-based inference, so our nomenclature is rather non-standard.

Case-wise amortized methods require a separate optimization loop for each observed data set and model. When case-wise methods incorporate a training phase (e.g., APT in a sequential regime [60]), it must be repeated for each new data set and model, since the observed data is part of the optimization criterion. The general form of the case-wise optimization criterion for obtaining optimal neural network parameters is given by:

$$\boldsymbol{\varphi}^*_{i,j} = \underset{\boldsymbol{\varphi}}{\arg\min}\, \mathbb{E}_{p(\boldsymbol{\theta}_j \,|\, \boldsymbol{x}^{(obs)}_{1:N_i}, \mathcal{M}_j)}[-\log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_j \,|\, \boldsymbol{x}_{1:N}, \mathcal{M}_j)] \tag{7.1}$$

where we have a separate set of neural network parameters $\boldsymbol{\varphi}_{i,j}$ for each data set i and model j. The case-wise approach to amortizing Bayesian inference is illustrated in Figure 7.2.

Model-wise amortized methods require a global upfront training phase before any real data are collected via simulations from each joint Bayesian model $p(\boldsymbol{\theta}_j, \boldsymbol{x}, N \,|\, \mathcal{M}_j)$. During inference, model-wise methods operate entirely in a feed-forward manner, that is, they involve no training or optimization in this phase. Thus, the upfront training effort amortizes over all observed data sets from the generative scope of model $\mathcal{M}_j$ defined by its corresponding prior $p(\boldsymbol{\theta}_j)$ and simulator $g_j(\boldsymbol{\theta}_j, \boldsymbol{\xi})$. However, in the frequent case of multiple candidate models, one still needs to perform separate optimization loops for each of the models in $\mathcal{M}$. The general form of the model-wise optimization criterion for obtaining optimal neural network parameters is given by:

$$\boldsymbol{\varphi}^*_{j} = \underset{\boldsymbol{\varphi}}{\arg\min}\, \mathbb{E}_{p(\boldsymbol{\theta}_j, \boldsymbol{x}, N \,|\, \mathcal{M}_j)}[-\log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_j \,|\, \boldsymbol{x}_{1:N}, \mathcal{M}_j)] \tag{7.2}$$

where we only have a separate set of neural network parameters for each model j. This is due to the fact that the expectation runs over the model-implied joint distribution $p(\boldsymbol{\theta}_j, \boldsymbol{x}, N \,|\, \mathcal{M}_j)$ and the observed data does not enter the optimization phase. The model-wise approach to amortizing Bayesian inference is illustrated in Figure 7.3.

Finally, a meta-amortized approach requires an even costlier upfront training phase involving simulations from multiple models (i.e., multi-model forward inference). The resulting benefit

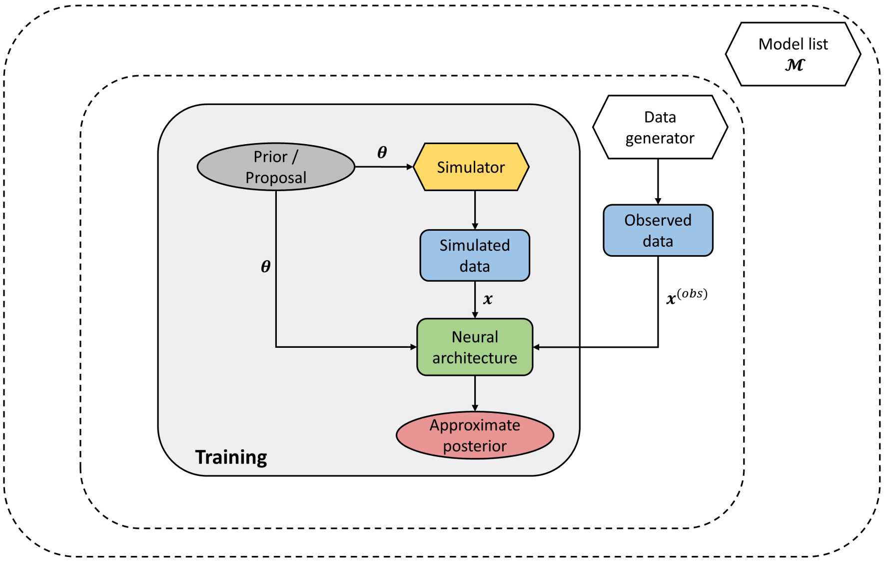

Figure 7.2: Case-wise amortized Bayesian inference with a generative neural architecture capable of fully Bayesian inference. The gray-shaded plane indicates the scope of amortization.

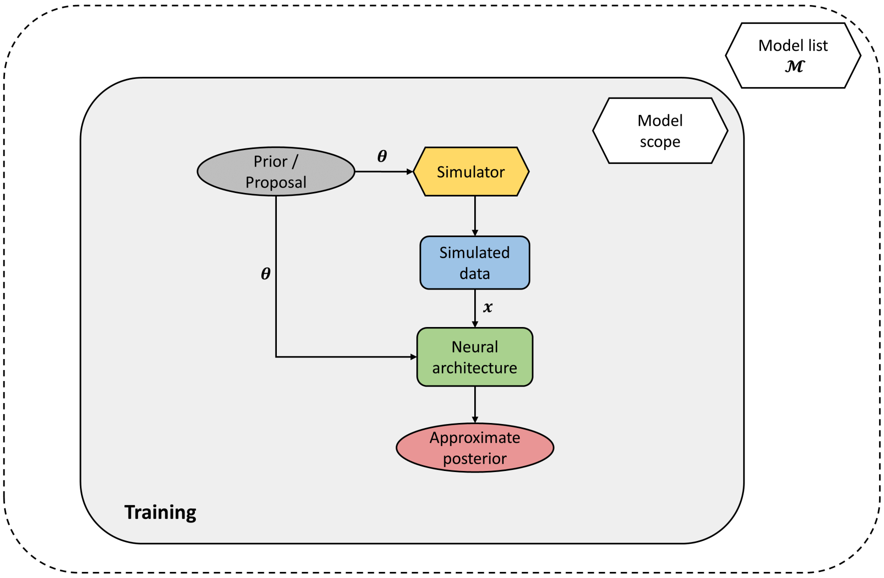

Figure 7.3: Model-wise amortized Bayesian inference with a generative neural architecture capable of fully Bayesian inference. The gray-shaded plane indicates the scope of amortization and, in contrast to case-wise approaches, includes the entire generative scope of the model.

manifests itself in the outcome that a single (composite) network is able to account for all models and all possible data sets arising from the models. Accordingly, the general form of the meta-amortized optimization criterion is given by:

$$\boldsymbol{\varphi}^* = \underset{\boldsymbol{\varphi}}{\arg\min}\, \mathbb{E}_{p(\mathcal{M},\boldsymbol{\theta},\boldsymbol{x},N)}[-\log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \mathcal{M} \,|\, \boldsymbol{x}_{1:N})] \tag{7.3}$$

where the expectation runs over the full joint model $p(\boldsymbol{\theta}, \boldsymbol{x}, N, \mathcal{M})$. The meta-amortized criterion not only opens new possibilities but also brings new challenges to which we turn next.

7.3 Learning a Multi-Model Posterior

In order to approximate the multi-model posterior $p(\boldsymbol{\theta}, \mathcal{M} \,|\, \boldsymbol{x}_{1:N})$, we seek neural network parameters $\boldsymbol{\varphi}$ which minimize our meta-amortized criterion from Equation 7.3. We can expand the latter as follows:

$$\begin{aligned}
\boldsymbol{\varphi}^* &= \underset{\boldsymbol{\varphi}}{\arg\min}\, \mathbb{E}_{p(\mathcal{M})}\left[\mathbb{E}_{p(\boldsymbol{x},N \,|\, \mathcal{M})}\left[\mathbb{E}_{p(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N},\mathcal{M})}[-\log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \mathcal{M} \,|\, \boldsymbol{x}_{1:N})]\right]\right] && (7.4)\\
&= \underset{\boldsymbol{\varphi}}{\arg\min} - \sum_{j=1}^{J} \int_{\mathcal{X}} \int_{\Theta_j} p(\mathcal{M}_j, \boldsymbol{\theta}_j, \boldsymbol{x}_{1:N})\, \log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_j, \mathcal{M}_j \,|\, \boldsymbol{x}_{1:N})\, d\boldsymbol{\theta}_j\, d\boldsymbol{x} && (7.5)
\end{aligned}$$

which we can approximate via Monte Carlo simulations from $p(\mathcal{M}, \boldsymbol{\theta}, \boldsymbol{x}_{1:N})$ (i.e., multi-model forward inference):

$$\widehat{\boldsymbol{\varphi}} = \underset{\boldsymbol{\varphi}}{\arg\min} -\frac{1}{B}\sum_{b=1}^{B} \log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_j^{(b)}, \mathcal{M}_j^{(b)} \,|\, \boldsymbol{x}_{1:N}^{(b)}) \tag{7.6}$$

Correspondingly, we can treat Equation 7.6 as a loss function and minimize it with any stochastic gradient descent method. To derive a tractable criterion, we can further expand Equation 7.6 into:

$$\widehat{\boldsymbol{\varphi}} = \underset{\boldsymbol{\varphi}}{\arg\min}\, \frac{1}{B}\sum_{b=1}^{B} -\log q_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_j^{(b)} \,|\, \boldsymbol{x}_{1:N}^{(b)}, \mathcal{M}_j^{(b)}) - \log q_{\boldsymbol{\varphi}}(\mathcal{M}_j^{(b)} \,|\, \boldsymbol{x}_{1:N}^{(b)}) \tag{7.7}$$

The above formulation has two important components: the (amortized) approximate parameter posterior $q_{\boldsymbol{\varphi}}(\boldsymbol{\theta} \,|\, \boldsymbol{x}_{1:N}, \mathcal{M})$ and the (amortized) approximate model posterior $q_{\boldsymbol{\varphi}}(\mathcal{M} \,|\, \boldsymbol{x}_{1:N})$. We will explore an architecture for meta-amortized inference consisting of three neural network components: an inference network (parameterized by $\boldsymbol{\phi}$), a summary network (parameterized by $\boldsymbol{\psi}$), and an evidence network (paramaterized by $\boldsymbol{\eta}$). Thus, $\boldsymbol{\varphi}$ represents the collection of all neural network parameters, $\boldsymbol{\varphi} = (\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\eta})$. The inference network is responsible for approximating each parameter posterior $p(\boldsymbol{\theta}_j \,|\, \boldsymbol{x}_{1:N}, \mathcal{M}_j)$. The summary network is responsible for extracting maximally informative summary vectors from raw data. Last, the evidence network is responsible for approximating the model posterior $p(\mathcal{M}, \boldsymbol{x}_{1:N})$. We now describe how to render optimization of Equation 7.7 tractable.

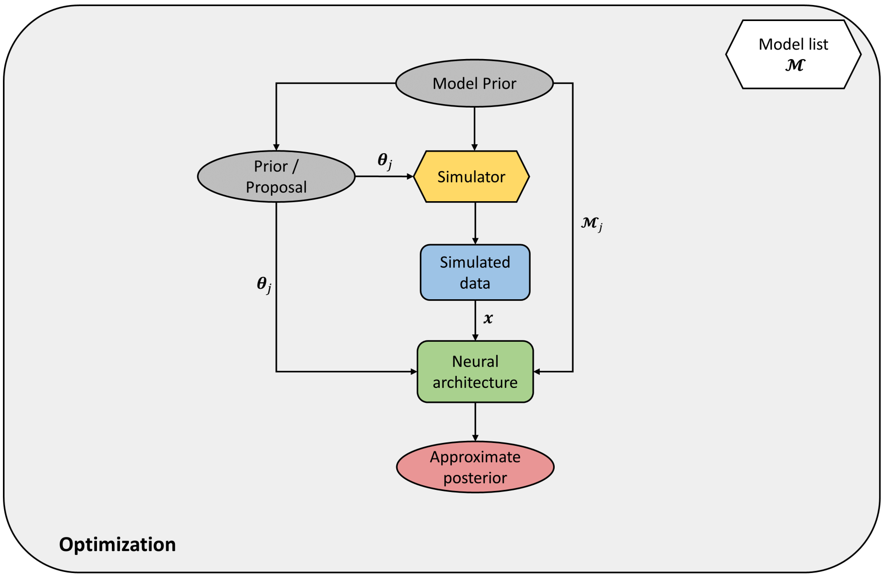

Figure 7.4: Meta-amortized Bayesian inference with a generative neural architecture capable of fully Bayesian inference. The gray-shaded plane indicates the scope of amortization. In contrast to case-wise and model-wise approaches, the latter includes the generative scopes of all models in the model list $\mathcal{M}$.

First, we can represent the (multi-model) parameter posterior via a doubly conditional INN implementing a normalizing flow between $\boldsymbol{\theta}_j$ and $\boldsymbol{z}_j$ for all j

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_j \,|\, \boldsymbol{x}_{1:N}, \mathcal{M}_j) = p(\boldsymbol{z}_j = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}_j; h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}), \boldsymbol{m}))\left|\det\left(\frac{\partial f_{\boldsymbol{\phi}}(\boldsymbol{\theta}_j; h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}), \boldsymbol{m})}{\partial \boldsymbol{\theta}_j}\right)\right| \tag{7.8}$$

where $h_{\boldsymbol{\psi}}$ is any (alogrithmically aligned) summary network with trainable parameters $\boldsymbol{\psi}$ and $\boldsymbol{m}$ is a one-hot encoded vector representation of the abstract model index $\mathcal{M}_j$. Writing $\boldsymbol{J}_{f_{\boldsymbol{\phi}}}$ as a shorthand for the Jacobian of the learnable transformation, we can derive the following loss function for a batch of B simulated model indices, parameters, and data sets:

$$\mathcal{L}_{KL}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \frac{1}{B}\sum_{b=1}^{B}\left(\frac{\left\|f_{\boldsymbol{\phi}}\left(\boldsymbol{\theta}_j^{(b)}; h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)}), \boldsymbol{m}^{(b)}\right)\right\|_2^2}{2} - \log\left|\det \boldsymbol{J}_{f_{\boldsymbol{\phi}}}^{(b)}\right|\right), \tag{7.9}$$

which is a modified version of the BayesFlow criterion (Equation 5.40) with a model index included as a further conditioning input for the cINN.

Second, we can represent the model posterior using our evidential formulation

$$q_{\boldsymbol{\eta}}(\mathcal{M} \,|\, \boldsymbol{x}_{1:N}) = \mathbb{E}_{\text{Dir}(f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}))}[\boldsymbol{\pi}], \tag{7.10}$$

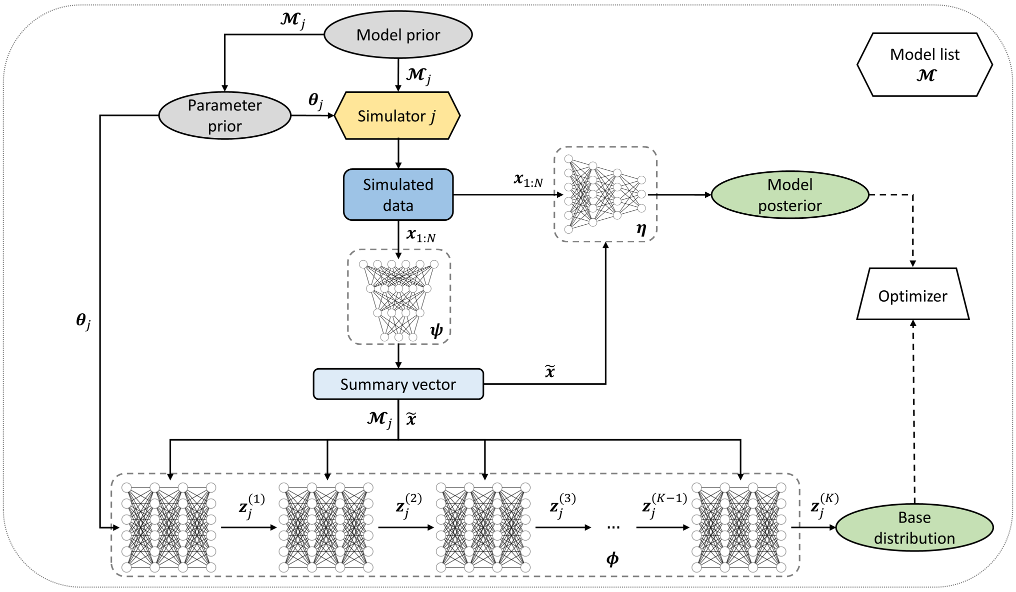

Figure 7.5: A possible framework for meta-amortized Bayesian inference connecting together an inference network ($\boldsymbol{\phi}$), a summary network ($\boldsymbol{\psi}$), and an evidence network ($\boldsymbol{\eta}$). All three networks are optimized together and trained with an arbitrary number of simulations from the joint Bayesian model $p(\mathcal{M})p(\boldsymbol{\theta} \mid \mathcal{M})p(\boldsymbol{x} \mid \boldsymbol{\theta}, \mathcal{M})$.

where $\text{Dir}(f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}))$ denotes a Dirichlet density with concentration parameters provided by an evidential network $f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N})$ with parameters $\boldsymbol{\eta}$. Note also, that the evidential network might directly use the representation provided by the summary network h as a single input, $f_{\boldsymbol{\eta}}(h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}))$, or as an additional input concatenated with the raw data, $f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}, h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}))$. Moreover, our evidential formulation allows us to re-use the concepts for learning absolute evidence introduced in Chapter 6. However, if model misspecification is not considered an issue, one could use any probabilistically calibrated classifier for $q_{\boldsymbol{\eta}}(\mathcal{M} \mid \boldsymbol{x}_{1:N})$. Accordingly, we can minimize the following (unregularized cross-entropy) loss function:

$$\mathcal{L}_{CE}(\boldsymbol{\eta}, \boldsymbol{\psi}) = \frac{1}{B} \sum_{b=1}^{B} \left[- \sum_{j=1}^{J} m_j^{(b)} \log \left(\frac{f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}^{(b)}, h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)}))_j}{\sum_{j'=1}^{J} f_{\boldsymbol{\eta}}(\boldsymbol{x}_{1:N}^{(b)}, h_{\boldsymbol{\psi}}(\boldsymbol{x}_{1:N}^{(b)}))_{j'}} \right) \right], \tag{7.11}$$

which assumes that the evidential network processes the output of the summary network, in addition to the raw simulated data.

Finally, putting the two together, our composite loss for meta-amortized inference becomes

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\eta}) = \mathcal{L}_{KL}(\boldsymbol{\phi}, \boldsymbol{\psi}) + \mathcal{L}_{CE}(\boldsymbol{\eta}, \boldsymbol{\psi}) \tag{7.12}$$

In this way, multi-model Bayesian inference is amortized through a single set of network parameters $\widehat{\boldsymbol{\varphi}} = (\widehat{\boldsymbol{\phi}}, \widehat{\boldsymbol{\psi}}, \widehat{\boldsymbol{\eta}})$ obtained via backpropagation through the entire composite architecture. An example implementation of such an architecture is illustrated in Figure 7.5. The practical utility and advantages of such a framework are yet to be demonstrated in simulated experiments and on observed data in a research context.

7.4 Use Cases and Challenges for Meta-Amortized Inference

A realization of the imaginable architecture depicted in Figure 7.5 would ensure that the computational burden of many Bayesian tasks (see red-shaded boxes in Figure 7.1) is attenuated via amortization. During inference, posterior draws from each parameter posterior $p(\boldsymbol{\theta}_j \mid \boldsymbol{x}_{1:N}^{(obs)}, \mathcal{M}_j)$ can be efficiently obtained by feeding the observed data and the desired model index through the summary and inference networks. Amortized evaluation of computational faithfulness and model adequacy as well as posterior predictive checks are all consequences of amortizing multi-model posterior inference. The model posterior $p(\mathcal{M} \mid \boldsymbol{x}_{1:N})$ can be estimated by feeding the output of the summary network from the previous step together with the observed data to the evidence network. Thus, model comparison via Bayes factors or model aggregation via Bayesian model averaging can also be performed efficiently upon convergence.

However, the actual implementation of the neural architecture depicted in Figure 7.5 is not as straightforward as its conceptualization. When it comes to the architecture of the network responsible for parameter inference, it appears necessary to distinguish between three main scenarios encountered in multi-model inference.

The first corresponds to *prior sensitivity analysis* in which the consequences of different prior configurations for subsequent Bayesian updating are systematically investigated. In this case, the simulator for each $\mathcal{M}_j$ is the same and only the corresponding $p(\boldsymbol{\theta} \mid \mathcal{M}_j)$ differ in their distributional form. Prior sensitivity analysis is perhaps also the easiest case to tackle, since it requires no essential structural changes to the cINN, which is augmented to accept the one-hot encoded model index as an additional conditioning variable.

The second corresponds to a setting in which, once again, the simulator remains conceptually and functionally the same, but some components of $\boldsymbol{\theta}$ are treated as fixed and some as varying. In this case, some parts of the latent space $\boldsymbol{z}$ are shared among all models, whereas others are missing whenever a subset of the model parameters is treated as fixed. In other words, a complete parameter vector is reduced from $\boldsymbol{\theta} \in \mathbb{R}^D$ to $\boldsymbol{\theta}_j \in \mathbb{R}^{D_j}$ with $D_j \leq D$ for each model j. Two potential approaches for performing meta-amortized Bayesian inference in such a setting appear viable. In the first, each reduced parameter space Θ_j is augmented to Θ'_j, with additional dummy parameters following a simple distribution (e.g., Gaussian), such that each augmented $\boldsymbol{\theta}'_j$ has the same dimensionality. A disadvantage of this approach is, that the inference network needs to learn an identity transformation between the dummy parameters and the corresponding latent variables, which seems inelegant. In the second approach, missing parameters are encoded with zeros and the loss function is masked, such that each $\boldsymbol{z}_j$ is optimized only from the remaining parameters. A disadvantage of this approach is that it is less straightforward to implement and might lead to training instabilities.

The third scenario corresponds to the task of comparing essentially different generative mechanisms assumed to account for the same data. In this case, each $\mathcal{M}_j$ is implemented as a different simulator g_j and the number of parameters might or might not differ between the simulators.

Such a scenario would require learning disjoint latent spaces z_j for each simulator. It might even necessitate separate inference networks (with a shared summary network) for each model in $\mathcal{M}$ or a completely different neural architecture altogether.

As could be gathered from this short exposition, many problems lurking on the path towards a universal framework for meta-amortized simulation-based inference remain unsolved and many unsuspected challenges are yet to reveal themselves. However, since meta-amortized inference appears to be a desirable goal for many scientific domains, future research should explore various promising avenues for actually attaining it.

8 Applications

This chapter briefly reviews four concrete applications of our ideas for amortized Bayesian inference to real-world modeling problems. The details of these modeling scenarios have already been described in the corresponding papers [31, 94, 135, 172], so the purpose of this chapter is merely to convey the gist of each particular application and describe how our developed frameworks contributed to solving the problem of inference.

8.1 A Bayesian Brain Model of Adaptive Behavior

In this work[1], we proposed and validated a new computational Bayesian model accounting for individual performance in the Wisconsin Card Sorting Test (WCST), an established clinical tool for measuring set-shifting and deficient inhibitory processes on the basis of environmental feedback [6, 67].

Performance in WCST is usually measured via a rough summary metric such as the number of correct/incorrect responses or pre-defined psychological scoring criteria (see for instance [67]). These metrics form the basis for inferring the underlying cognitive processes recruited by the task. However, a major shortcoming of this approach is that it merely assumes the cognitive processes to be inferred without specifying an explicit process model. Moreover, summary measures do not utilize the full information present in the data, such as trial-by-trial fluctuations or relevant agent-environment interactions. For this reason, crude scoring measures are often insufficient to disentangle the dynamics of the relevant cognitive components.

To address this shortcoming, we formalized the interaction between the task's structure, the received feedback, and the participant's behavior by building a model of the underlying information processing mechanisms used to infer the hidden rules of the task environment. Furthermore, we embedded the new model within the mathematical framework of the Bayesian Brain Theory (BBT), according to which beliefs about hidden environmental states are dynamically updated following the logic of Bayesian inference [48, 91].

The simple controlled setting (environment) realized by the WCST consists of a target and a set of stimulus cards with geometric figures which vary according to three perceptual features: color (red, green, blue, yellow), shape (triangle, star, cross, circle), and number of objects (1, 2, 3, 4). Participants in the test have to infer the correct classification principle by trial and error using the examiner's or computer's feedback. The feedback carries a positive or a negative signal informing the participant whether her choice of action was appropriate or not. Moreover, modeling adaptive behavior in the WCST from a Bayesian perspective is straightforward, since observable actions

[1] M. D'Alessandro, S. T. Radev, A. Voss, and L. Lombardi. "A Bayesian brain model of adaptive behavior: an application to the Wisconsin Card Sorting Task". *PeerJ* 8, 2020, e10316

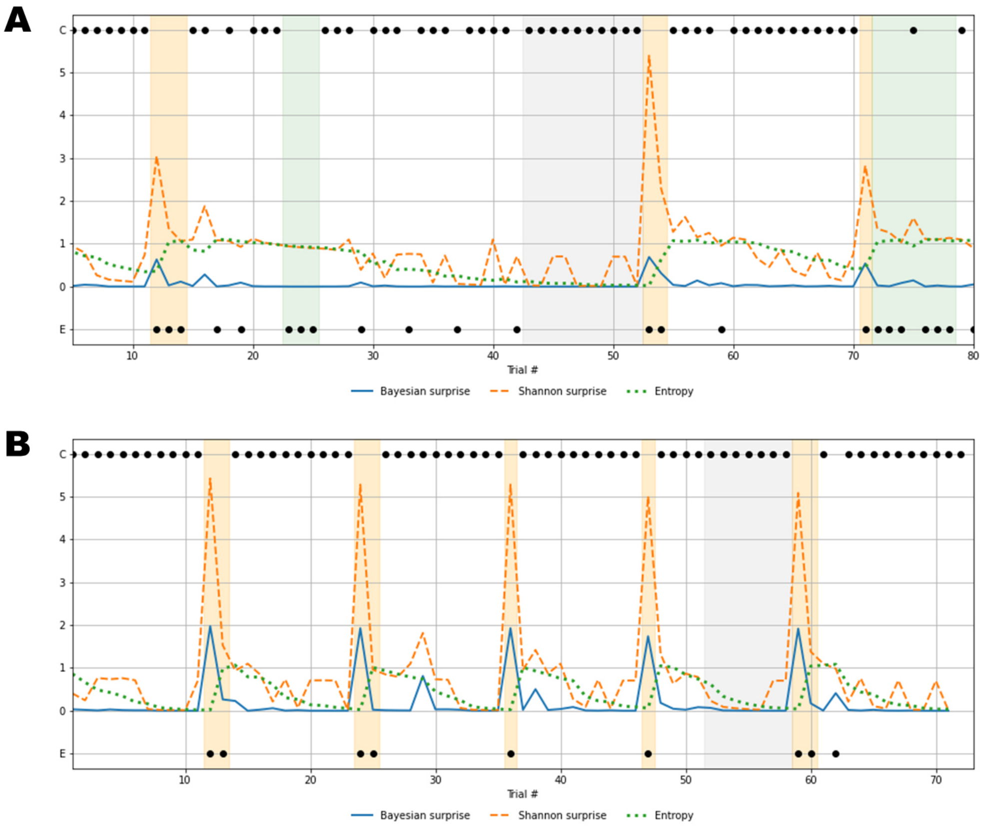

Figure 8.1: Estimated information processing dynamics of two exemplary individuals [31]. (**A**) Trial-by-trial information-theoretic measures of an individual with SD characterized by very low flexibility and very high information loss; (**B**) Trial-by-trial information-theoretic measures of a healthy control individual characterized by relatively high flexibility and low information loss. Labels C and E on the y-axis indicate correct and error responses.

emerge from the interaction between the internal probabilistic model of the agent and a set of discrete environmental states.

The main contributions of our modeling work were thus threefold. First, we developed a two-level model of adaptive agent-environment interaction, consisting of a cognitive and an information-theoretic component. The cognitive component decomposes performance in the WCST into two interpretable parameters: *flexibility* and *information loss*. The information-theoretic component transforms the parameters into dynamic measures of belief updating, surprise, and internal model uncertainty. Second, we performed extensive simulation studies for ensuring reasonable computational faithfulness and model sensitivity. Third, we applied the model to a sample of individuals with substance dependence (SD) and a sample of healthy controls to account for (mal)adaptive task performance in a principled way. For the latter two tasks, we had to resort to amortized inference with BayesFlow, since the likelihood function of our custom dynamic model is unknown. Overcoming this intractability with BayesFlow (using a recurrent summary network), we could perform both simulation-based calibration, parameter recovery, and inference on real data in a matter of seconds, once training had converged. The entire training phase took approximate 12 hours wall-clock time on a laptop with a graphics card.

Our initial application showed promising results in explaining adaptive behavior in the WCST. Figure 8.1 depicts the model-derived information processing dynamics of an individual with SD (upper panel) and a healthy control (lower panel). Indeed, patterns of belief updating (*Bayesian surprise*), surprisal (*Shannon surprise*), and model uncertainty (*Entropy*) are very different for the two individuals, highlighting the ability of the model to discriminate between sub-optimal and nearly-optimal performance via multiple sources of information (see [31] for a detailed interpretation).

8.2 Jumping to Conclusion? A Lévy-Flight Model of Decision Making

In this work[2], we formally tested whether a Lévy flight model, assuming an α-stable noise distribution with a free parameter α can provide a more accurate description of performance in simple binary decision making tasks than a classical diffusion model, assuming a Gaussian noise distribution.

Distributions with *fat tails*, such as the Cauchy distribution or the Lévy distribution, are characterized by an increased probability for extreme events, compared to a Gaussian distribution [95]. Moreover, fat-tailed models incorporating so-called Lévy flights have been applied in a variety of research contexts. For instance, such models have proven useful to account for animal foraging behavior. The Lévy flight foraging hypothesis states that in certain natural environments, (truncated) Lévy flights optimize random searches. Accordingly, the hypothesis implies that biological organisms have evolved to exploit occasional large divergences in their wandering movements during foraging, which are best accounted for by Lévy flights [168].

In our study on human decision making [172], we compared the relative fit of four evidence accumulation models applied to a color discrimination and a lexical decision task. In the color

[2] E. M. Wieschen, A. Voss, and S. Radev. "Jumping to conclusion? a lévy flight model of decision making". *TQMP* 16:2, 2020, pp. 120–132

discrimination task, participants were tasked to indicate whether there were more orange or more blue pixels in a set of specifically designed stimuli. In the lexical decision task, participants were tasked to indicate whether a presented string of letters was an existing German word or a meaningless sequence. The candidate models included: $\mathcal{M}_1$ - a parsimonious version of the diffusion model with Gaussian noise; $\mathcal{M}_2$ - a model with an α-stable distribution for the noise of evidence accumulation; $\mathcal{M}_3$ a full version of the diffusion model with inter-trial variability for drift, starting point and non-decision time; and $\mathcal{M}_4$ a model with alpha as a free parameter and all previous inter-trial variability parameters. Note, that the all-time favorite Gaussian distribution is a special case of the stable family with a stability value of $\alpha = 2.0$. Lower values of α imply a higher probability of extreme events and thus occasional large *jumps* in the evidence accumulation process.

The inclusion of stable noise in the accumulation process renders numerical evaluation of the likelihood intractable. Thus, we train a separate BayesFlow network for each of the candidate models, in order to ensure that all models are comparably estimated within the same Bayesian framework. The training phase for each model took less than 12 hours on a laptop with GPU acceleration. In contrast, subsequent parameter recovery, calibration checks, and inference on the experimental data took a couple of seconds.

Our initial results suggest, in accordance with previous results [169], that the simple Lévy model ($\mathcal{M}_1$) yields a superior fit than the simple diffusion model ($\mathcal{M}_2$) for both experimental tasks. In addition, the complex Lévy model ($\mathcal{M}_4$) had a superior fit than the complex diffusion model ($\mathcal{M}_3$). Finally, each of the complex models (including inter-trial variability parameters) exhibited a superior fit than each of the simpler models. We speculate, that such a result might be explained by a particular property of the experiments: The longer duration possibly induces larger fluctuations in performance which is best captured by the inter-trial variability parameters (see [172] for a more in-depth interpretation).

8.3 Insights from Bayesian Modeling in a One Million Sample

In this work, we applied BayesFlow to elucidate cross-sectional age differences in cognitive parameters as indexed by the main diffusion model parameters. A large number of studies from the last decades have reported that processing speed, typically measured as mean response time (RT) in simple cognitive tasks, significantly slows down in old age and starts to decline in young and middle adulthood [78, 143]. We challenged this notion by carrying out a comprehensive model-based analysis on a massive, publicly available data set ($M > 1\,000\,000$) collected during the course of Project Implicit [173]. Notably, this sample was multiple orders of magnitude larger than the data sets used in all previous diffusion model studies combined. Accordingly, our approach was able to provide unique and robust findings on age-related patterns regarding processing speed, decision caution, and non-decision components of RTs.

Furthermore, applying Bayesian diffusion modeling to a sample of such magnitude appears to be a task insurmountable by standard Bayesian (or non-Bayesian, for that matter) methods. Thus, we resorted to BayesFlow with a deep invariant summary network for efficient amortized inference. In this way, fully Bayesian inference with BayesFlow (training and inference phase) on the entire sample took less than two days on a standard computer. We also estimated, that

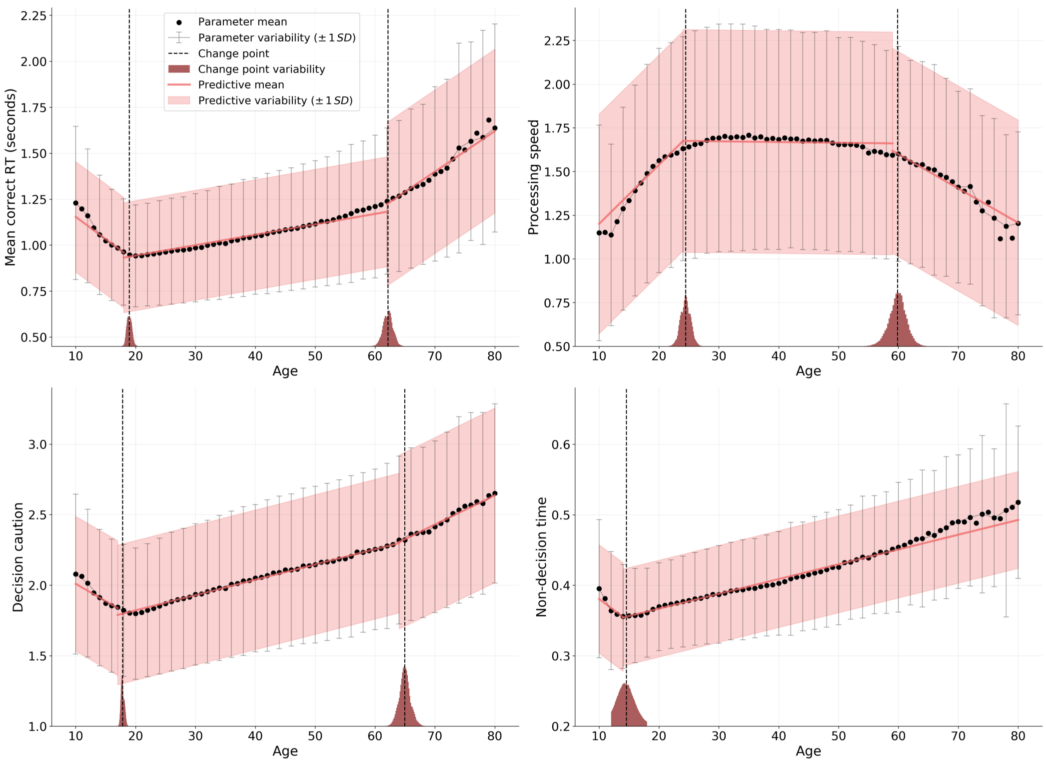

Figure 8.2: Mean correct response times (RTs) and main diffusion model parameters as a function of age. Black points indicate parameter means computed separately for each age (in years). Bars indicate standard deviations (only shown for every second year). Red lines denote the Bayesian piece-wise ridge regression model's mean predictions, which describe the observed means fairly well. The shaded red region denotes the uncertainty (standard deviation) of the piece-wise model's predictions. The dashed lines indicate the mean change points estimated from the per-age-group averaged data, with the full posterior distributions (scaled for readability) of the change points shown at the bottom of each plot. Both the data- and model-implied standard deviations highlight the great variability within each year of age. Nevertheless, the year-specific means suggest a clear and consistent pattern for mean correct RT and each parameter. The figure depicts drift rates and boundary separations for the incongruent condition and non-decision times obtained from correct responses.

applying MCMC for the same number of posterior samples per data set (participant) would have taken more years than are currently available to any human scientist.

Our analysis pipeline followed the steps advocated by a principled Bayesian workflow [144] which ensure a transparent presentation of computational faithfulness and model adequacy. Following parameter estimation, we applied a Bayesian change point regression of each cognitive estimate (and also mean RT) on age. The results from one of the two experimental conditions are depicted in Figure 8.2 (parameter estimates in the other condition do not exhibit qualitatively different age-related patterns).

Importantly, our results suggest a clear non-linear association between drift rate (as an index of processing speed) and age, which was strikingly different than the one implied by mean RTs and far more informative than the age differences found in previous diffusion model studies (cf. Figure 8.2). Thus, our model-based analysis suggests a picture of age differences in cognitive parameters yielding a radically different implication than the one based on model-free analysis of raw RT data.

8.4 OutbreakFlow: Model-Based Bayesian Inference of Disease Outbreak Dynamics

In this work[3], we applied a version of BayesFlow for dynamic (stateful) models to infer important disease characteristics and transmission dynamics of the Covid-19 pandemic in Germany. Inference of hidden disease-related parameters is of utmost importance in the case of new outbreaks in order to forecast their progression and guide effective public health measurements. Accordingly, mathematical models that provide a reliable representation of the processes driving the dynamics of an epidemic are an essential tool for this task (see for example [81]).

In order to account for the specific nature of the initial Covid-19 outbreak in Germany, we specified a custom compartmental model consisting of three sub-models: a disease model, an observation model, and an intervention model.

The *disease model* is represented by a system of non-linear ordinary differential equations (ODEs) comprising six compartments: susceptible (S), exposed (E - infected individuals who do not show symptoms and are not yet infectious), infected (I - symptomatic cases that are infectious), carrier (C - infectious individuals who recover without being detected), recovered (R), and dead (D).

The *intervention model* represents the time-varying transmission rate $\lambda(t)$. Following [34], we defined three change points encoding an assumed transmission rate reduction in response to public health measures (e.g., lockdown, social distancing) imposed by the German authorities. Each change point is represented by a piece-wise linear function with three degrees of freedom: the strength of interventions and the time interval (start and end point) for the effect to fully manifest itself.

The *observation model* represents the deviations between officially reported case counts and their true values. It comprises three sources of systematic and unsystematic errors: the reporting

[3] S. T. Radev, F. Graw, S. Chen, N. Mutters, V. Eichel, T. Bärnighausen, and U. Köthe. "Model-based Bayesian inference of disease outbreak with invertible neural networks". *arXiv preprint arXiv:2010.00300*, 2020

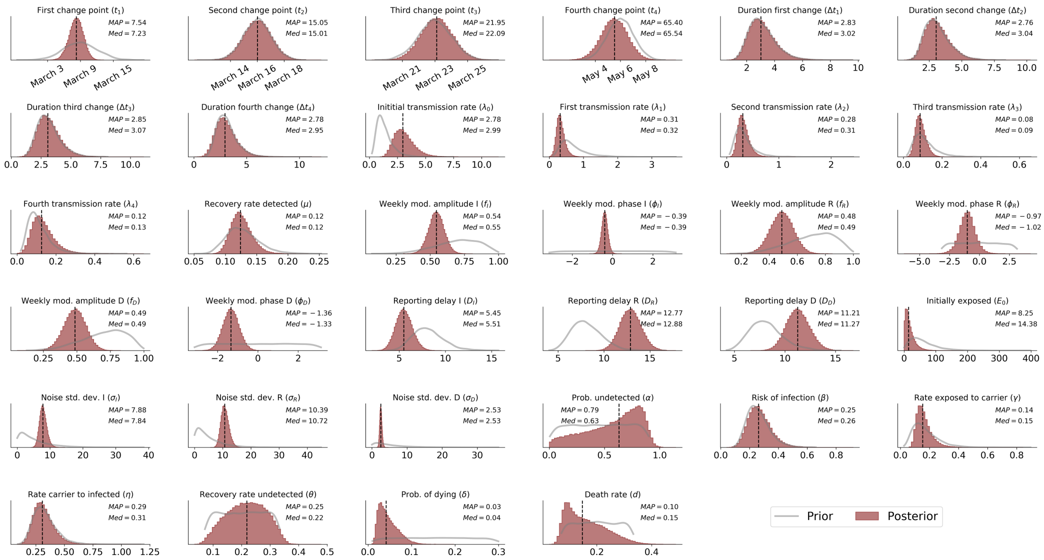

Figure 8.3: Marginal posteriors of all 34 model parameters inferred from data from entire Germany alongside median and MAP summary statistics. Gray lines depict prior distributions for comparison with the posteriors. Vertical dashed lines indicate posterior medians.

delay, the weekly modulation (since testing and reporting activities are considerably reduced on weekends), and a symmetric *t*-distributed noise term describing random fluctuations.

Due to the complexity of this composite model and the need to apply the same model repeatedly to different federal states, we resort to efficient simulation-based inference with BayesFlow (see [135] for details regarding network architectures). Furthermore, amortized Bayesian inference appears especially advantageous in epidemiological contexts, where the same model is estimated in multiple populations (countries, cultures) or at different scales (states, regions). Indeed, in the current application, we were able to demonstrate efficient amortized inference and excellent predictive performance with a single architecture applied simultaneously to epidemiological data from Germany as a whole and all sixteen German federal states.

Marginal parameter posteriors for Germany as a whole are depicted in Figure 8.3. Posterior predictions and forecasts for new infections, recoveries, and deaths are further depicted in Figure 8.4. We observe that median predictions of our model follow very closely the reported cumulative number of cases across all federal states. Furthermore, the officially reported cases are very well represented by the uncertainty bounds derived from the parameter posteriors, with prediction uncertainty growing as we move towards the future (cf. predictions after the dotted vertical lines in Figure 8.4). When interpreting these results, the reader should be aware that mechanistic models like ours only describe the average behavior of entire compartments, in contrast to agent-based models. Accordingly, the given CIs quantify our uncertainty about the inferred parameter averages and *cannot* be interpreted as a measure of the variability between individual cases.

Our estimates suggest that a considerable number of individuals (a fraction of 60-80 % of cases) might have gone undetected through the course of the Covid-19 outbreak in Germany, confirming results from previous studies in other countries [32, 115, 128]. However, our posteriors also suggest

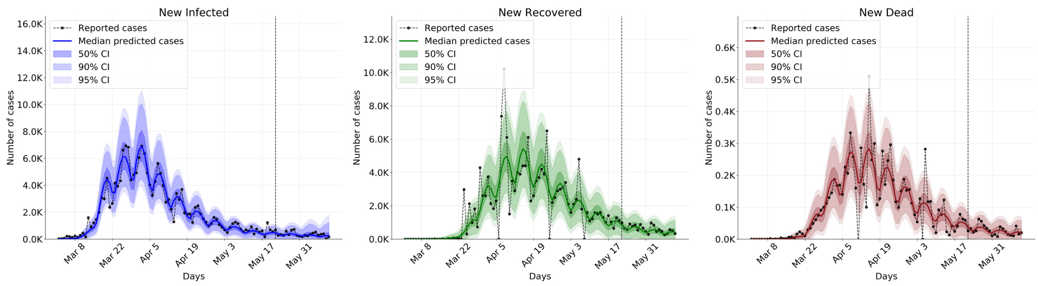

Figure 8.4: Model-based predictions and forecasts of new cases obtained by inferring model parameters from epidemiological data available for reported infected, assumed recovered and deaths by Covid-19 from entire Germany. Cases to the left of the vertical dashed line were used for posterior checking (model training) and cases to the right for posterior forecasts (predictions) on upcoming data.

that there is non-negligible uncertainty surrounding this estimate when derived in a purely model-based manner. Moreover, different summary statistics (e.g., means, medians, MAPs) derived from non-symmetric posteriors offer slightly different conclusions. The latter observation highlights the need to consider the full posteriors and corresponding credibility intervals when aiming to draw substantive conclusions and possible forecasts for the progression of the epidemic or the effect of specific public health interventions.

9 Outlook

The current thesis presented frameworks and ideas for scaling up many steps of a complete Bayesian workflow with a focus on cognitive modeling. A cornerstone notion of this thesis was to employ generative neural networks for amortized Bayesian parameter inference, model comparison and validation when working with intractable simulators whose behavior as a whole is too complex to be described analytically. We presented various frameworks for tackling different types of models (e.g., stateless vs. stateful) and Bayesian tasks (e.g., parameter estimation, model comparison, model calibration). A common theme was splitting Bayesian analysis into two conceptual phases: i) a training phase, in which the networks gradually become domain experts in solving the Bayesian tasks they are optimized for, and ii) a downstream inference phase, in which the networks are efficiently applied to extract information from real-world observations about quantities of interest (e.g., model parameters or model plausibility). Further, we explored potential developments towards meta-amortized Bayesian inference and discussed related challenges standing in the way of such a generalized framework. Finally, we presented some applications of BayesFlow to a number of complex estimation problems. In the following, we briefly go through some further topics left for future research beyond those mentioned in previous chapters.

Dynamic parameters In this thesis, we have focused exclusively on models defined by a fixed number of parameters $\boldsymbol{\theta}$. However, some dynamic models might incorporate parameters which are a function of time (or another variable), implying that a different set of parameters $\boldsymbol{\theta}(t)$ is available at each time point t. For instance, realistic spatio-temporal models of disease outbreaks strive to capture relevant disease characteristics $\boldsymbol{\theta}(t, s)$ as a function of time t and space s [24], introducing yet another dimension to the parameter space. While this setting poses no inherent problems for our evidential framework for model comparison, it presents a challenge for parameter estimation with BayesFlow.

One approach to amortized inference with such models would be to re-parameterize the problem so that we can estimate a fixed-size set of parameters $\boldsymbol{\omega}$ of which the time-varying parameter vector $\boldsymbol{\theta}(\boldsymbol{\omega}, t)$ is a deterministic function. This approach is then easily amenable to amortized inference with BayesFlow, and is the one we followed in [31] for recovering trial-by-trial information processing dynamics or in [135] for recovering the time-varying transmission rate of Covid-19 in Germany. However, not all models naturally admit such a re-parameterization, so estimating the original $\boldsymbol{\theta}(t)$ might be inescapable in these cases. Thus, another viable approach is to modify the summary and inference networks in order to capture the dynamic structure of the problem. For instance, the inference network can be easily implemented as a generative recurrent network which outputs a latent embedding $\boldsymbol{z}(t)$ for each $\boldsymbol{\theta}(t)$. In this way, it can interact with a recurrent summary network in the form of a probabilistic sequence-to-sequence architecture [124, 151] Transformer networks utilizing neural attention mechanisms [83, 164] appear as another promising option for tackling dynamic models without recurrent networks.

Hyperparameter optimization A common aspect of our frameworks for amortized Bayesian inference is the potentially large number of hyperparameter settings that might require fine-tuning by the user for optimal performance on a given Bayesian task. For instance, the most important hyperparameters for BayesFlow are: optimizer settings (e.g., learning rate, adaptive weights, decay); summary network design (e.g., type of modular architecture, number of layers and neurons in each module); inference network design (e.g., type of flow, number of layers, structure of each internal network); training schedule (e.g., online vs. offline learning). This makes general experience with neural networks highly advantageous when working with our frameworks, to say the least.

So far, in our simulated experiments and applications, we could empirically ascertain that some hyperparameters are more important than others. For instance, optimizer settings appear to be vital for stable training. When working with the Adam optimizer [84], smaller learning rates (i.e., $\alpha < 0.001$) and the inclusion of learning rate decay generally lead to more stable convergence than larger learning rates and no decay. On the other hand, using larger networks consisting of 3 to 10 coupling layers does not seem to hurt performance or destabilize training, even if the simulator to be inverted is relatively simple [133]. Based on our results, we expect that a single architecture should be able to perform well on similar simulators from a given domain (e.g., one architecture for all EAMs [172] or one architecture for all compartmental models [135]). Future research should investigate the impact of modern hyperparameter optimization methods, such as Bayesian optimization [41]. Moreover, Bayesian optimization, or other black-box optimization methods or search algorithms can easily be integrated into our frameworks (e.g., in a pre-training phase with utilizing a small number of simulations).

Optimal experimental design Optimal experimental design (OED) in a Bayesian context is a mathematical framework for making efficient use of limited experimental resources when performing Bayesian modeling [21, 46, 110]. A majority of OED approaches revolve around the notion of expected information gain (EIG), which quantifies the expected reduction in entropy (uncertainty) when replacing the prior with the posterior under the marginal distribution over experimental observations. For instance, in static design optimization (DO), a researcher sets up a simulation involving different models in different experimental contexts and picks the configuration which yields the highest EIG. No further optimization happens during the actual experiment. Differently, in adaptive design optimization (ADO), the EIG is computed (estimated) on each trial and subsequent trials are chosen in order to maximize the discriminability between candidate models or the sharpness of the posterior in a single-model setting.

Unfortunately, obtaining accurate approximations of the EIG even for simpler models is computationally demanding and nearly infeasible with non-amortized methods for Bayesian updating. Variational OED offers a promising approach for amortizing different aspects of OED [46]. Moreover, utilizing neural networks for efficiently maximizing a lower bound on the EIG (i.e., variational autoencoders, VAEs, [87]), variational methods are able to yield considerable efficiency gains. However, vanilla VAEs maximizing a lower bound on the actual (intractable) criterion, the so called ELBO criterion, suffer from some rather consequential problems, as aptly demonstrated by [178]. Further, in contrast to normalizing flows realized via invertible neural networks, vanilla variational methods offer no theoretical guarantee for learning the correct target posterior when employed for the task of Bayesian updating.

Our proposed frameworks for amortized Bayesian parameter estimation and model comparison appear suitable for amortizing OED. For instance, BayesFlow can be adapted for estimating the EIG in either static DO or ADO. Further, our evidential framework can be employed in multi-model DO or ADO contexts. Such an integration is possible, since our networks can be augmented to process arbitrary contextual information. Moreover, we can emulate Bayesian updating by training the networks with a variable number of observations N (using algorithmically aligned networks), such that amortization over increasing N is enabled during inference. Thus, future research should investigate the utility of our frameworks for amortizing OED and compare them to variational approaches.

Model-aware learning Finally, our frameworks currently operate in a model-agnostic manner, that is, the neural estimators treat the simulator purely as a black-box data generator. For researchers, on the other hand, the neural networks (in addition to reality itself) are uninterpretable black-boxes while the simulator serves, at least in theory, as a human-interpretable, white-box computational model. Thus, it is possible that the networks can profit from some prior "knowledge" of the model's structure (e.g., in the form of gradients or other information) or generative scope in order to learn even more efficiently the resulting probabilistic mappings. The networks themselves could guide the model to produce more realistic artificial observations, for instance, by restricting the generative scope of the simulator.

Sequential neural posterior estimation (SNPE, [60]) methods offer a neat way to gradually transform the prior $p(\boldsymbol{\theta})$ into a sharper proposal $\widehat{p}(\boldsymbol{\theta})$ (eventually becoming the target posterior) through iterative refinement. In this way, the generative scope of the joint Bayesian model $p(\boldsymbol{\theta}, \boldsymbol{x})$ also becomes narrower, concentrating around the actually observed data $\boldsymbol{x}^{(obs)}$. Thus, SNPE methods implement one promising form of network-simulator interaction. One consequence of this approach, however, is that such an interaction necessarily reduces amortization to a case-wise level, since the neural density estimator needs to be re-trained for each observed data set. Needless to say, such repetitions become increasingly computationally taxing in data-rich applications. Future research should therefore explore other forms of *model-aware* learning, which enable model-wise or even meta-amortized Bayesian inference.

10 Conclusion

Reality is noisy and messy, and there is no grand simulator of things in sight. What is more, our models can only restrict their scope to increasingly smaller empirical nooks, solving, at best, tiny fractions of an infinite jigsaw puzzle. As we, researchers in need of cognition, go through the process of building new models and discarding old ones, we require the right tools to foster our epistemic achievements. At the time of writing, deep learning continues to enjoy a vibrant hype, refurbishing the methodological equipment of many quantitative sciences. The behavioral sciences, despite being more resistant to change than fellow disciplines, are also enjoying their fair share of the rapidly expanding trend. When it comes to model-based inference, deep learning innovations are currently transforming the way models are fit to data and employed for drawing substantive conclusions or deriving reliable forecasts. Moreover, uncertainty (an ancient concept) and its quantification are becoming more and more important in deep learning theory and practice. Bayesian methods, deeply rooted in probability theory, are currently viewed by many researchers as the gold-standard for uncertainty-aware inference, but other approaches or generalizations might push through in the not-so-distant future. In a way, this thesis ventured into a discourse between deep learning and scientific modeling with a focus on cognitive science and mathematical psychology. It brought together ideas for dramatically accelerating Bayesian inference by using non-Bayesian neural networks designed to deal with the data types encountered by researchers working in various areas of knowledge. The general idea of using black-box estimators to learn white-box scientific models from computer simulations is certainly not new, but is still largely underutilized in the behavioral and cognitive sciences. Most importantly, future research should further foster the discourse between deep learning (or *artificial intelligence*, in marketing jargon) and the behavioral sciences due to the potential upside of such a creative interaction. Indeed, human researchers and decision-makers can definitely learn something from deep learning agents surpassing human performance in various real-world tasks. On the other hand, neural networks can also learn something from studying the structure of human behavior and cognition. Luckily, the global connectedness of the modern world makes such mutual learning a rather effortless endeavor. Finally, models of human behavior and cognition need to come to terms with the buzzword of the century: *complexity*. We have begun to realize, that simple models can only do so much in aiding our understanding of emergent phenomena. On the other hand, we have grown suspicious of opaque, overparameterized neural networks capable of solving overly specific tasks. It is thus very well possible, that future models of cognition and behavior become more and more uninterpretable (i.e., more black-boxy) in the pursuit of complexity. On the other hand, future neural network might become more and more interpretable (i.e., less black-boxy) due to our need for cognition. As always, predictions are hard, especially about the future.

Bibliography

1. L. Abbott and T. B. Kepler. "Model neurons: from hodgkin-huxley to hopfield". In: *Statistical mechanics of neural networks*. Springer, 1990, pp. 5–18.
2. J. R. Anderson. *How can the human mind occur in the physical universe?* Vol. 3. Oxford University Press, 2009.
3. L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. "Analyzing inverse problems with invertible neural networks". In: *Intl. Conf. on Learning Representations*. 2019.
4. L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe. "Guided image generation with conditional invertible neural networks". *arXiv preprint arXiv:1907.02392*, 2019.
5. M. Belkin, D. Hsu, S. Ma, and S. Mandal. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". *Proceedings of the National Academy of Sciences* 116:32, 2019, pp. 15849–15854.
6. E. A. Berg. "A simple objective technique for measuring flexibility in thinking". *The Journal of general psychology* 39:1, 1948, pp. 15–22.
7. J. O. Berger. "Robust Bayesian analysis: sensitivity to the prior". *Journal of statistical planning and inference* 25:3, 1990, pp. 303–328.
8. M. Betancourt. "A conceptual introduction to Hamiltonian Monte Carlo". *arXiv preprint arXiv:1701.02434*, 2017.
9. A. Biedermann and F. Taroni. "Bayesian networks for evaluating forensic DNA profiling evidence: a review and guide to literature". *Forensic Science International: Genetics* 6:2, 2012, pp. 147–157.
10. S. Bieringer, A. Butter, T. Heimel, S. Höche, U. Köthe, T. Plehn, and S. T. Radev. "Measuring QCD splittings with invertible networks". *arXiv preprint arXiv:2012.09873*, 2020.
11. D. Bigoni, O. Zahm, A. Spantini, and Y. Marzouk. "Greedy inference with layers of lazy maps". *arXiv:1906.00031*, 2019.
12. D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. "Variational inference: A review for statisticians". *Journal of the American statistical Association* 112:518, 2017, pp. 859–877.
13. B. Bloem-Reddy and Y. W. Teh. "Probabilistic symmetries and invariant neural networks". *Journal of Machine Learning Research* 21:90, 2020, pp. 1–61.
14. M. G. Blum and O. François. "Non-linear regression models for Approximate Bayesian Computation". *Statistics and computing* 20:1, 2010, pp. 63–73.

15. O. Breidbach. "Weltordnungen und Körperwelten. Das Tableau des Gewussten und seine Repräsentation bei Robert Fludd". *Schramm, H./Schwarte, L./Lazardzig, J.(Hg.): Instrumente in Kunst und Wissenschaft. Zur Architektonik kultureller Grenzen im* 17, 2006, pp. 41–65.
16. A. N. Burkitt. "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input". *Biological cybernetics* 95:1, 2006, pp. 1–19.
17. P.-C. Bürkner, J. Gabry, and A. Vehtari. "Approximate leave-future-out cross-validation for Bayesian time series models". *Journal of Statistical Computation and Simulation* 90:14, 2020, pp. 2499–2523.
18. G. Buzsaki. *Rhythms of the Brain*. Oxford University Press, 2006.
19. B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. "Stan: A probabilistic programming language". *Journal of statistical software* 76:1, 2017.
20. R. Caruana. "Multitask learning". *Machine learning* 28:1, 1997, pp. 41–75.
21. D. R. Cavagnaro, J. I. Myung, M. A. Pitt, and J. V. Kujala. "Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science". *Neural computation* 22:4, 2010, pp. 887–905.
22. P. C. Cheeseman. "In defense of probability." In: *IJCAI*. Vol. 2. 1985, pp. 1002–1009.
23. L. Chen, G. Zhang, and E. Zhou. "Fast greedy map inference for determinantal point process to improve recommendation diversity". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5622–5633.
24. Y. Chen, Q. Li, H. Karimian, X. Chen, and X. Li. "Spatio-temporal distribution characteristics and influencing factors of COVID-19 in China". *Scientific Reports* 11:1, 2021, pp. 1–12.
25. K. Choi, M. Wu, N. Goodman, and S. Ermon. "Meta-amortized variational inference and learning". In: *International Conference on Learning Representations*. 2019.
26. J. M. J. da Costa, H. R. B. Orlande, and W. B. da Silva. "Model selection and parameter estimation in tumor growth models using approximate Bayesian computation-ABC". *Computational and Applied Mathematics* 37:3, 2018, pp. 2795–2815.
27. K. Cranmer, J. Brehmer, and G. Louppe. "The frontier of simulation-based inference". *Proceedings of the National Academy of Sciences*, 2020.
28. F. Crick and J. Clark. "The astonishing hypothesis". *Journal of Consciousness Studies* 1:1, 1994, pp. 10–16.
29. K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François. "Approximate Bayesian computation (ABC) in practice". *Trends in Ecology & Evolution* 25:7, 2010, pp. 410–418.
30. M. D'Alessandro, G. Gallitto, A. Greco, and L. Lombardi. "A Joint modelling approach to analyze risky decisions by means of diffusion tensor imaging and behavioural data". *Brain sciences* 10:3, 2020, p. 138.

31. M. D'Alessandro, S. T. Radev, A. Voss, and L. Lombardi. "A Bayesian brain model of adaptive behavior: an application to the Wisconsin Card Sorting Task". *PeerJ* 8, 2020, e10316.
32. M. Day. *Covid-19: four fifths of cases are asymptomatic, China figures indicate*. 2020.
33. B. De Finetti. *Fondamenti logici del ragionamento probabilistico*. Azzoguidi, 1930.
34. J. Dehning, J. Zierenberg, F. P. Spitzner, M. Wibral, J. P. Neto, M. Wilczek, and V. Priesemann. "Inferring change points in the spread of COVID-19 reveals the effectiveness of interventions". *Science*, 2020.
35. T. S. Deisboeck and G. S. Stamatakos. *Multiscale cancer modeling*. CRC press, 2010.
36. G. Detommaso, J. Kruse, L. Ardizzone, C. Rother, U. Köthe, and R. Scheichl. "HINT: Hierarchical invertible neural transport for general and sequential Bayesian inference". *arXiv:1905.10687*, 2019.
37. L. Dinh, D. Krueger, and Y. Bengio. "Nice: Non-linear independent components estimation". *arXiv preprint arXiv:1410.8516*, 2014.
38. L. Dinh, J. Sohl-Dickstein, and S. Bengio. "Density estimation using real nvp". *arXiv preprint arXiv:1605.08803*, 2016.
39. M. P. DOUGHERTY, C. GETTYS, and E. OGDEN. "MINERVA-DM: A memory processes model for judgments of likelihood". *Psychological review* 106:1, 1999, pp. 180–209.
40. C. Durkan, I. Murray, and G. Papamakarios. "On contrastive learning for likelihood-free inference". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2771–2781.
41. K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. "Towards an empirical foundation for assessing bayesian optimization of hyperparameters". In: *NIPS workshop on Bayesian Optimization in Theory and Practice*. Vol. 10. 2013, p. 3.
42. S. Engblom, R. Eriksson, and S. Widgren. "Bayesian epidemiological modeling over high-resolution network data". *Epidemics* 32, 2020, p. 100399.
43. A. Etz, Q. F. Gronau, F. Dablander, P. A. Edelsbrunner, and B. Baribault. "How to become a Bayesian in eight easy steps: An annotated reading list". *Psychonomic Bulletin & Review* 25:1, 2018, pp. 219–234.
44. N. J. Evans and M. Servant. "A comparison of conflict diffusion models in the flanker task through pseudolikelihood Bayes factors." *Psychological review* 127:1, 2020, p. 114.
45. S. Farrell and S. Lewandowsky. *Computational modeling of cognition and behavior*. Cambridge University Press, 2018.
46. A. Foster, M. Jankowiak, E. Bingham, P. Horsfall, Y. W. Teh, T. Rainforth, and N. Goodman. "Variational bayesian optimal experimental design". *arXiv preprint arXiv:1903.05480*, 2019.
47. J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

48. K. Friston. "A theory of cortical responses". *Philosophical transactions of the Royal Society B: Biological sciences* 360:1456, 2005, pp. 815–836.
49. J. Gabry, D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. "Visualization in Bayesian workflow". *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 182:2, 2019, pp. 389–402.
50. F. Galton. *Vox populi*. 1907.
51. A. E. Gelfand. "Gibbs sampling". *Journal of the American statistical Association* 95:452, 2000, pp. 1300–1304.
52. A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.
53. A. Gelman, J. Hwang, and A. Vehtari. "Understanding predictive information criteria for Bayesian models". *Statistics and computing* 24:6, 2014, pp. 997–1016.
54. A. Gelman and X.-L. Meng. "Simulating normalizing constants: From importance sampling to bridge sampling to path sampling". *Statistical science*, 1998, pp. 163–185.
55. A. Gelman, D. B. Rubin, et al. "Inference from iterative simulation using multiple sequences". *Statistical science* 7:4, 1992, pp. 457–472.
56. A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák. "Bayesian workflow". *arXiv preprint arXiv:2011.01808*, 2020.
57. F. A. Gers, J. A. Schmidhuber, and F. A. Cummins. "Learning to forget: continual prediction with LSTM". *Neural Computation* 12:10, 2000, pp. 2451–2471.
58. T. Gneiting, F. Balabdaoui, and A. E. Raftery. "Probabilistic forecasts, calibration and sharpness". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69:2, 2007, pp. 243–268.
59. T. Gneiting and A. E. Raftery. "Strictly proper scoring rules, prediction, and estimation". *Journal of the American Statistical Association* 102:477, 2007, pp. 359–378.
60. D. Greenberg, M. Nonnenmacher, and J. Macke. "Automatic posterior transformation for likelihood-free inference". In: *International Conference on Machine Learning*. 2019, pp. 2404–2414.
61. A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. "A kernel two-sample test". *The Journal of Machine Learning Research* 13:1, 2012, pp. 723–773.
62. Q. F. Gronau, A. Sarafoglou, D. Matzke, A. Ly, U. Boehm, M. Marsman, D. S. Leslie, J. J. Forster, E.-J. Wagenmakers, and H. Steingroever. "A tutorial on bridge sampling". *Journal of mathematical psychology* 81, 2017, pp. 80–97.
63. P. Grünwald, T. Van Ommen, et al. "Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it". *Bayesian Analysis* 12:4, 2017, pp. 1069–1103.
64. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On calibration of modern neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330.

65. F. Hartig, J. M. Calabrese, B. Reineking, T. Wiegand, and A. Huth. "Statistical inference for stochastic simulation models–theory and application". *Ecology letters* 14:8, 2011, pp. 816–827.

66. W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications", 1970.

67. R. Heaton. "Wisconsin card sorting test manual; revised and expanded". *Psychological Assessment Resources*, 1981, pp. 5–57.

68. J. Hermans, V. Begy, and G. Louppe. "Likelihood-free MCMC with approximate likelihood ratios". *arXiv preprint*, 2019.

69. J. M. Hernández-Lobato and R. Adams. "Probabilistic backpropagation for scalable learning of bayesian neural networks". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1861–1869.

70. A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". *The Journal of physiology* 117:4, 1952, pp. 500–544.

71. M. D. Hoffman and A. Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.* 15:1, 2014, pp. 1593–1623.

72. T. Hu, Z. Chen, H. Sun, J. Bai, M. Ye, and G. Cheng. "Stein neural sampler". *arXiv preprint arXiv:1810.03545*, 2018.

73. F. Huber, T. Krisztin, and P. Piribauer. "Forecasting global equity indices using large Bayesian VARs". *Bulletin of Economic Research* 69:3, 2017, pp. 288–308.

74. E. Hüllermeier and W. Waegeman. "Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction". *arXiv preprint arXiv:1910.09457*, 2019.

75. E. M. Izhikevich. "Simple model of spiking neurons". *IEEE Transactions on neural networks* 14:6, 2003, pp. 1569–1572.

76. E. M. Izhikevich. "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks* 15:5, 2004, pp. 1063–1070.

77. E. T. Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

78. A. R. Jensen. *Clocking the mind: Mental chronometry and individual differences*. Elsevier, 2006.

79. B. Jiang, T.-y. Wu, C. Zheng, and W. H. Wong. "Learning summary statistic for approximate Bayesian computation via deep neural network". *Statistica Sinica*, 2017, pp. 1595–1618.

80. A. Jsang. *Subjective Logic: A formalism for reasoning under uncertainty*. Springer Publishing Company, Incorporated, 2018.

81. M. J. Keeling and P. Rohani. *Modeling infectious diseases in humans and animals*. Princeton University Press, 2011.

82. A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 5580–5590.

83. T. H. Kim, M. S. Sajjadi, M. Hirsch, and B. Scholkopf. “Spatio-temporal transformer network for video restoration”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 106–122.

84. D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980*, 2014.

85. D. P. Kingma, T. Salimans, and M. Welling. “Variational dropout and the local reparameterization trick”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*. 2015, pp. 2575–2583.

86. D. P. Kingma, M. Welling, et al. “An Introduction to Variational Autoencoders”. *Foundations and Trends® in Machine Learning* 12:4, 2019, pp. 307–392.

87. D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. *arXiv preprint arXiv:1312.6114*, 2013.

88. D. P. Kingma and P. Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”, 2018. arXiv: 1807.03039 [stat.ML].

89. D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. “Improved variational inference with inverse autoregressive flow”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4743–4751.

90. E. Klinger, D. Rickert, and J. Hasenauer. “pyABC: distributed, likelihood-free inference”. *Bioinformatics* 34:20, 2018, pp. 3591–3593.

91. D. C. Knill and A. Pouget. “The Bayesian brain: the role of uncertainty in neural coding and computation”. *TRENDS in Neurosciences* 27:12, 2004, pp. 712–719.

92. I. Kobyzev, S. Prince, and M. Brubaker. “Normalizing flows: An introduction and review of current methods”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.2992934. URL: http://dx.doi.org/10.1109/TPAMI.2020.2992934.

93. L. Konicar, S. Radev, K. Prillinger, M. Klöbl, R. Diehm, N. Birbaumer, R. Lanzenberger, P. Plener, and L. Poustka. “Volitional modification of brain activity in adolescents with Autism Spectrum Disorder: A Bayesian analysis of Slow Cortical Potential neurofeedback”. *NeuroImage: Clinical*, 2021, p. 102557.

94. M. von Krause, S. T. Radev, and A. Voss. “Processing speed is high until age 60: insights from Bayesian modeling in a one million sample (with a little help of deep learning)”, Manuscript submitted for publication.

95. S. Levy. “Wealthy people and fat tails: An explanation for the Lévy distribution of stock returns”, 1998.

96. Y. Li and Y. Gal. “Dropout inference in Bayesian neural networks with alpha-divergences”. In: *International conference on machine learning*. PMLR. 2017, pp. 2052–2061.

97. H. Lin and S. Jegelka. "Resnet with one-neuron hidden layers is a universal approximator". In: *Advances in Neural Information Processing Systems*. 2018, pp. 6169–6178.
98. R. Liu and J. Zou. "The effects of memory replay in reinforcement learning". In: *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2018, pp. 478–485.
99. C. Louizos and M. Welling. "Multiplicative normalizing flows for variational bayesian neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2218–2227.
100. J.-M. Lueckmann, G. Bassetto, T. Karaletsos, and J. H. Macke. "Likelihood-free inference with emulator networks". In: *Symposium on Advances in Approximate Bayesian Inference*. PMLR. 2019, pp. 32–53.
101. J.-M. Lueckmann, P. J. Gonçalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. "Flexible statistical inference for mechanistic models of neural dynamics". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1289–1299.
102. D. J. MacKay. "Bayesian neural networks and density networks". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 354:1, 1995, pp. 73–80.
103. D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
104. J.-M. Marin, P. Pudlo, A. Estoup, and C. Robert. *Likelihood-free model choice*. Chapman and Hall/CRC Press Boca Raton, FL, 2018.
105. D. W. Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". *Journal of the society for Industrial and Applied Mathematics* 11:2, 1963, pp. 431–441.
106. D. Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.
107. U. K. Mertens, A. Voss, and S. Radev. "ABrox—A user-friendly Python module for approximate Bayesian computation with a focus on model comparison". *PloS one* 13:3, 2018, e0193981.
108. M. Mestdagh, S. Verdonck, K. Meers, T. Loossens, and F. Tuerlinckx. "Prepaid parameter estimation without likelihoods". *PLoS computational biology* 15:9, 2019, e1007181.
109. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. "Equation of state calculations by fast computing machines". *The journal of chemical physics* 21:6, 1953, pp. 1087–1092.
110. J. I. Myung, D. R. Cavagnaro, and M. A. Pitt. "A tutorial on adaptive design optimization". *Journal of mathematical psychology* 57:3-4, 2013, pp. 53–67.
111. B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. "A modern take on the bias-variance tradeoff in neural networks". *arXiv preprint arXiv:1810.08591*, 2018.

112. R. M. Neal et al. "MCMC using Hamiltonian dynamics". *Handbook of markov chain monte carlo* 2:11, 2011, p. 2.

113. K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel. "Asymptotically unbiased estimation of physical observables with neural samplers". *Physical Review E* 101:2, 2020, p. 023304.

114. S. Nowozin, B. Cseke, and R. Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization". In: *Advances in neural information processing systems*. 2016, pp. 271–279.

115. D. P. Oran and E. J. Topol. "Prevalence of Asymptomatic SARS-CoV-2 Infection: A Narrative Review". *Annals of Internal Medicine*, 2020.

116. P. Orbanz and D. M. Roy. "Bayesian models of graphs, arrays and other exchangeable random structures". *IEEE transactions on pattern analysis and machine intelligence* 37:2, 2014, pp. 437–461.

117. B. Paige and F. Wood. "Inference networks for sequential Monte Carlo in graphical models". *International Conference on Machine Learning* 48, 2016, pp. 3040–3049.

118. J. J. Palestro, G. Bahg, P. B. Sederberg, Z.-L. Lu, M. Steyvers, and B. M. Turner. "A tutorial on joint models of neural and behavioral measures of cognition". *Journal of Mathematical Psychology* 84, 2018, pp. 20–48.

119. J. J. Palestro, P. B. Sederberg, A. F. Osth, T. Van Zandt, and B. M. Turner. *Likelihood-free methods for cognitive science*. Springer, 2018.

120. S. Palminteri, V. Wyart, and E. Koechlin. "The importance of falsification in computational cognitive modeling". *Trends in cognitive sciences* 21:6, 2017, pp. 425–433.

121. G. Papamakarios and I. Murray. "Fast ε-free inference of simulation models with Bayesian conditional density estimation". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 1036–1044.

122. G. Papamakarios, D. Sterratt, and I. Murray. "Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 837–848.

123. M. Park, W. Jitkrittum, and D. Sejdinovic. "K2-ABC: Approximate Bayesian computation with kernel embeddings". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 398–407.

124. S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1672–1678.

125. M. D. Parno and Y. M. Marzouk. "Transport Map Accelerated Markov Chain Monte Carlo". *SIAM/ASA Journal on Uncertainty Quantification* 6:2, 2018, pp. 645–682.

126. J. Piironen and A. Vehtari. "Comparison of Bayesian predictive methods for model selection". *Statistics and Computing* 27:3, 2017, pp. 711–735.

127. M. Plummer et al. "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling". In: *Proceedings of the 3rd international workshop on distributed statistical computing*. Vol. 124. 125.10. Vienna, Austria. 2003, pp. 1–10.

128. P. Poletti, M. Tirani, D. Cereda, F. Trentini, G. Guzzetta, G. Sabatino, V. Marziano, A. Castrofino, F. Grosso, G. Del Castillo, et al. "Probability of symptoms and critical disease after SARS-CoV-2 infection". *arXiv preprint arXiv:2006.08471*, 2020.

129. M. Pospischil, Z. Piwkowska, T. Bal, and A. Destexhe. "Comparison of different neuron models to conductance-based post-stimulus time histograms obtained in cortical pyramidal cells using dynamic-clamp in vitro". *Biological cybernetics* 105:2, 2011, p. 167.

130. M. Pospischil, M. Toledo-Rodriguez, C. Monier, Z. Piwkowska, T. Bal, Y. Frégnac, H. Markram, and A. Destexhe. "Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons". *Biological cybernetics* 99:4-5, 2008, pp. 427–441.

131. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

132. P. Pudlo, J.-M. Marin, A. Estoup, J.-M. Cornuet, M. Gautier, and C. P. Robert. "Reliable ABC model choice via random forests". *Bioinformatics* 32:6, 2015, pp. 859–866.

133. S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe. "BayesFlow: Learning complex stochastic models with invertible neural networks". *IEEE Transactions on Neural Networks and Learning Systems*, 2020, pp. 1–15. DOI: 10.1109/TNNLS.2020.3042395.

134. S. T. Radev, M. D'Alessandro, P.-C. Bürkner, U. K. Mertens, A. Voss, and U. Köthe. "Amortized Bayesian model comparison with evidential deep learning", Manuscript submitted for publication.

135. S. T. Radev, F. Graw, S. Chen, N. Mutters, V. Eichel, T. Bärnighausen, and U. Köthe. "Model-based Bayesian inference of disease outbreak with invertible neural networks". *arXiv preprint arXiv:2010.00300*, 2020.

136. S. T. Radev, U. K. Mertens, A. Voss, and U. Köthe. "Towards end-to-end likelihood-free inference with convolutional neural networks". *British Journal of Mathematical and Statistical Psychology* 73:1, 2020, pp. 23–43.

137. S. T. Radev, A. Voss, E. M. Wieschen, and P.-C. Bürkner. "Amortized Bayesian inference for models of cognition". *International Conference on Cognitive Modelling (ICCM) Conference Proceedings*, 2020.

138. R. Ratcliff, A. Thapar, P. Gomez, and G. McKoon. "A diffusion model analysis of the effects of aging in the lexical-decision task." *Psychology and aging* 19:2, 2004, p. 278.

139. O. Ratmann, C. Andrieu, C. Wiuf, and S. Richardson. "Model criticism based on likelihood-free inference, with an application to protein network evolution". *Proceedings of the National Academy of Sciences* 106:26, 2009, pp. 10576–10581.

140. L. Raynal, J.-M. Marin, P. Pudlo, M. Ribatet, C. P. Robert, and A. Estoup. "ABC random forests for Bayesian parameter inference". *Bioinformatics* 35:10, 2018, pp. 1720–1728.

141. C. Robert and G. Casella. "A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data". *Statistical Science*, 2011, pp. 102–115.

142. N. Said, M. Engelhart, C. Kirches, S. Körkel, and D. V. Holt. "Applying mathematical optimization methods to an ACT-R instance-based learning model". *PloS one* 11:7, 2016, e0158832.

143. T. A. Salthouse. "Selective review of cognitive aging". *Journal of the International Neuropsychological Society: JINS* 16:5, 2010, p. 754.

144. D. J. Schad, M. Betancourt, and S. Vasishth. "Toward a principled Bayesian workflow in cognitive science." *Psychological Methods*, 2020.

145. M. Sensoy, L. Kaplan, and M. Kandemir. "Evidential deep learning to quantify classification uncertainty". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3179–3189.

146. G. Shafer. *A mathematical theory of evidence*. Vol. 42. Princeton university press, 1976.

147. T. Shiono. "Estimation of agent-based models using Bayesian deep learning approach of BayesFlow". *Available at SSRN 3640351*, 2020.

148. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". *nature* 529:7587, 2016, pp. 484–489.

149. S. Steegen, F. Tuerlinckx, A. Gelman, and W. Vanpaemel. "Increasing transparency through a multiverse analysis". *Perspectives on Psychological Science* 11:5, 2016, pp. 702–712.

150. M. Sunnåker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. "Approximate bayesian computation". *PLoS computational biology* 9:1, 2013, e1002803.

151. I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". *arXiv preprint arXiv:1409.3215*, 2014.

152. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

153. N. A. Taatgen, C. Lebiere, and J. R. Anderson. "Modeling paradigms in ACT-R". *Cognition and multi-agent interaction: From cognitive modeling to social simulation*, 2006, pp. 29–52.

154. N. N. Taleb. *The black swan: The impact of the highly improbable*. Vol. 2. Random house, 2007.

155. S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman. "Validating Bayesian inference algorithms with simulation-based calibration". *arXiv preprint*, 2018.

156. V. Tchuiev and V. Indelman. "Inference over distribution of posterior class probabilities for reliable bayesian classification and object-level perception". *IEEE Robotics and Automation Letters* 3:4, 2018, pp. 4329–4336.

157. T. Toni. "ABC SMC for parameter estimation and model selection with applications in systems biology". *Nature Precedings*, 2011, pp. 1–1.

158. T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". *Journal of the Royal Society Interface* 6:31, 2009, pp. 187–202.

159. B. M. Turner, B. U. Forstmann, M. Steyvers, et al. *Joint models of neural and behavioral data*. Springer, 2019.

160. B. M. Turner and P. B. Sederberg. "A generalized, likelihood-free method for posterior estimation". *Psychonomic bulletin & review* 21:2, 2014, pp. 227–250.

161. B. M. Turner, P. B. Sederberg, and J. L. McClelland. "Bayesian analysis of simulation-based models". *Journal of Mathematical Psychology* 72, 2016, pp. 191–199.

162. M. Usher and J. L. McClelland. "The time course of perceptual choice: the leaky, competing accumulator model." *Psychological review* 108:3, 2001, p. 550.

163. S. Van Erp, J. Mulder, and D. L. Oberski. "Prior sensitivity analysis in default Bayesian structural equation modeling." *Psychological Methods* 23:2, 2018, p. 363.

164. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need". *arXiv preprint arXiv:1706.03762*, 2017.

165. A. Vehtari, A. Gelman, and J. Gabry. "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC". *Statistics and computing* 27:5, 2017, pp. 1413–1432.

166. A. Vehtari, J. Ojanen, et al. "A survey of Bayesian predictive methods for model assessment, selection and comparison". *Statistics Surveys* 6, 2012, pp. 142–228.

167. R. Verity, L. C. Okell, I. Dorigatti, P. Winskill, C. Whittaker, N. Imai, G. Cuomo-Dannenburg, H. Thompson, P. G. Walker, H. Fu, et al. "Estimates of the severity of coronavirus disease 2019: a model-based analysis". *The Lancet Infectious Diseases*, 2020.

168. G. Viswanathan, E. Raposo, and M. Da Luz. "Lévy flights and superdiffusion in the context of biological encounters and random searches". *Physics of Life Reviews* 5:3, 2008, pp. 133–150.

169. A. Voss, V. Lerche, U. Mertens, and J. Voss. "Sequential sampling models with variable boundaries and non-normal noise: A comparison of six models". *Psychonomic bulletin & review* 26:3, 2019, pp. 813–832.

170. J. Walker, C. Doersch, A. Gupta, and M. Hebert. "An uncertain future: Forecasting from static images using variational autoencoders". In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851.

171. S. Watanabe. "WAIC and WBIC are information criteria for singular statistical model evaluation". In: *Proceedings of the Workshop on Information Theoretic Methods in Science and Engineering*. 2013, pp. 90–94.

172. E. M. Wieschen, A. Voss, and S. Radev. "Jumping to conclusion? a lévy flight model of decision making". *TQMP* 16:2, 2020, pp. 120–132.

173. K. Xu, B. Nosek, and A. Greenwald. "Psychology data from the race implicit association test on the project implicit demo website". *Journal of Open Psychology Data* 2:1, 2014.

174. K. Xu, J. Li, M. Zhang, S. S. Du, K.-i. Kawarabayashi, and S. Jegelka. "What can neural networks reason about?" *arXiv preprint arXiv:1905.13211*, 2019.

175. Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. "Improved variational autoencoders for text modeling using dilated convolutions". In: *International conference on machine learning*. PMLR. 2017, pp. 3881–3890.

176. Y. Yao, A. Vehtari, D. Simpson, A. Gelman, et al. "Using stacking to average Bayesian predictive distributions (with discussion)". *Bayesian Analysis* 13:3, 2018, pp. 917–1007.

177. T. Yarkoni and J. Westfall. "Choosing prediction over explanation in psychology: Lessons from machine learning". *Perspectives on Psychological Science* 12:6, 2017, pp. 1100–1122.

178. S. Zhao, J. Song, and S. Ermon. "Infovae: Information maximizing variational autoencoders". *arXiv preprint arXiv:1706.02262*, 2017.

179. V. M. Zolotarev. *One-dimensional stable distributions*. Vol. 65. American Mathematical Soc., 1986.