Statistics and Computing (2026) 36:162
https://doi.org/10.1007/s11222-026-10911-y

ORIGINAL PAPER

Amortized Bayesian Mixture Models

Šimon Kucharský[1] · Paul-Christian Bürkner[1]

Received: 17 January 2025 / Accepted: 20 May 2026

Abstract
Finite mixtures are a broad class of models useful in scenarios where observed data is generated by multiple distinct processes but without explicit information about the responsible process for each data point. Estimating Bayesian mixture models is computationally challenging due to issues such as high-dimensional posterior inference and label switching. Furthermore, traditional methods such as MCMC are applicable only if the likelihoods for each mixture component are analytically tractable. Amortized Bayesian Inference (ABI) is a simulation-based framework for estimating Bayesian models using generative neural networks. This allows the fitting of models without explicit likelihoods, and provides fast inference. ABI is therefore an attractive framework for estimating mixture models. This paper introduces a novel extension of ABI tailored to mixture models. We factorize the posterior into a distribution of the parameters and a distribution of (categorical) mixture indicators, which allows us to use a combination of generative neural networks for parameter inference, and classification networks for mixture membership identification. The proposed framework accommodates both independent and dependent mixture models, enabling filtering and smoothing. We validate and demonstrate our approach through synthetic and real-world datasets.

Keywords Finite Mixture Models · Bayesian inference · Simulation based inference · Amortized inference

1 Introduction

Fast and accurate estimation of statistical quantities is an ongoing problem in statistical research (Cranmer et al. 2020; Hermans et al. 2022; Papamakarios and Murray 2016). One major difficulty arises when the observed data is generated by multiple distinct processes, but the specific process responsible for each data point is unknown. Mixture models are commonly used to address this issue (McLachlan and Basford 1988; Frühwirth-Schnatter 2006; Zucchini et al. 2016; Visser and Speekenbrink 2022; Scrucca et al. 2023).

Although mixture models have been useful across a wide range applications (e.g., Schaaf et al. 2019; Zavadskiy et al. 2024; Kucharský, v., Tran, N.- H., Veldkamp, K., Raijmakers, M., Visser, I. 2021; Hadj-Amar et al. 2023, to name a few), estimating mixture models can be challenging for several reasons. In practice, the most relevant obstacles are: (1) obtaining full Bayesian inference for mixtures can be computationally demanding, (2) fitting Bayesian mixtures using standard methods (such as MCMC) requires the model likelihood to be analytically tractable, and (3) label switching issues, where the likelihood is invariant to permutations of mixture component labels, which makes the model identified only up to the permutation of the labels.

Given these challenges, there is a need for methods that can perform fast and accurate inference for mixture models while removing limitations of traditional methods. A promising candidate for resolving these issues is Amortized Bayesian Inference (ABI, Gershman and Goodman 2014; Radev et al. 2022; Ritchie et al. 2016; Papamakarios and Murray 2016), which offers fast approximation of the posterior distributions even for models that are otherwise not analytically tractable. However, these methods have yet to be fully adapted to handle mixture models, in particular, when we desire estimating the parameters of the mixture model and the categorical latent mixture indicators as a joint probability distribution. This gap motivates the development of a new approach that uses modern deep learning techniques to enable scalable ABI for mixture models.

Our work provides the following contributions:

- We develop a deep learning architecture that takes advantage of a factorization of mixture models where parameter posteriors can be estimated directly from data, and mix-

✉ Šimon Kucharský
simon.kucharsky@tu-dortmund.de

[1] Department of Computational Statistics, TU Dortmund University, Dortmund, Germany

Published online: 11 June 2026

ture membership classification is estimated based on data and the parameter estimates. This allows full Bayesian inference providing the joint posterior distribution of all quantities of interest.
- The implementation is amortized, meaning that the inference is considerably faster than traditional – non-amortized – methods, such as MCMC or other Approximate Bayesian Computation methods.
- The amortized mixture implementation extends amortized posterior estimation, which is typically limited to continuous quantities, with the amortized estimation of categorical latent variables with bounded cardinality.
- The proposed framework is able to handle independent mixture models as well as dependent mixtures. In the case of dependent mixtures, our method is able to perform both filtering: inferring the latent state at each time given observations up to that time, and smoothing: inferring latent states using the full sequence of observations.
- We validate our method on a battery of toy and real world examples. Where possible, we compare our results to that of `Stan` (Carpenter et al. 2017) as a gold standard for MCMC estimation.
- The method is implemented in a `Python` (Rossum and Drake 2010) library `BayesFlow` (Radev et al. 2022) that provides user-friendly interface for ABI.

The article is organized as follows. Section 2 describes the methods used and developed in this article. Section 2.1 explains Bayesian mixture models and describes challenges when fitting them with traditional methods. Section 2.2 and Section 2.3 provide a primer explaining existing methods for simulation-based amortized inference that our approach is built upon. Section 2.4 describes methods developed in this article that implement amortized estimation of mixture models, and Section 2.5 discusses possible alternative approaches. Section 3 presents applications of our method in three case studies - two simple toy examples to showcase how the approach works for independent and dependent mixtures, and one empirical study that showcases inferences on real data. We summarize our work in Section 4.

2 Methods

In this section, we introduce our proposed ABI framework for Bayesian mixture models. First, we will define Bayesian mixture models and highlight challenges that arise when fitting them with traditional, non-amortized methods. Second, we will explain the general idea behind simulation based inference and ABI. Then, we will explain how ABI can be extended to provide amortized inference for mixture models.

2.1 Bayesian mixture models

We define Bayesian mixture models as a joint distribution $p(y, z, \theta)$: $y \in \mathcal{Y}$ represents all relevant observable variables, whereas $z \in \mathbb{N}$ and $\theta \in \mathcal{R}$ represent typically latent (unobservable) quantities; z represents the latent mixture indicators, while θ represents parameters of the mixture model: the mixture weights as well as parameters required by all mixture components. We focus on situations where the model is assumed (or known) to consist of a mixture of K populations (or processes), each with their own model $p(y_i \mid z_i, \theta)$ for the observable data $y_{1,\ldots,N}$, each associated with a latent mixture membership indicator $z_{1,\ldots,N} \in \{1, \ldots, K\}$.

We assume that a generative model can be created by factorizing the joint distribution of the mixture model into a prior $p(z, \theta)$ and a likelihood $p(y \mid z, \theta)$: $p(y, z, \theta) = p(z, \theta)p(y \mid z, \theta)$. Further, we assume that it is possible to obtain random samples of $z^{(s)}$ and $\theta^{(s)}$ from the prior, and it is possible to obtain random samples of synthetic data $y^{(s)}$ from the likelihood, conditionally on $z^{(s)}$ and $\theta^{(s)}$. The generative model can be therefore implemented as a computer program which samples the triple $(z^{(s)}, \theta^{(s)}, y^{(s)})$ as follows:

$$\begin{aligned} z^{(s)}, \theta^{(s)} &\sim p(z, \theta) \\ y^{(s)} &\sim p(y \mid z^{(s)}, \theta^{(s)}). \end{aligned} \tag{1}$$

Typically, most mixture models used in practice can be cast as a special case of this factorization:

$$\begin{aligned} \theta^{(s)} &\sim p(\theta) \\ z^{(s)} &\sim p(z \mid \theta^{(s)}) \\ y^{(s)} &\sim p(y \mid z^{(s)}, \theta^{(s)}). \end{aligned} \tag{2}$$

The aim of Bayesian analysis is to estimate the joint posterior of all unobserved variables, $p(\theta, z \mid y)$. This joint posterior is notoriously too complex to calculate analytically (Diebolt and Robert 1994). Thus, approximation methods are typically needed to estimate the posterior distribution (Frühwirth-Schnatter 2001; Marin et al. 2005).

The estimation of Bayesian mixture models is generally achieved using one of two common approaches, somewhat analogically to using "classification" vs. "mixture" likelihoods for fitting mixture models with maximum likelihood estimation (Mclachlan 1982; Ganesalingam 1989).

Conceptually, the first approach samples from the joint posterior of the model parameters θ and latent indicators z directly:

$$p(\theta, z \mid y) \propto p(\theta) \prod_{i=1}^{N} p(z_i \mid \theta) p(y_i \mid z_i, \theta). \tag{3}$$

Such approach is typically implemented using MCMC with Gibbs sampling or its extensions (Diebolt and Robert 1994; Marin et al. 2005; Celeux et al. 2000). The obvious computational obstacle is that given a set of N data points each generated from one of K processes, there is K^N possible mixture membership permutations and it may be extremely difficult to sample from this distribution (Marin et al. 2005). The MCMC sampler may venture into a set of "trapping states" from which it may take an enormous number of steps to escape from (Marin et al. 2005; Diebolt and Robert 1994; Celeux et al. 2000). Although sampling from the full joint distribution is feasible for relatively small sample sizes and number of mixture components using MCMC samplers, such approach tends to scale badly with increasing sample size and with more mixture components.

Probabilistic programming languages such as `Stan` (Carpenter et al. 2017) do not allow such implementation in the first place, because gradient-based sampling methods like Hamiltonian Monte Carlo require continuous variables for evaluating the gradients. As such, using discrete parameters directly in the MCMC is not allowed. Instead, models with discrete parameters must be handled with alternative approaches. The most common alternative is to factorize the joint posterior as $p(\theta, z \mid y) = p(\theta \mid y)p(z \mid \theta, y)$; first we sample from the posterior distribution of θ with z marginalized out,

$$p(\theta \mid y) \propto p(\theta) \prod_{i=1}^{N} p(y_i \mid \theta), \tag{4}$$

where $p(y_i \mid \theta) = \sum_{k=1}^{K} p(z_i = k \mid \theta) p(y_i \mid z_i = k, \theta)$ is the likelihood of the observations with the mixture indicators being marginalized out. Subsequently, the distribution of $p(z \mid \theta, y)$ can be fully determined conditionally on the parameters θ and observations y:

$$p(z_i \mid y_i, \theta) = \frac{p(z_i \mid \theta) p(y_i \mid z_i, \theta)}{p(y_i \mid \theta)}. \tag{5}$$

Both factorizations described above are valid only if the latent indicators are independent and observations are conditionally independent given latent indicators (i.e., in the case of exchangeable data). In many cases, this assumption is not satisfied. For example, in hidden Markov models (HMMs; Rabiner 1989), the observables are indeed independent conditionally on the latent indicators, but the latent indicators themselves form a Markov chain: the probability of the current state depends on the previous state(s). Other types of mixture models might exhibit other kinds of dependencies between observables or states – depending on the exact nature of these dependencies, other forms of factorizations might be possible (for examples, see May et al. 2024; Samé 2020; Ambroise et al. 1997; Hadj-Amar et al. 2023).

In any case, estimating mixture parameters and mixture memberships requires evaluating the likelihood density. As a result, the likelihood has to be analytically tractable under each mixture component for applicability of the above described density-based approaches.

2.2 Simulation-Based Inference

Statistical inference requires specifying the likelihood function $p(y \mid \theta)$ that describes the link between parameters and data. However, scientific models are often formulated only as a *simulation program* that may render the likelihood analytically intractable (Cranmer et al. 2020), for example, because the model involves differential equations without analytical solutions, or other complex generative procedures (e.g., Radev et al. 2022; Brehmer 2021; Lueckmann et al. 2021; Boelts et al. 2022).

The Bayesian model is then available only as a probabilistic generative model of the triple of the prior $p(\theta)$ for model parameters θ, a stochastic model $p(\nu \mid \theta)$ for nuisance variables (i.e., noise) ν, and a *simulation program* $g : (\theta, \nu) \to y$ that generates synthetic data y. The complete forward model can be defined as

$$y = g(\theta, \nu) \text{ with } \nu \sim p(\nu \mid \theta), \theta \sim p(\theta). \tag{6}$$

We can sample from this stochastic model repeatedly to obtain the pairs of data-generating parameters θ along with the observable data y. The likelihood is implicitly defined as an integral over all possible execution paths of the generative model (represented by the stochastic variables ν),

$$p(y \mid \theta) = \int p(y, \nu \mid \theta) d\nu, \tag{7}$$

even though the analytic solution to that integral may be unknown and therefore an explicit analytic form of the likelihood $p(y \mid \theta)$ unavailable (Cranmer et al. 2020).

Estimating statistical models with intractable likelihoods is commonly referred to as *likelihood-free inference*, even though as presented in Equation 7, the likelihood function exists, albeit it may be implicit. A more apt designation of inference without explicit analytic likelihoods is *simulation-based inference* (SBI, Cranmer et al. 2020), since these approaches rely on using (often extensive) Monte Carlo simulations from the generative model to perform inference on the model parameters.

A downside of SBI methods is computational complexity. When the likelihood is tractable, density-based methods (e.g., MCMC) are often preferred over SBI, because using the likelihood directly typically requires less computational

resources to achieve comparable accuracy (Schmitt et al. 2024c; Zeghal et al. 2022; Brehmer et al. 2020).

2.3 Amortized Bayesian Inference

Computational complexity remains a significant challenge for methods that approximate posterior distributions (Papamakarios and Murray 2016). When posterior inference must be performed repeatedly, such as when applying a model to multiple datasets or during model evaluation procedures like simulation-based calibration (Talts et al. 2018) and cross-validation (see e.g., Vehtari et al. 2017; Bürkner et al. 2020), the cost of standard SBI methods but also MCMC can quickly become prohibitive. In such cases, standard methods may be computationally infeasible and thus impractical.

Amortized Bayesian Inference (Gershman and Goodman 2014; Ritchie et al. 2016; Radev et al. 2022; Papamakarios and Murray 2016; Gonçalves et al. 2020) is a solution to both the problem of implicit likelihoods, and the problem of computational demands for inference. ABI divides model fitting into two distinct stages. In the first — *training* — stage, neural networks learn the posterior based on simulated data from the generative model. In the second — *inference* — stage, given any observed data y^{obs}, samples from the posterior distribution are simply obtained by generating samples from the trained inference network. Most of the computational resources are expended during the training stage, allowing us to *amortize* (pay upfront) the cost of inference, making subsequent fitting of the model during the inference stage substantially more effective.

Using neural networks to approximate posterior distributions is often referred to as neural posterior estimation (NPE). In this approach, the target posterior $p(\theta \mid y)$ is represented by a surrogate density $q_\phi(\theta \mid y)$, parametrized by a set of learnable network weights ϕ of an *inference network* f_ϕ. Such network is often implemented as a normalizing flow (Rezende 2015; Dinh et al. 2016; Kobyzev et al. 2020; Papamakarios et al. 2021). Other ML generative models, such as diffusion models (Sharrock et al. 2024), flow matching (Lipman et al. 2022; Dax et al. 2023), consistency models (Schmitt et al. 2024c), and others, can be used for this purpose, and would be compatible with the methods developed in this article. Given that normalizing flows have demonstrated strong performance across various disciplines (e.g., von Krause et al. 2022; Schumacher et al. 2024; Schumacher and Bürkner, P.- C., Voss, A., Köthe, U., Radev, S.T. 2023), we focus on normalizing flows as one example of NPE.

Normalizing flows (Kobyzev et al. 2020; Dinh et al. 2016; Rezende 2015) are implemented using conditional invertible neural networks (CINNs, Ardizzone et al. 2019). The flow serves as a learnable transformation f_ϕ between the posterior (target) distribution and a base distribution, conditioned on the data. The transformation f_ϕ transforms the (potentially intractable) target distribution into this base distribution. The base distribution is typically chosen to be simple and tractable, for which both density evaluation and sampling are easy and efficient, e.g., the Gaussian. Hence the name "normalizing flow": f_ϕ *normalizes* the potentially intractable target distribution to a multivariate Gaussian; if $\theta \sim q_\phi(\theta \mid y)$, then $\xi = f_\phi(\theta;\ y) \sim \text{MvN}(0, \mathbb{I})$.

To sample from the target distribution, we start with the base distribution and reverse the transformation. We first obtain S samples from the base distribution $(\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(S)})$. Then, we pass these samples through the transformation inverse f_ϕ^{-1},

$$\theta^{(s)} = f_\phi^{-1}\left(\xi^{(s)};\ y\right) \text{ for } s \in \{1, \ldots, S\}. \tag{8}$$

Alternatively to sampling, the target density for some point θ is available by evaluating the corresponding density of the base distribution at the point $\xi = f_\phi\left(\theta;\ y\right)$, adjusted for the transformation f_ϕ via the change-of-variables formula,

$$q_\phi\left(\theta \mid y\right) = p_{\text{base}}\Big(\xi = f_\phi\left(\theta;\ y\right)\Big) \left|\det \frac{\partial f_\phi\left(\theta;\ y\right)}{\partial \theta}\right|. \tag{9}$$

Architectures of normalizing flows are specifically designed such that these operations (forward and inverse transform, evaluating the Jacobian adjustment) are computationally efficient but at the same time the transform f_ϕ sufficiently expressive (Durkan et al. 2019; Kobyzev et al. 2020; Dinh et al. 2016; Rezende 2015).

To ensure that the network accurately captures the target posterior, it is trained to minimize the Kullback-Leibler (KL) divergence between the approximated posterior $q_\phi(\theta \mid y)$ and the true posterior $p(\theta \mid y)$. The optimization objective of the inference network can therefore be written as,

$$\begin{aligned}
\widehat{\phi} &= \arg\min_\phi \mathbb{E}_{(\theta,y)\sim p(\theta,y)}\Big[\text{KL}\Big(p(\theta \mid y) \,||\, q_\phi(\theta \mid y)\Big)\Big] \\
&\propto \arg\min_\phi \mathbb{E}_{(\theta,y)\sim p(\theta,y)}\Big[-\log q_\phi(\theta \mid y)\Big] \\
&\approx \arg\min_\phi \frac{1}{M}\sum_{m=1}^{M} -\log q_\phi(\theta^{(m)} \mid y^{(m)}).
\end{aligned} \tag{10}$$

The true posterior density $p(\theta \mid y)$ can be dropped from the equation because it is constant with respect of the learnable network weights ϕ. The expectation is approximated using a Monte-Carlo estimate over M draws generated from the Bayesian model $(\theta^{(m)}, y^{(m)}) \sim p(\theta, y)$. The simulated data $y^{(m)}$ serve as conditioning input to the inference network, which outputs the corresponding approximate posterior densities $q_\phi(\theta^{(m)} \mid y^{(m)})$ according to Eq. 9. The network weights ϕ are optimized by minimizing the negative log-density of the simulated parameters conditionally

on the simulated data (Eq. 10). Because the network is trained on *simulated* parameter-observations pairs $(\theta^{(m)}, y^{(m)})$, ABI falls under the umbrella of SBI.

In addition to the *inference network*, ABI approaches can be expanded with *summary networks*. A summary network compresses raw data y into summary statistics $h_\psi(y)$, i.e., a lower-dimensional representation of the data (embeddings). This simplifies the task of the *inference network* that is therefore concerned with an inference conditioned on a smaller number of informative inputs rather than on a (potentially large) number of individual data points. The architecture of the summary network needs to reflect the structure of the data; for example, permutation invariant networks are suitable to summarize exchangeable data, recurrent neural networks are suitable for time-series data, and so forth. These networks can typically also take input of various size but their output is of fixed length. This allows the ABI amortize over different designs (e.g., varying sample sizes). The inference and summary network are trained concurrently using the modified loss,

$$\widehat{\phi}, \widehat{\psi} = \arg\min_{\phi,\psi} \mathbb{E}_{(\theta,y)\sim p(\theta,y)}\Big[-\log q_\phi(\theta \mid h_\psi(y))\Big]. \quad (11)$$

Given enough capacity of the summary network, the learned summary statistics are approximately sufficient with regards to the target of the posterior inference: Using the summary statistics does not alter the target posterior distribution, if swapped with the raw data: $p(\theta \mid y) = p(\theta \mid h(y))$ (Chen et al. 2021; Radev et al. 2022). Optionally, we might use an additional penalty term that forces the summaries be distributed according to a specified distribution (e.g., a Gaussian). This regularizes the summary network and allows additional model checks, such as detecting *simulation gaps* (i.e., when the data used for inference differ significantly from the data seen during training; Schmitt et al. 2024a).

Once trained, the networks can be used for inference, that is, for posterior estimation given any observed data set y^{obs}. First, the data is passed through the trained summary network to obtain its embedding $h_{\widehat{\psi}}(y^{\text{obs}})$. Then, analogically to Eq 8, sample S draws from the base distribution $(\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(S)})$. The posterior draws $(\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(S)})$ are obtained via the *inverse pass* through the inference network,

$$\theta^{(s)} = f_{\widehat{\phi}}^{-1}\big(\xi^{(s)}; h_{\widehat{\psi}}(y^{\text{obs}})\big) \text{ for } s \in \{1, \ldots, S\}. \quad (12)$$

Alternatively, the approximate posterior density for any values of parameters θ is available analogically to Eq. 9,

$$\begin{aligned} p(\theta \mid y^{\text{obs}}) &\approx q_{\widehat{\phi}}\big(\theta \mid h_{\widehat{\psi}}(y^{\text{obs}})\big) \\ &= p\left(\xi = f_{\widehat{\phi}}\Big(\theta; h_{\widehat{\psi}}\big(y^{\text{obs}}\big)\Big)\right) \left|\det \frac{\partial f_{\widehat{\phi}}\big(\theta; h_{\widehat{\psi}}\big(y^{\text{obs}}\big)\big)}{\partial \theta}\right|. \end{aligned} \quad (13)$$

It is this separation of *training* and *inference* that makes ABI such a promising method. Since sampling from the posterior distribution using ABI only requires generating samples from the base distribution and passing the data through the trained networks (Eq. 12), it is relatively fast compared to non-amortized methods during inference. Often times, models that take minutes, hours or even days to fit with non-amortized methods can take as little time as fractions of a second with amortized methods (Radev et al. 2022; Hermans et al. 2022). The efficiency of ABI during inference is particularly helpful in scenarios where data needs to be fit in real time, when many datasets need to be fit with the same model (e.g., von Krause et al. 2022), when the same data set needs to be fit many times (e.g., cross-validation, Bürkner 2017) – sometimes under different assumptions or processing steps (e.g., for sensitivity analysis, Elsemüller et al. 2024), or for validating the model on a large number of simulations (e.g., simulation-based calibration, Talts et al. 2018). In many cases, such uses of Bayesian models are borderline unfeasible with non-amortized methods (Hermans et al. 2022). ABI makes these procedures within reach in a matter of seconds. Since at no point during training or inference the model likelihood or priors need to be evaluated, this approach is also applicable in scenarios with analytically intractable likelihoods or priors.

2.4 Neural estimation of Mixture Models

The goal is to estimate the joint posterior of the model parameters and the mixture indicators. In the following, we will work with the following factorization:

$$p(\theta, \{z_i\} \mid \{y_{ij}\}) = p(\theta \mid \{y_{ij}\}) p(\{z_i\} \mid \{y_{ij}\}, \theta). \quad (14)$$

Here we used the set notation to highlight that the number of observational units in $i = 1, \ldots, N$ and the number of observations per unit $j = 1, \ldots, P_i$ can vary across different datasets.

Using the factorization above, we separate the problem in two parts, one of which entails estimating the parameter posterior $p(\theta \mid \{y_{ij}\})$ that was already described in Section 2.3. This posterior contains the distribution of all mixture parameters, that is, mixture weights and parameters of all mixture components. What remains is formulating a neural approximation of the second term $p(\{z_i\} \mid \{y_{ij}\}, \theta)$, which yields the distribution of mixture indicators. Since this is essentially a classification problem, common network architectures, such as the multilayer perceptron (MLP, Rosenblatt 1958; Baum 1988; Murtagh 1991), are adequate candidate architectures. However, because the number of observations per unit can vary, the input of the classification network must be brought into the same dimensions. For this reason, we introduce an (optional) summary network h_ω that creates summary

embedding for each observational unit $\{h_\omega(y_i)\}_{i=1}^N$ (similar to the *local* summary network used in amortized multilevel models, Habermann et al. 2024). The task of the local summary network is to extract relevant information about each observational unit. The choice of the local summary network depends on the structure of the data within the units. Independent observations within units may be summarized by permutation-invariant networks such as Deep Sets (Zaheer et al. 2017) or Set Transformers (Lee et al. 2019). Time-series or otherwise dependent measures within units might need to be summarized via networks that take dependencies into account, e.g., convolutional neural networks (CNNs; Lecun et al., 1998), recurrent neural networks (RNNs; Elman, 1990; Hochreiter and Schmidhuber, 1997), Transformers (Vaswani et al. 2017), and so on.

The exact architecture of the classification network depends on structure of the data, which dictates how to factorize the joint distribution $p(z \mid y, \theta)$. Some examples of data structures are shown in Figure 1. In the case of exchangeable observational units, the joint distribution can be factorized as $p(z \mid y, \theta) = \prod_{i=1}^N p(z_i \mid y_i, \theta)$. This means that the probability distribution of each mixture indicator z_i can be approximated independently — conditioned on the parameters θ and observation y_i — with a *classification* network as follows,

$$p(z_i \mid y_i, \theta) \approx q_\alpha(z_i \mid y_i, \theta) = \text{softmax}\Big(f_\alpha\big(h_\omega(y_i), \theta\big)\Big) \text{ for } i \in \{1, \ldots, N\}, \tag{15}$$

here f_α stands for a MLP with a series of hidden layers connected by learnable weights α followed by non-linear activations, and an output layer with K nodes. The final softmax activation converts the output into a set of mutually exclusive probabilities for each class, essentially implementing soft classification (Wahba 2002; Liu et al. 2011). Soft classification refers to the calculation of class probabilities rather than producing classification labels directly, which is typical for hard categorical classification.

When the observational units are not exchangeable (e.g., ordered time series, spatial data), the joint distribution $p(z \mid y, \theta)$ may not be factorized easily due to dependencies across observational units. In such cases, directly modeling the joint probability distribution of all mixture indicators could require complex and computationally expensive architectures (Mark et al. 2018). In this article, we address this problem using *local decoding*: we *assume* that the distribution can be factorized by conditioning on other observational units. Rather than modeling the full joint probability distribution of all mixture indicators (Viterbi 1967; Lember et al. 2019), we approximate each indicator's probability distribution one at a time, significantly simplifying the computational process (Särkkä and Svensson 2023).

One method of local decoding is *filtering* where the probability distribution of mixture indicator z_t is based on the sequence of observations up until the t^{th} data point, denoted as $\{y_i\}_{i=1}^t$,

$$p(z_t \mid \{y_i\}_{i=1}^t, \theta) \approx q_\alpha(z_t \mid \{y_i\}_{i=1}^t, \theta) = \text{softmax}\Big(f_\alpha\big(\{h_\omega(y_i)\}_{i=1}^t, \theta\big)\Big) \text{ for } t \in \{1, \ldots, N\}, \tag{16}$$

where f_α is a *forward* network that takes the set of observations (or their embeddings) and outputs activation pertaining to the last element in the set. A forward network can be any network that captures dependencies between observations, such as recurrent neural networks (RNNs), gated recurrent units (GRUs), transformers, an so forth.

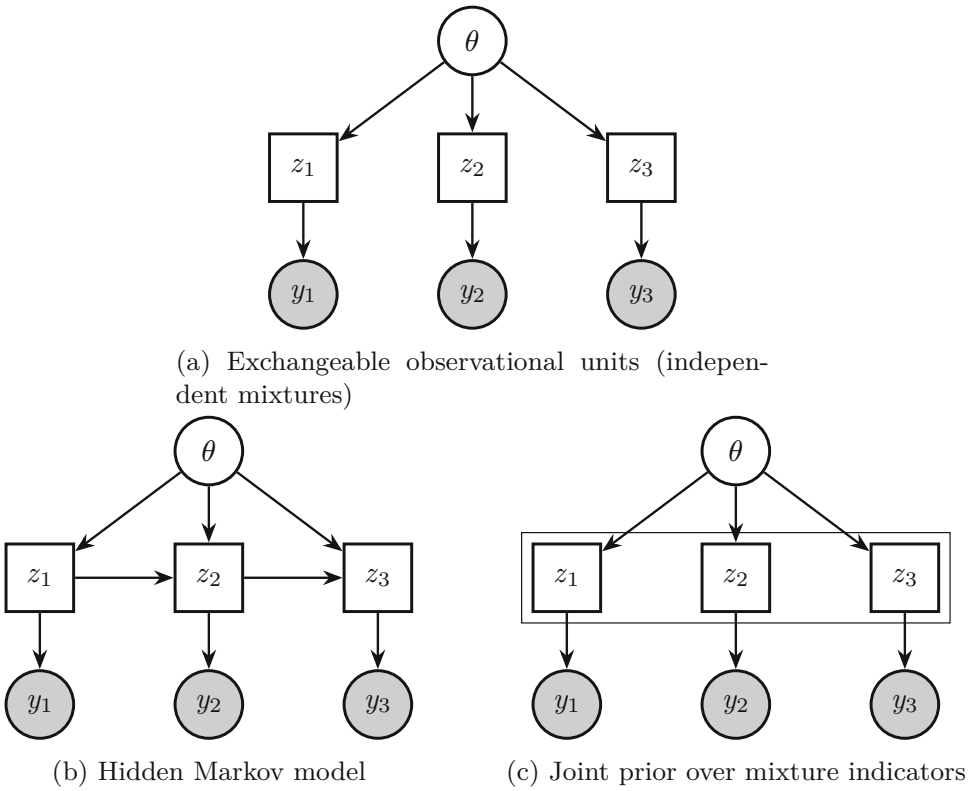
Both in the context of *classification* and in the context of *filtering*, the networks can be trained with realizations from the Bayesian generative model, where the tuple (y, θ) serves as an input, and z is the predicted target. Traditional classification loss functions, such as categorical cross-entropy, are well-suited for this inference problem. Specifically, the following loss function can be used,

$$\widehat{\alpha}, \widehat{\omega} = \arg\min_{\alpha,\omega} \mathbb{E}_{(\theta,y,z)\sim p(\theta,y,z)}\Big[-\log q_\alpha\big(\{z_i\} \mid \{h_\omega(y_i)\}, \theta\big)\Big]. \tag{17}$$

An extension of *filtering* is *smoothing* (qv, Särkkä and Svensson 2023; Mark et al. 2018), where the probability of a mixture indicator z_t is based on all available observations instead of only the observations that precede the current data point. To compute the smoothing estimate, we combine the output of the forward network with an output of a *backward* network f_β. The backward network is applied to the *remaining* observations after t, and processes them in the reversed order denoted as $\{y_i\}_{i=N}^{t+1}$,

$$\begin{aligned} p(z_t \mid \{y_i\}_{i=1}^N, \theta) &\approx q_{\alpha,\beta}(z_t \mid \{y_i\}_{i=1}^N, \theta) \\ &= \text{softmax}\Big(f_\alpha\big(\{h_\omega(y_i)\}_{i=1}^t, \theta\big) + \\ &\quad f_\beta\big(\{h_\omega(y_i)\}_{i=N}^{t+1}, \theta\big)\Big) \text{ for } t \in \{1, \ldots, N\}. \end{aligned} \tag{18}$$

Although the forward and backward networks are, in principle, separate, in many cases it is justified for them to share the same weights such that $\alpha = \beta$. This is because both networks process the same type of information and have the same inferential target (the probability distribution of z). In some cases, the predictions from either the forward network or the backward network might systematically outperform the combined outputs of both networks. In that case, training the networks using the smoothed predictions in Eq. 18 could lead to a solution that effectively mimics the one network which performs better while ignoring the performance



(a) Exchangeable observational units (independent mixtures)

(b) Hidden Markov model

(c) Joint prior over mixture indicators

Fig. 1 Examples of dependencies between observational units in mixture models. (a) Exchangeable observational units permit factorizing the distribution of mixture indicators as $p(z \mid y, \theta) = \prod_{i=1}^{N} p(z_i \mid y_i, \theta)$. For non-exchangeable observational units such as in (b) and (c), local decoding is used to factorize the joint distribution; *filtering* as $p(z \mid y, \theta) = \prod_{i=1}^{N} p(z_i \mid y_1, \ldots, y_i, \theta)$, or *smoothing* as $p(z \mid y, \theta) = \prod_{i=1}^{N} p(z_i \mid y_1, \ldots, y_N, \theta)$. Figure inspired by Bürkner et al. (2021)

of the other network. To avoid such a scenario, instead of using the smoothed classification probabilities from Eq. 18, both networks are trained with their own loss, each based on the classification probabilities in the forward and backward direction, respectively,

$$\begin{aligned}\widehat{\alpha}, \widehat{\beta}, \widehat{\omega} = \underset{\alpha,\beta,\omega}{\arg\min}\, \mathbb{E}_{(\theta,y,z)\sim p(\theta,y,z)} \\ \Big[-\log q_\alpha(\{z_i\} \mid \{h_\omega(y_i)\}, \theta) \\ -\log q_\beta(\{z_i\} \mid \{h_\omega(y_i)\}, \theta)\Big],\end{aligned} \tag{19}$$

to ensure that both networks are optimized for the classification task on their own. Smoothing in Eq. 18 is not part of the training loss. Instead, once the forward and backward networks have been trained separately (Eq. 19), their outputs can be combined according to Eq. 18 to yield smoothed classification probabilities. Thus, Eq. 18 serves purely as a post-training prediction rule rather than a learning rule.

The losses in Eq. 11 and Eq. 17 or Eq. 19 can be combined, and all networks trained concurrently on the same set of training examples generated from the Bayesian generative model. Concurrent training also offers an opportunity for weight sharing between the NPE and classification tasks. Figure 2 shows a schematic representation of joint training of the networks used for posterior and mixture inference. For example, the local summary network h_ω can also be used during NPE to compress each observational unit before the global summary network h_ψ is applied to compress the whole data set for posterior inference, leading to the following training objective involving the local (h_ω) and global (h_ψ) summary networks, the posterior network (q_ϕ), and the forward (q_α) and backward (q_β) networks:

$$\begin{aligned}\widehat{\phi}, \widehat{\psi}, \widehat{\alpha}, \widehat{\omega} = \underset{\phi,\psi,\alpha,\omega}{\arg\min}\, \mathbb{E}_{(\theta,y,z)\sim p(\theta,y,z)} \\ \Big[-\log q_\phi\big(\theta \mid h_\psi(\{h_\omega(y_i)\})\big) \\ -\log q_\alpha(\{z_i\} \mid \{h_\omega(y_i)\}, \theta) \\ -\log q_\beta(\{z_i\} \mid \{h_\omega(y_i)\}, \theta)\Big].\end{aligned} \tag{20}$$

Once the networks are trained, they can be used to make fast inferences from any data. Figure 3 shows a schematic representation of using the networks for inference. The parameter posteriors are sampled as explained in Eq. 12. For the mixture classification, each sample from the posterior, $\theta^{(s)}$, is subsequently used in the mixture membership classification, leading to variations in the classification network output that is a result of the variability in the parameter values. This way, the uncertainty in parameter values is propagated to express the resulting uncertainty in mixture classification.

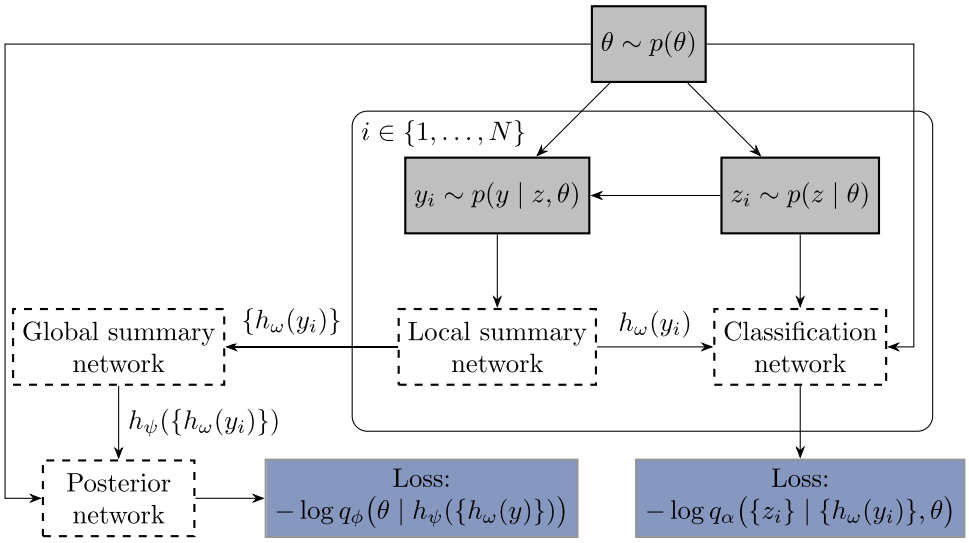


Fig. 2 Schematic representation of training amortized mixture models. The boxes highlighted in gray represent the inputs (i.e., the training set) sampled from the Bayesian generative model. The observations y_i are individually passed through the local summary network. For parameter posterior training, the complete set of local summaries is further passed through the global summary network. The global summary is passed together with the true parameters θ to the posterior network to obtain the loss from Eq. 11. For classification training, the local summaries are concatenated with the true parameters θ, and together passed with the true mixture indicators z_i to the classification network, to obtain the loss from Eq. 17 (or in case of separate forward and backward networks, Eq. 19). Combining the two losses results in the joint loss in Eq. 20. The objective of training is to minimize the total loss by optimizing network weights $\phi, \psi, \omega, \alpha$. To simplify notation, we omitted indices $\theta^{(m)}, y_i^{(m)}, z_i^{(m)}$ indicating the training sample $m \in \{1, \ldots, M\}$

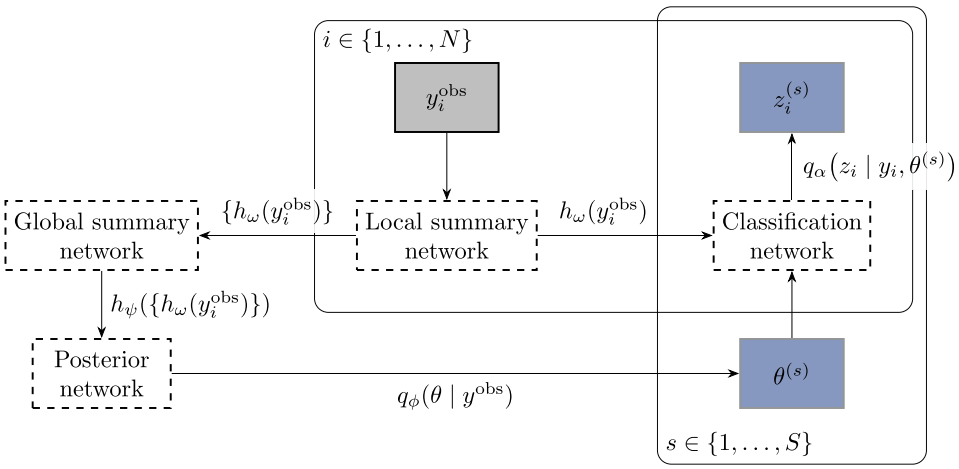


Fig. 3 Schematic representation of the use of amortized mixture models for inference. The observations y_i are individually passed through the local summary network. For parameter posterior inference, the complete set of local summaries is further passed through the global summary network. The global summary is passed to the posterior network to generate samples $\theta^{(s)}$ from the approximate posterior distribution. For classification, the local summaries are concatenated with the parameter samples $\theta^{(s)}$ and passed through the classification network to obtain the approximate mixture membership probabilities. If desired, the mixture indicators z_i can be sampled from this approximate distribution

2.5 Alternative factorizations

The factorization used in Eq. 14 reflects the approach described in Eq. 4 where parameter posteriors are estimated first with the latent mixture indicators marginalized out (and subsequently recomputed). However, as explained in Section 2.1, this is not the only option to estimate mixture models. Instead, it would be possible to use an alternative factorization,

$$p(\theta, \{z_i\} \mid \{y_i\}) = p(\{z_i\} \mid \{y_i\})\, p(\theta \mid \{z_i\}, \{y_i\}), \tag{21}$$

which would imply a different network architecture. Specifically, a classification network would need to predict mixture membership based on the (1) local and (2) global summary networks,

$$p_\alpha\left(z_i \mid \{y_j\}_{j=1}^N\right) \approx \text{softmax}\left(f_\alpha\left(h_\omega(y_i), h_\psi\left(\{h_\omega(y_j)_{j=1}^N\}\right)\right)\right) \quad \text{for } i \in \{1, \ldots, N\}, \tag{22}$$

which requires conditioning the classification network on the entire data set, as opposed to conditioning it on the parameters.

The mixture probability distribution $p(\{z_i\} \mid \{y_i\})$ by definition marginalizes out the parameter values, i.e., provides only the expectation of the probability distribution with respect to the parameter values, $p(\{z_i\} \mid \{y_i\}) = \mathbb{E}_{\theta|y}\left(p(\{z_i\} \mid \{y_i\}, \theta)\right)$. In principle, estimating the distribution of $\{z_i\}$ on its own would also be suitable for applications when parameter posteriors are not of interest, only mixture membership classification is. A notable example are Bayesian non-parametric mixtures with infinite number of components, where clustering of the data is the main focus of inference (see Miller and Harrison 2018). Other applications of mixture models typically involve more detailed analysis of the joint probability distribution of θ and $\{z_i\}$. At minimum, parameter posteriors are used to check model fit or misspecification.

To estimate the joint distribution using the alternative factorization, the parameter posterior network would have to be informed by both the data and the latent indicators. One way how to achieve this is to introduce an additional global summary network h_ξ that would learn embeddings of the concatenation of the local summary embeddings and the latent indicators,

$$p(\theta \mid \{z_i\}, \{y_i\}) \approx q_\phi\left(\theta \mid h_\xi\left(\{h_\omega(y_i), z_i\}\right)\right). \tag{23}$$

During inference, the mixture indicators would have to be sampled from a categorical distribution according to $\{z_i\} \sim p_\alpha(\{z_i\} \mid \{y_i\})$ for every sample s from the parameter posterior. Note that if posterior samples of the mixture indicators are required, then draws from the categorical distribution are necessary even for the factorization used in this article. However, in most applications the quantities of interest are the mixture membership *probabilities* rather than specific sampled realizations. Since these probabilities are provided directly by the classification network, additional sampling from the categorical distribution is not needed. Another disadvantage of this approach is the need to use two global summary networks – one that is used for classification, and one that is used for posterior inference. For these reasons, we decided to use only the factorization described in our case studies in Section 2.4.

2.6 Related work

The present article puts a lot of emphasis on estimating the joint distribution of categorical and continuous variables – in the present case, the latent mixture membership indicators, and the parameter of the mixture model. Modeling of continuous and discrete quantities was also proposed by Boelts et al. (2022), used the same principle (factorizing the joint distribution) to model continuous and discrete data via neural networks (neural likelihood estimation). In the context of posterior inference, Schröder and Macke (2023) proposed simultaneous inference over discrete model components and their parameters, essentially allowing model-averaged inference and prediction. The conceptual difference of this work is that whereas predicting model posterior requires a single classification conditioned on the whole dataset, inference for mixture models requires multivariate classification conditioned on individual observational units. The split between inferring *local* and *global* variables also makes the mixtures application a case related to multilevel and graphical models (Habermann et al. 2024; Arruda et al. 2025; Heinrich et al. 2024); in particular, the currently used factorization is conceptually similar to two-level multilevel models described by Habermann et al. (2024).

3 Case studies

We evaluate our proposed approach for amortized Bayesian mixture models on three case studies. The first two case studies present idealized synthetic examples that demonstrate the use of our approach in the case of independent and dependent mixture models, respectively. The last example shows an application on a real world data set. All applications of ABI are implemented using the `BayesFlow` software package (Radev et al. 2022) in `Python` (Rossum and Drake 2010). Code associated with this article is accessible at osf.io/7wvyk/.

3.1 Evaluation

An implementation of any model needs to be validated in order to ascertain whether conclusions made using the model are not misleading (Hermans et al. 2022). For each application, we apply a range of checks to validate the models we showcase in this article. We follow a principled Bayesian workflow (Schad et al. 2019; Gelman et al. 2020; Gabry et al. 2019).

Simulation-based calibration (SBC, Talts et al. 2018; Modrák et al. 2023) is used to validate whether the parameter posterior approximation is well calibrated with respect to the *true* posterior. In SBC, parameters are repeatedly sampled from the prior, synthetic data are generated from these

parameters, and the model is refitted to each dataset (generate posterior draws). For each parameter, we then compute the rank of the true (data-generating) parameter value among its posterior draws. When the inferred posterior distribution is well calibrated with respect to the true data-generating process, these ranks follow a uniform distribution. To visualize this, we plot the fractional ranks: the normalized ranks divided by the number of posterior samples. Specifically, we show the difference between the expected ECDF of a uniform distribution and the observed ECDF of the fractional rank statistic. Ideally, the ECDF difference fluctuates close to zero. A failed SBC check indicates issues with computational validity of the parameter posterior approximation, which could be due to not sufficient training, or a lack of network expressiveness. See Modrák et al. (2023); Talts et al. (2018) for more details about SBC. Additionally, parameter recovery, posterior z-score, and posterior contraction are used to judge whether the data in combination with the statistical model lead to meaningful inferences. Poor results can indicate poor posterior calibration, but can also mean problems with parameter identifiability, or a lack of information provided by the data.

To detect model misspecification, we take several approaches. First, we adopt a method from the neural network literature that tests whether the observed data deviates from the typical data encountered during training (i.e., *simulation gaps*; Schmitt et al. 2024a). Specifically, we use the Maximum Mean Discrepancy (MMD) as a discrepancy measure. During training, the global summary network is regularized with an MMD loss to ensure that the learned summary statistics approximately follow a Gaussian distribution. During inference, we compute the MMD between the empirical data and data simulated from the generative model, and compare it against a reference (null) distribution derived from the simulation model. This enables a formal test of whether the observed data are consistent with the data-generating process assumed during training (Schmitt et al. 2024a). From the Bayesian statistics perspective, we also perform *posterior predictive checks* (e.g., Gelman and Meng, X.- L., Stern, H. 1996) to assess whether replicated data drawn from the posterior predictive distribution resemble the observed data. Since the first two case studies are demonstrations using synthetic data, the statistical models are by definition well specified. Therefore, we conduct model misspecification checks only in the final case study.

As a last step, we investigate the accuracy of the neural inference by comparing it to results from state-of-the-art MCMC. To this end, models presented in our case studies were evaluated against the results using the probabilistic programming language `Stan` (Carpenter et al. 2017). We compare posterior samples from `BayesFlow` and `Stan` to contrast our implementation of ABI to MCMC. Additionally, we also use the classifier two-sample test (C2ST; Lopez-Paz and Oquab 2016) as a quantitative measure of distribution similarity. In our implementation of C2ST, ABI and MCMC posterior samples are labeled as separate classes and used to train a neural network classifier (MLP). The classifier's predictive performance is estimated via 5-fold cross-validation, and using classification accuracy as the evaluation metric. The resulting mean accuracy constitutes the C2ST score; values close to 0.5 indicate that the two sample sets are indistinguishable, whereas higher values suggest greater divergence between the underlying distributions.

3.2 Parameter constraints

Some parameters in the current models are naturally constraint – for example, mixture proportions are bounded between zero and one and must sum to one. Across all experiments and applications, we ensured that all of the model parameters estimated by the amortized posterior estimator are trained on the unconstrained real space. Because the networks only see the unconstrained parameters during training, the parameter posterior network is subsequently making inferences on the unconstrained parameter space. Similarly, the classification networks are also conditioned on parameters in the unconstrained space. Where appropriate, we transform the parameters from the unconstrained space into the constrained space for easier interpretation of the results. Here we explain the parameter transformations used across the case studies.

Unit simplex

Unit simplex parameters π are constrained such that $0 \leq \pi_k \leq 1$ for $k \in \{1, \ldots, K\}$ and $\sum_k^K \pi_k = 1$. Parameters constrained on the unit simplex in mixture models typically come in form of mixture proportions. Further, in hidden Markov models, each row of the latent state transition matrix is a unit simplex. To explain our notation,

$$\pi \sim \text{Dirichlet}(2, 2, 2) \tag{24}$$

indicates that the elements of the vector π are generated from a Dirichlet distribution with $K = 3$ parameters. By construction, only $K - 1$ parameters are needed to reconstruct the entire vector, since all values need to sum to one.

During training, the parameters are transformed into an unconstrained space $\theta \in \mathbb{R}^{K-1}$, defined for $1 \leq k < K$,

$$\theta_k = \text{logit}^{-1}\left(\pi_k + \log\left(\frac{1}{K-k}\right)\right). \tag{25}$$

During inference, the posterior parameter network outputs the posterior distribution on the unconstrained space. To convert the parameters back to the constrained space, the following transformation is applied,

$$\pi_k = \begin{cases} \left(1 - \sum_j^{k-1} \theta_j\right)\theta_k & \text{if } 1 \leq k < K \\ 1 - \sum_{j=1}^{K-1} \pi_j & \text{if } k = K. \end{cases} \tag{26}$$

Ordered vector

A mixture model with the same distribution under each mixture component is identifiable up to the permutation of the mixture labels. To prevent label switching (see Jasra et al. 2005), one simple solution is to impose order constraints on parameters. While this naive approach can suffer from several drawbacks if used in isolation to solve the label switching problem (e.g., Celeux et al. 2000; Marin et al. 2005), we use order constraints in combination with (1) weakly informed priors (all case studies) and (2) differences between mixture components likelihoods (case study 3) to ensure model identifiability. To explain the notation,

$$(\mu_1, \mu_2, \mu_3) \sim \text{Normal}\Big((-2, 0, 2), \mathbb{I}\Big)_{\mu_1 < \mu_2 < \mu_3} \tag{27}$$

indicates an order constraint of the mean parameters in a $K = 3$ component mixture such that $\mu_1 < \mu_2 < \mu_3$. During sampling from the generative model, the order constraints are achieved by rejection sampling; first, we draw from the normal distribution, and if the draw does not satisfy the constraint the draw is repeated. In principle, rejection sampling is not an efficient way to draw from the constrained distribution. However, in our cases, the prior components are already relatively well separated, leading to low rejection rates. As a result, rejection sampling did not significantly slow down sampling from the Bayesian generative model.

During training, parameters are transformed to an unconstrained space, where

$$\theta_k = \begin{cases} \mu_1 & \text{if } k = 1 \\ \log(\mu_k - \mu_{k-1}) & \text{if } 1 \leq k < K. \end{cases} \tag{28}$$

During inference, the parameter posterior network returns the posterior distribution on the unconstrained space. To convert the parameters back into the constrained space, the following transformation is applied,

$$\mu_k = \begin{cases} \theta_1 & \text{if } k = 1 \\ \mu_{k-1} + \exp(\theta_k) & \text{if } 1 < k \leq K. \end{cases} \tag{29}$$

3.3 Case Study 1: Gaussian mixture model

The first experiment used as an example is a simple independent mixture model with three mixture components. The generative model is as follows:

$$\begin{aligned} N &\sim \text{Uniform}(150, 250) \\ P &\sim \text{Uniform}(2, 4) \\ (\mu_1, \mu_2, \mu_3) &\sim \text{Normal}\Big((-2, 0, 2), \mathbb{I}\Big)_{\mu_1 < \mu_2 < \mu_3} \\ \pi &\sim \text{Dirichlet}(2, 2, 2) \\ z_i &\sim \text{Categorical}(\pi) \\ &\text{for } i \in \{1, \ldots, N\} \\ y_{ij} &\sim \text{Normal}(\mu_{z_i}, 1) \\ &\text{for } i \in \{1, \ldots, N\};\ j \in \{1, \ldots, P\} \end{aligned} \tag{30}$$

The context variables N and P (number of observational units and number of observations per unit, respectively) varied during training so that the networks learn to amortize over different dataset sizes.

The Deep Set architecture (Zaheer et al. 2017) produces embedding for exchangeable data, such that the summary embedding is identical for any permutation of the data points. We used a Deep Set network as the local summary network h_ω to obtain an embedding for each y_i individually, producing a set of embeddings $\{h_\omega(y_i)\}$. This approach compresses the data at the subject level, ensuring that the representation of each observational unit has a consistent dimensionality, regardless of the number of observations per unit p. In this case study, the local summary could be easily handcrafted (i.e., by computing the arithmetic mean). To showcase the capabilities of the framework in general, we still use a neural network, albeit it need not be complex: it contains two dense layers in the inner and outer function of the Deep Set, as well as two equivariant layers. The output size of the summary network was set to two: While in principle the summary network could capture all information with a single output (the minimal number of sufficient statistics is 1 in this case), giving the network more flexibility allowed it to learn the representation more easily.

The global summary network distills a fixed length embedding $h_\psi(\{h_\omega(y_i)\})$ from the set of individual local summaries, $\{h_\omega(y_i)\}$). The global embedding is used for parameter posterior inference, and so has a more demanding task than the local inference network; if only because it needs to extract summary statistics that relate to the five free parameters in the model. The global network is also implemented as a Deep Set (Zaheer et al. 2017), but we let the network possess more expressive power; the number of dense layers in the inner and outer functions were doubled (i.e., four), while the number of equivariant layers was set to three.

The embedding from the global summary network is concatenated with the context variables n and p and passed to an invertible spline coupling network with 14 layers, implemented according to Radev et al. (2022). This network is used for parameter posterior inference. As a classification

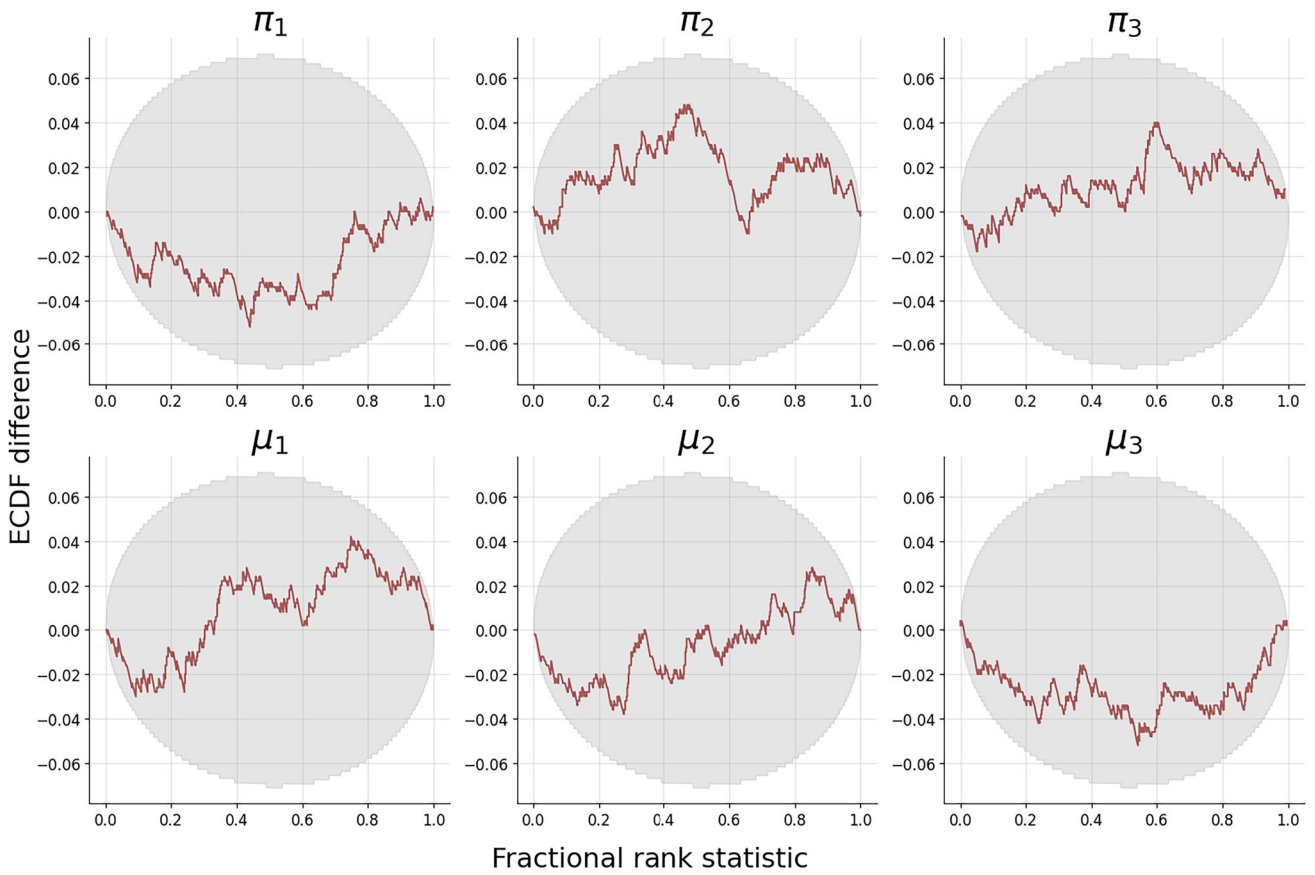


Fig. 4 *Case Study 1: Gaussian mixture model.* Simulation-based calibration results displayed as difference between the empirical and expected cumulative distribution function of the fractional rank statistic. The shaded area corresponds to the 95% Confidence bands

network, a multilayer perceptron model composed of 12 fully connected layers with the ReLU activation function was used.

All networks were trained jointly in an online training regime, for a total duration of 100 epochs, 2000 iterations each, with each iteration made of 128 sampled data sets from the Bayesian generative model. Figure 4 shows the results of the simulation-based calibration (Talts et al. 2018) of the approximator of the parameter posteriors when the context variables were fixed at $N = 200$, $P = 3$. The results indicate good calibration of the estimated posteriors. Parameter recovery plots (Figure 5) did not reveal issues with posterior approximation. Obtaining 1000 samples for 500 data sets with `BayesFlow`, without GPU acceleration, took about 12 sec in total. Obtaining 500 warmup and 1000 sampling iterations from a single chain in `Stan` (without parallelization) takes about 3 sec on the same hardware for a single data set – fitting the model on 500 data sets would have therefore taken about a day. Inference with `BayesFlow` is therefore about 2 orders of magnitudes faster than with `Stan` for this model.

Next, we show how the amortized inference compares to results obtained using MCMC with `Stan` (Carpenter et al. 2017). We simulated a single data set from the generative model, and fitted the mixture model on the simulated data with ABI and with MCMC. The true parameter values in this particular example are $\pi = (0.3, 0.5, 0.2)$, $\mu = (-1.8, -0.5, 1.2)$ and the context variables were set to $N = 200$, $P = 3$.

Figure 6 shows the joint posterior distribution of the parameters as estimated by ABI and MCMC, with 6000 samples using either method. Visual inspection indicates that the distributions are almost identical. The C2ST score is 0.53, indicating that distinguishing between the ABI and MCMC samples is a difficult task for a neural classifier. These results suggest that both methods sample from nearly the same distribution.

Figure 7 shows the results of the classification network when applied to a single data set drawn from the Bayesian generative model. The results indicate that the neural estimation of the classification probabilities are similar to the results calculated using the analytic likelihoods based on the parameter posteriors obtained with MCMC. The results suggest that the neural classifier is able to represent the mixture probabilities in a faithful manner. The 95% CI intervals also overlap for both methods, suggesting that the neural classifier is able to accurately propagate the uncertainty of the

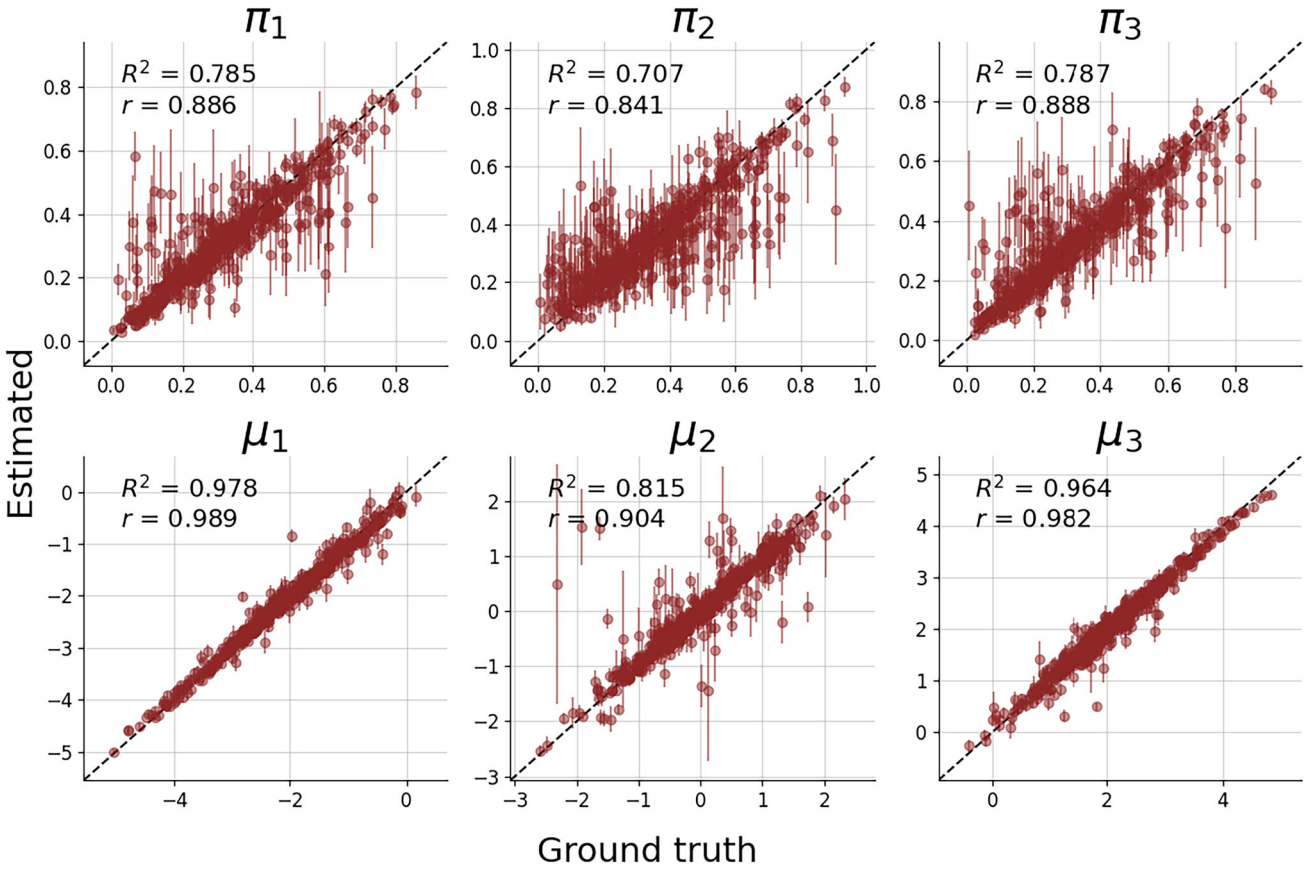


Fig. 5 Case Study 1: Gaussian mixture model. Parameter recovery shown as scatter plot between the true data generating parameter values and the estimated parameter values. The point estimates are the median, whereas the lines depict the 95% central credible interval

parameter values into the uncertainty of the classification probabilities. However, the neural estimates are visibly more "jittery", i.e., rather than producing smooth changes in the input to small change in the input, the output varies slightly more chaotically than expected. This might be caused by an overly complex classification network (12 fully connected layers) with a ReLU activation (which is sometimes prone to produce non-smooth output).

3.4 Case Study 2: Gaussian hidden Markov model

The second experiment builds on the first one, but introduces some changes. First, instead of the latent mixture indicators being sampled independently, they now follow a first-order Markov chain. The model can be therefore viewed as a hidden Markov model for evenly spaced time-series (Frühwirth-Schnatter 2006; Visser and Speekenbrink 2022; Zucchini et al. 2016). Second, each time point can consist of variable number of observations, instead of that being fixed for all time points. Lastly, the model is composed of only two mixture components (hidden states). The model can be written down as follows,

$$
\begin{aligned}
P_i &\sim \text{Uniform}(2, 5) && \text{for } i \in \{1, \ldots, N\} \\
\alpha_k &\sim \text{Dirichlet}(2, 2) && \text{for } k \in \{1, 2\} \\
(\mu_1, \mu_2) &\sim \text{Normal}\Big((-1.5, 1.5), \mathbb{I}\Big)_{\mu_1 < \mu_2} \\
z_1 &\sim \text{Categorical}(0.5, 0.5) \\
z_i &\sim \text{Categorical}(\alpha_{z_{i-1}}) && \text{for } i \in \{2, \ldots, N\} \\
y_{ij} &\sim \text{Normal}(\mu_{z_i}, 1) && \text{for } i \in \{1, \ldots, N\};\ j \in \{1, \ldots, P_i\},
\end{aligned}
\tag{31}
$$

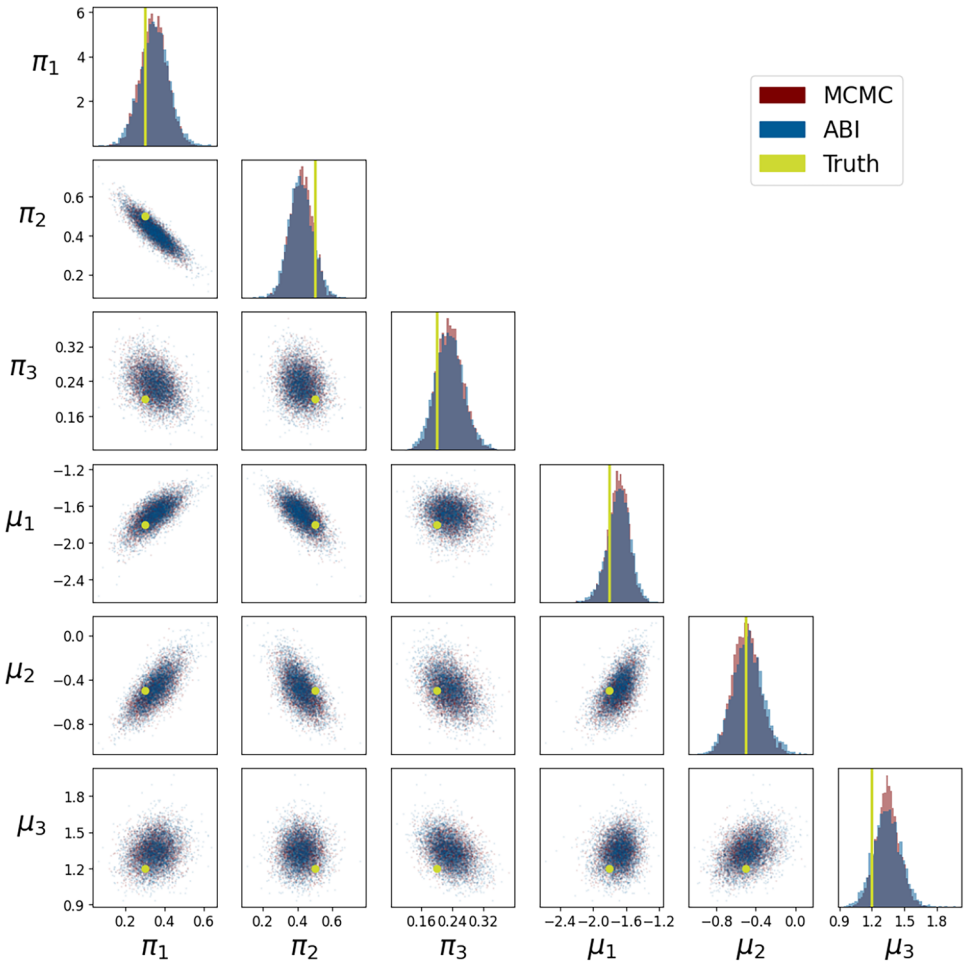


Fig. 6 *Case Study 1: Gaussian mixture model.* Joint posterior distribution of the parameters obtained with ABI and MCMC. Example using synthetic data. The diagonals show the marginal parameter posteriors, whereas off-diagonal elements show the pairwise scatter plots to display dependencies between parameters

where $N = 100$ is the number of time points (observational units) and P_i is the number of observations per time point. The 2×2 transition matrix with elements α_{ij} gives the probability of transitioning from state i to state j. Since each row of the transition matrix sums up to one, we will only show the results for the diagonal elements of the matrix α_{11} and α_{22}, corresponding to the probability of staying at the current state 1 or 2, respectively.

For the same reasons as in the first case study, Deep Set architecture (Zaheer et al. 2017) was used as the local summary network h_ω. However, in order to extract information from the temporal dependencies between the data points, an LSTM network (Gers et al. 1999) was used as the global summary network.

The classification network f_α is implemented as an LSTM layer with 32 units, followed by series of fully connected dense layers with ReLU activation. The LSTM layer allows the network to take into account other observational units, whereas the dense layers increase the expressiveness of the network.

All networks were trained jointly in an online training regime, for a total duration of 50 epochs, 500 iterations each, with each iteration made of 512 sampled data sets from the Bayesian generative model.

For validation of the posterior approximator, we simulated 1000 data sets using the Bayesian generative model, and fitted them with the amortized posterior approximator by generating 1000 posterior samples for each data set. Obtaining 1000 samples for 1000 data sets with `BayesFlow`, even without GPU acceleration, took about 12 sec. Obtaining 500 warmup and 1000 sampling iterations from a single chain in `Stan` (without parallelization) takes about 2 sec on the same hardware *for a single data set*. Inference with ABI is therefore about 2 orders of magnitudes faster than with MCMC for this model. As shown in Figure 8, the SBC revealed no clear patterns of miscalibration of the posterior approximator. Figure 9 does not reveal issues with estimating the mean parameters μ_1 and μ_2. Recovery of the transition probabilities (α_{11} and α_{22}) is slightly worse. A closer inspection reveals that parameter recovery is challenging in regions of the parameter space where the latent states are insufficiently separated ($\mu_2 - \mu_1 \geq 2/3$), as illustrated in the middle row of Figure 9. In contrast, when the states are well separated, recovery of the transition probabilities improves (bottom row

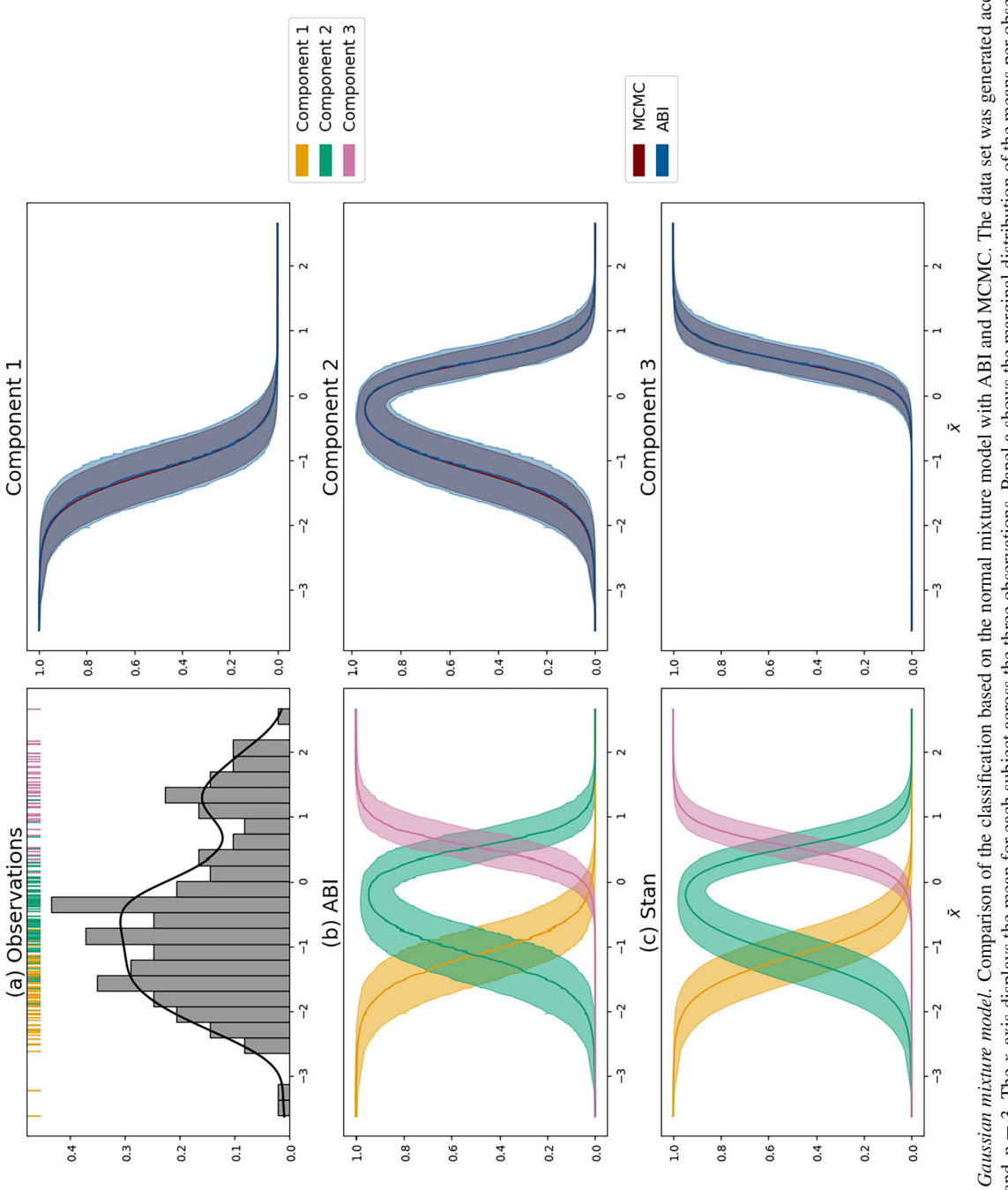


Fig. 7 *Case Study 1: Gaussian mixture model.* Comparison of the classification based on the normal mixture model with ABI and MCMC. The data set was generated according to the model in Eq. 30 with $n = 200$ and $p = 3$. The x-axis displays the mean for each subject across the three observations. Panel a shows the marginal distribution of the means per observational unit, with rug marks on top showing the individual points coming from three different mixture components. Panel b shows probability of membership in each of the respective mixture components, given the mean on the x-axis, according to ABI. Panel c shows the same but using MCMC. Panels on the right side show the same components probabilities, ABI and MCMC estimates overlayed on top of each other to allow direct comparison. The lines show the median of the classification distribution, the confidence bands display the 95% central credible interval

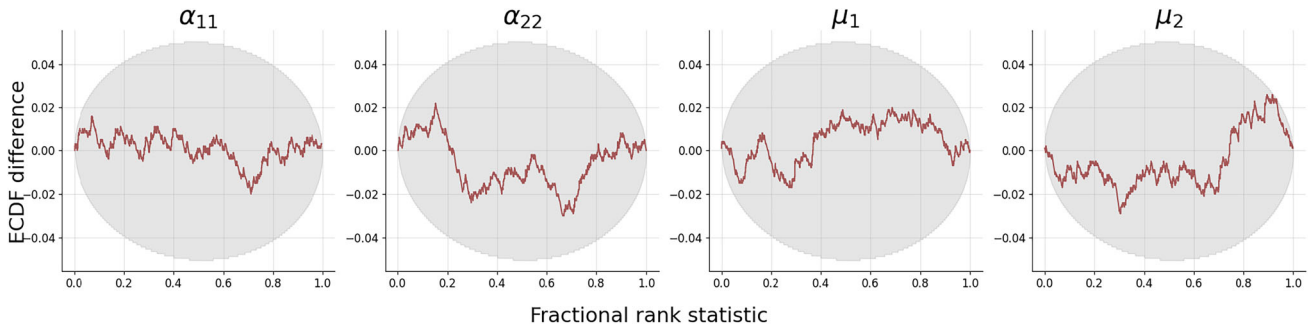


Fig. 8 *Case Study 2: Gaussian hidden Markov model.* Simulation-based calibration results displayed as difference between the empirical and expected cumulative distribution function of the fractional rank statistic. The shaded area corresponds to the 95% Confidence bands

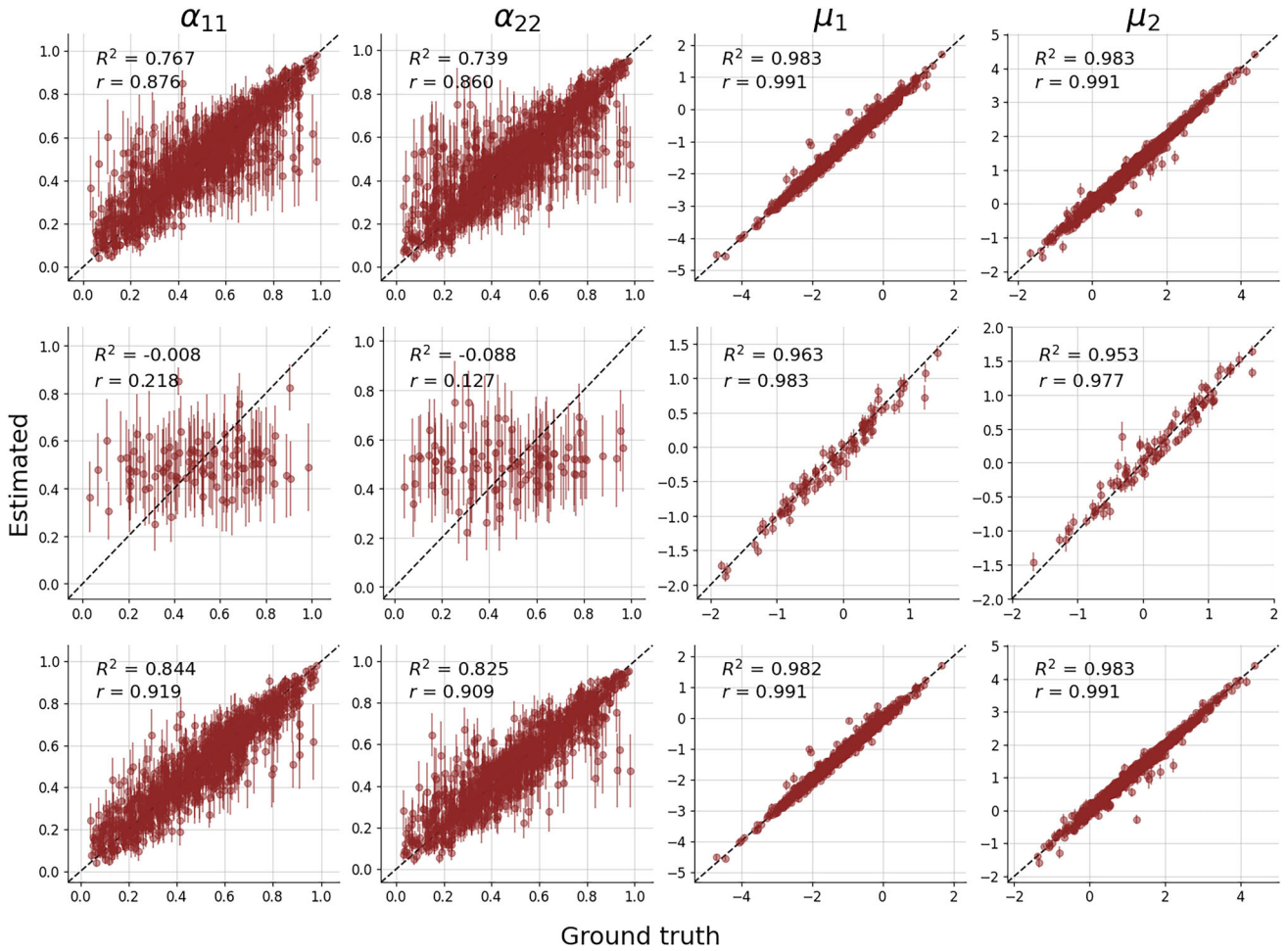


Fig. 9 *Case Study 2: Gaussian hidden Markov model.* Parameter recovery shown as scatter plot between the true data generating parameter values and the estimated parameter values. The point estimates are the median, whereas the lines depict the 95% central credible interval. Top row: Recovery across the entire parameter space. Middle row: Recovery for a subset of the parameter space where $\mu_2 - \mu_1 < 2/3$. Bottom row: Recovery for a subset of the parameter space where $\mu_2 - \mu_1 \geq 2/3$

of Figure 9). This pattern is characteristic of mixture models: poor separation between components increases uncertainty in the mixture indicators, which in turn reduces the precision of estimates for mixture or transition probabilities.

To showcase the application of the amortized HMM, we show the results applied to one synthetic example generated from the Bayesian generative model. Figure 10 shows the posterior distribution of the parameters. Both ABI and

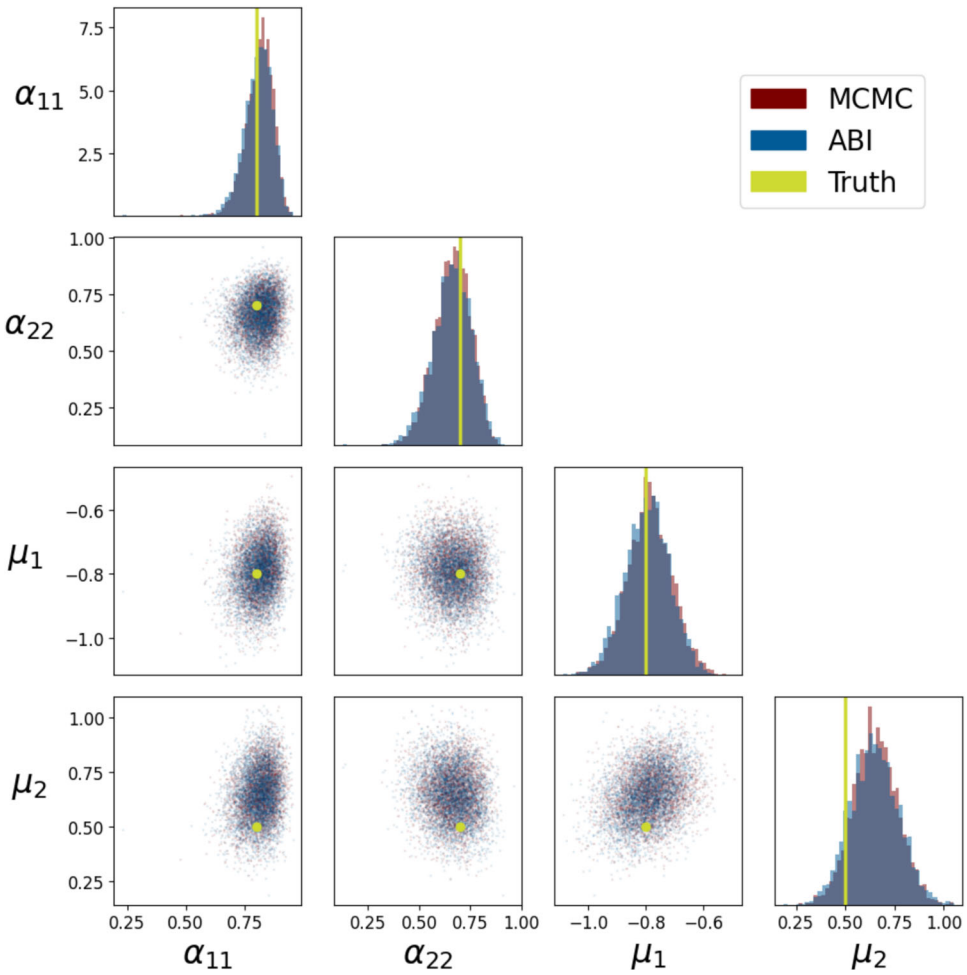


Fig. 10 *Case Study 2: Gaussian hidden Markov model.* Comparison between parameter posterior approximations using ABI and MCMC. Example using synthetic data. The diagonals show the marginal parameter posteriors, whereas off-diagonal elements show the pairwise scatter plots to display dependencies between parameters. The two methods return nearly identical results, yielding almost perfectly overlapping distributions

MCMC estimates are very similar. The C2ST score is 0.56, indicating that distinguishing between the ABI and MCMC samples is a difficult task for a neural classifier. These results suggest that both methods sample from nearly the same distribution.

ABI and MCMC also agree on the classification probabilities based on forward filtering, backward filtering, and smoothing, as shown in Figure 11, suggesting that the classification network is well calibrated as well, including calibration of the classification uncertainty as demonstrated by overlapping confidence intervals.

3.5 Case Study 3: Latent switches in cognitive processing

The current application is based on the empirical study reported by Dutilh and Wagenmakesr, Visser, van der Maas, (2011) that studied human decision making in speeded decision tasks. One prediction is that under different incentive conditions (i.e., varying reward for speed versus accuracy), participants are unable to control the speed-decision tradeoff on a continuum, but rather switch between distinct modes of behavior – under one mode, participants tend to guess randomly in order to provide fast responses, and under another mode, provide slower responses for the sake of improving their accuracy. These modes can be represented as latent states in a mixture model. Since the incentives in the experiment between trials are adjusted continually, it is expected that there will be some temporal dependencies between the states; here, we will model these dependencies with a Hidden Markov model.

The response behavior is modeled using evidence accumulation models (EAMs), a widely used class of cognitive process models of decision-making (e.g., Ratcliff 1978; Ratcliff and McKoon 2008; Bogacz et al. 2006; Ratcliff et al. 2016; Evans et al. 2020). EAMs assume that a decision is made by stochastically accumulating evidence over time toward a boundary that represents the amount of evidence required to trigger a response. The drift rate (ν) governs the mean rate of accumulation, while the boundary separation (α) reflects response caution: larger boundaries lead to slower but more accurate responses. A non-decision time (τ) accounts for residual processes such as perceptual encoding and motor execution. Noise in evidence accumulation can

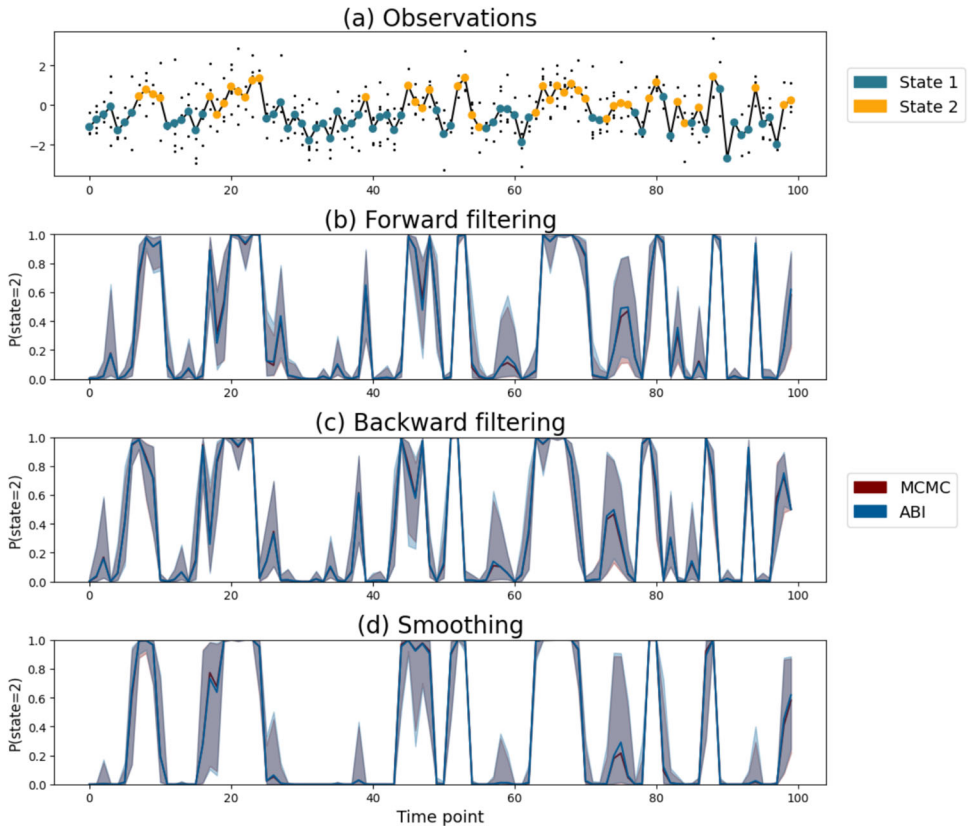


Fig. 11 *Case Study 2: Gaussian hidden Markov model.* Comparison between ABI and MCMC in the predicted classification probabilities using forward (panel b) and backward (panel c) filtering, and smoothing (panel c), using the Gaussian hidden Markov model. In panel a, individual observations are shown as black dots. Large colored dots depict the mean. In the classification plots, the solid lines show the posterior median, and the confidence bands display the 99% central credible interval. Example using synthetic data. The two methods return nearly identical results, yielding almost perfectly overlapping classification probabilities

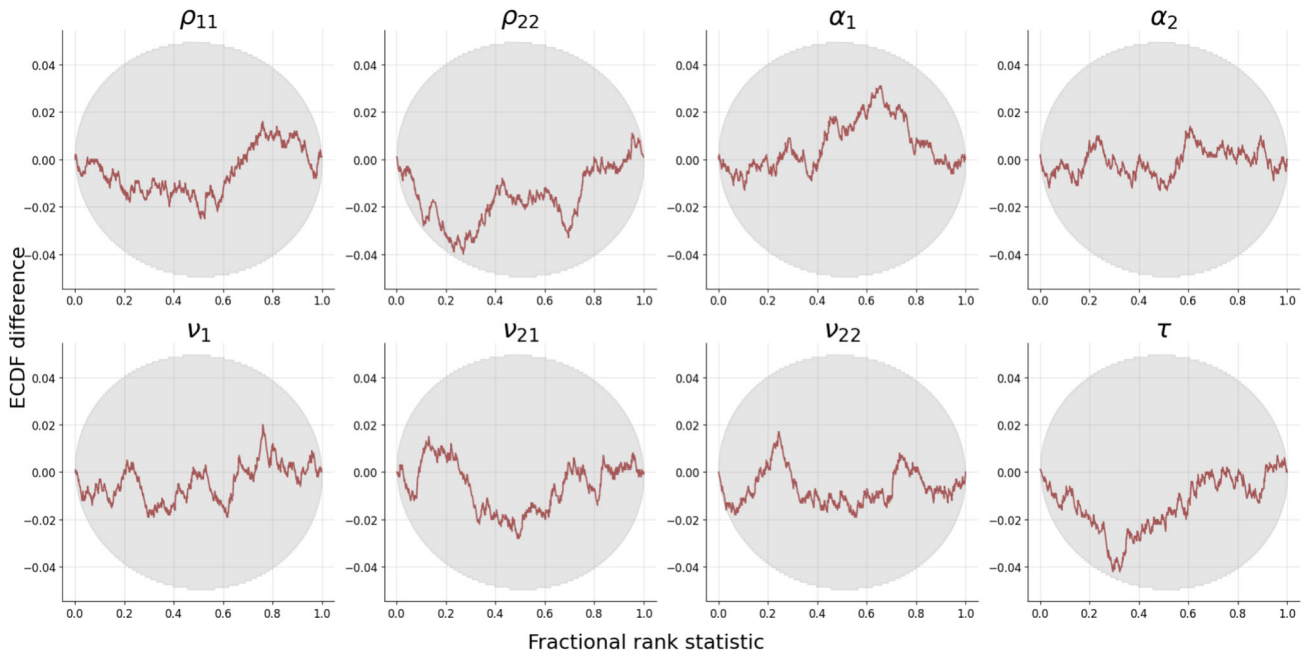


Fig. 12 *Case Study 3: Latent switches in cognitive processing.* Simulation-based calibration results displayed as difference between the empirical and expected cumulative distribution function of the fractional rank statistic. The shaded area corresponds to the 95% Confidence bands

arise both within and between trials, reflecting sensory noise or fluctuations in attention and task difficulty, etc (Tillman et al. 2020; Bogacz et al. 2006; Evans et al. 2020).

Under the *guessing state* ($z_i = 1$), response times arise as a result of a simple evidence accumulation process as a Wiener diffusion process (see Karatzas and Shreve 2014) with a drift ν_1. Once evidence reaches the threshold α_1, a response is

triggered at random with a non-decision time delay τ. This response time process is known as simple response times (Luce 1991); using the Wiener diffusion process implies that the response times follow the shifted Wald distribution (i.e., inverse Gaussian; Anders et al. 2016; Chhikara and Folks 2019).

Under the *controlled state* ($z_i = 2$), responses are generated from a four-parameter Racing Diffusion Model (RDM, Tillman et al. 2020). The RDM assumes that two parallel accumulators (one representing the correct response and one the incorrect response) race towards a decision boundary α_{22}. Each accumulator follows a Wiener diffusion process, characterized by its own drift rate (ν_{22} for the correct accumulator and ν_{21} for the incorrect accumulator). A decision is made when the first accumulator reaches the boundary α_2, determining both the response outcome y_i (1 if the "correct" accumulator wins, 0 otherwise) and the decision time t_i. A non-decision time τ is then added to account for sensory encoding and motor execution, such that the observed response time is $rt_i = t_i + \tau$.

The RDM captures characteristic phenomena of human decision-making (Tillman et al. 2020), including the continuous speed–accuracy trade-off: increasing the boundary α_2 or the drift rate ν_{22} leads to slower but more accurate responses. Despite its cognitive richness, the four-parameter RDM remains relatively parsimonious, making it a suitable observation model within a larger hierarchical framework (in this case, a single state of the HMM; Kucharský, Tran, Veldkamp, Raijmakers, Visser, 2021).

The full model can be summarized as follows,

$$
\begin{aligned}
\rho_k &\sim \text{Dirichlet}(2, 2) && \text{for } k \in \{1, 2\} \\
z_1 &\sim \text{Categorical}(0.5, 0.5) \\
z_i &\sim \text{Categorical}(\rho_{z_{i-1}}) && \text{for } i \in \{2, \ldots, N\} \\
\alpha_1 &\sim \text{Normal}(0.5, 0.3)_{T(0,\infty)} \\
\nu_1 &\sim \text{Normal}(5.5, 1.0)_{T(0,\infty)} \\
(\alpha_2 - \alpha_1) &\sim \text{Normal}(1.5, 0.5)_{T(0,\infty)} \\
\nu_{21} &\sim \text{Normal}(2.5, 0.5)_{T(0,\infty)} \\
(\nu_{22} - \nu_{21}) &\sim \text{Normal}(2.5, 1.0)_{T(0,\infty)} \\
\tau &\sim \text{Exponential}(5.0) \\
(rt_i, y_i) &\sim \begin{cases} \big(\text{Wald}(\alpha_1, \nu_1) + \tau, \text{Bernoulli}(0.5)\big) & \text{if } z_i = 1 \\ \text{RDM}(\alpha_2, \nu_{21}, \nu_{22}, \tau) & \text{if } z_i = 2 \end{cases} && \text{for } i \in \{1, \ldots, N\},
\end{aligned}
\tag{32}
$$

where $N = 400$ is the number of trials in the experiment (observational units). The observable variables are the response times (in seconds) rt_i and the choices (binary) y_i on experimental trial i. The 2×2 transition matrix with elements ρ_{ij} gives the probability of transitioning from the latent state i to j. Since each row of the transition matrix sums up to one, we will only show the results for the diagonal elements of the matrix α_{11} and α_{22}, the probability of staying in the guessing and controlled state, respectively. The response model comprises of six parameters: the non-decision time τ, the decision boundary α_1 and the drift rate ν_1, respectively, under the guessing state, the decision boundary α_2, the drift rate for the incorrect response ν_{21} and the correct response ν_{22} under the controlled state. The parameter priors were selected through prior predictive simulations such that model generates realistic patterns of observed data under each state Kucharský, Tran, Veldkamp, Raijmakers, Visser, (2021).

Since the observational units are always comprised only of one observation of response time rt_i and one observation of choice y_i, using a local summary network would be redundant. Thus, raw data is passed directly to the global summary network for posterior inference, as well as the classification network for the classification inference. The global summary network consists of four convolutional layers followed by a bidirectional LSTM with 256 units. The convolutional layers serve to extract and smooth local temporal patterns in the sequence of responses, while the LSTM captures short- and medium-range dependencies that reflect latent state persistence and switching in the underlying HMM. This combination allows the inference network to construct summary representations that are sensitive to temporal structure in the observed data. The final output layer (32 units) is regularized using the MMD loss to ensure approximately normal summary statistics (Schmitt et al. 2024a). This will allow us later to compute diagnostics of model misspecification when the model is applied to empirical data.

The classification network f_α is implemented as an LSTM layer with 32 units, followed by a series of fully connected layers with ReLU activation. The LSTM layer allows the network to take the temporal dependencies in the data, while

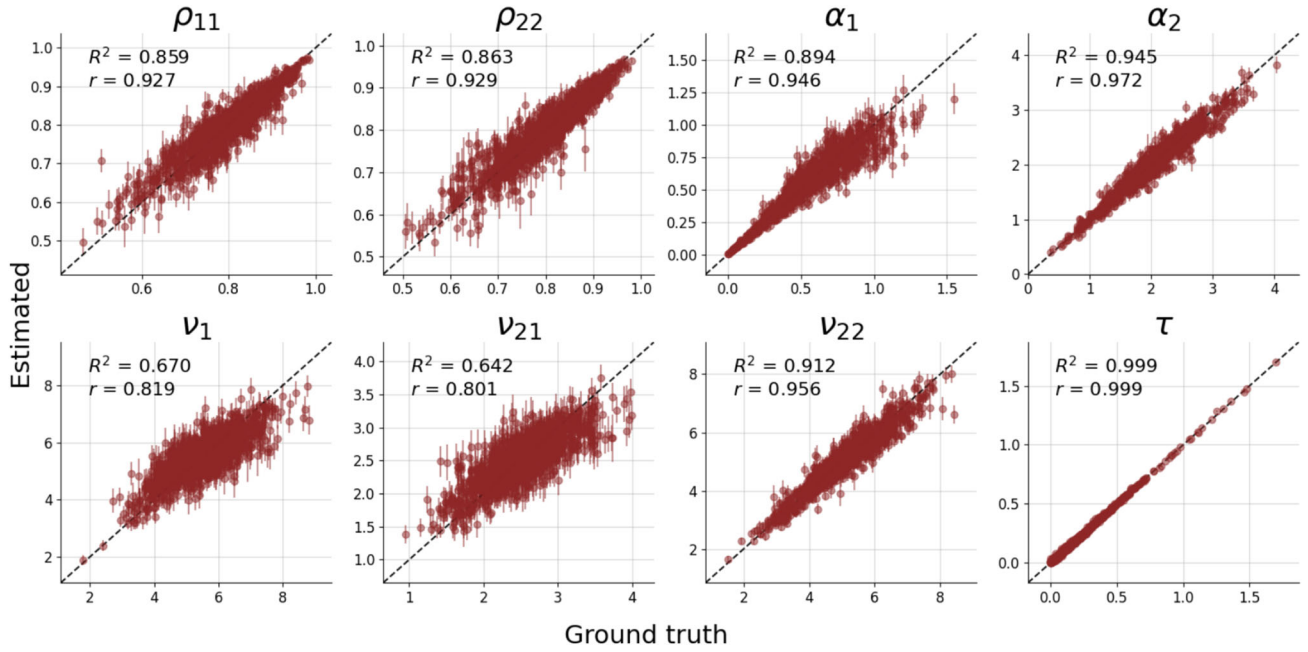


Fig. 13 Case Study 3: Latent switches in cognitive processing. Parameter recovery shown as scatter plot between the true data generating parameter values and the estimated parameter values. The point estimates are the median, whereas the lines depict the 95% central credible interval

the dense layers further increase the expressiveness of the network. The classification network was trained both in the forward and backward regime, so that it can be used for both *filtering* and *smoothing*.

All networks were trained jointly in an online training regime, for a total duration of 100 epochs, 1000 iterations each epoch, with each iteration made of 256 sampled data sets from the Bayesian generative model.

For validating the posterior approximator, we simulated 1000 data sets from the Bayesian generative model, and fitted them with the amortized posterior approximator, obtaining 1000 posterior samples for each data set. As shown in Figure 12, SBC revealed no miscalibration of the posterior approximator. Figure 13 shows that parameters can also be recovered. Overall the results of the validation simulations suggested that the amortized posterior works well.

To evaluate the model's performance on real data, the model was fitted to data from 11 participants in an experiment reported by Dutilh and Wagenmakesr, Visser, van der Maas, (2011). For reference, we used `BayesFlow` and `Stan` to fit all data sets with ABI and MCMC, respectively. Obtaining 4000 samples for all 11 data sets took about 40 sec using `BayesFlow`, and about 15 minutes with `Stan`.

To check for model misspecification, we computed the MMD of the summary statistics returned by the global summary network, and compared it to the distribution of MMD statistics computed on prior predictive data sets. As shown in Table 1, for seven out of the eleven data sets, the computed MMD did not exceed critical values that would identify model misspecification. The C2ST score (Lopez-Paz and Oquab 2016) generally agreed with MMD; datasets associated with high MMD are also ones where classifying posterior samples from MCMC and ABI is relatively easier.

Table 1 MMD test and Pareto $\hat{k}$ diagnostic. Highlighted values indicate failed diagnostics. Failed MMD test ($\alpha = 0.1$) indicates a simulation gap and suggests that the amortized posterior may have a problem generalizing to the current data set and the posterior is therefore not trustworthy. Pareto $\hat{k} > 0.7$ usually indicates that Pareto smoothed importance sampling will not be effective in correcting the amortized posteriors

	MMD test			
Data set	Statistic	p-value	Pareto $\hat{k}$	C2ST
A	3.84	**.025**	**3.81**	0.99
B	3.75	**.084**	0.49	0.62
C	3.52	.664	0.46	0.56
D	3.68	.168	0.41	0.71
E	3.67	.192	0.53	0.65
F	3.44	.895	0.59	0.58
G	3.77	**.072**	**1.61**	0.98
H	3.67	.197	0.67	0.55
I	3.63	.282	0.32	0.55
J	3.72	.453	0.66	0.56
K	3.78	**.060**	**0.76**	0.82

An example of a posterior associated with a typical value of MMD is shown in Figure 14. The posterior approximations of ABI and MCMC almost perfectly overlap, both in terms of their marginal distributions, as well as the join distribution

as inspected by pairwise scatter plots. This suggests that the ABI posterior estimate is able to correctly capture the values of the parameters as well as their correlations. The low value of MMD correctly identified that the amortized posterior is indeed well calibrated for this data set, and the estimate is therefore trustworthy.

For the other four data sets, the parameter posteriors approximated by ABI may not be trustworthy. For example, as shown for participant B in Figure 15, it appears that ABI slightly underestimates parameters α_1 and ν_1.

Following the amortized Bayesian workflow (Schmitt et al. 2024b), it is possible to correct the approximated posteriors using Pareto smoothed importance sampling (PSIS, Vehtari et al. 2024, 2017). For each posterior sample obtained from the neural approximator, we obtain the log posterior density using the neural approximator, as well as the (unnormalized) analytic log posterior density; since we have already implemented this model in `Stan`, we used the `Stan` model to compute the analytic density. The log importance weights are calculated as the difference between the log neural density and log analytic density. The Pareto smoothed weights and k-statistic were computed using the `Arviz` package (Kumar et al. 2019).

Table 1 also shows the Pareto $\hat{k}$ diagnostic. For the data set B, the importance weights can be used for correcting the posterior distribution obtained with ABI.

However, for some data sets (mainly, A and G), the ABI posteriors are too far from the true posterior to be corrected by importance sampling. For example, for data set A, the MCMC posterior lies on the tails of the prior for the parameters α_1 and ν_1 (Figure 16). Subsequently, ABI poorly generalizes and substantially underestimates the parameters. This poor generalization is captured by the high MMD value and its p-value. Furthermore, the ABI posterior is too far from the true posterior, rendering importance sampling inefficient and unreliable – which is captured by the high Pareto $\hat{k}$ diagnostic in Table 1. This indicates that precise posterior inference for the data sets A and G is not available using the current networks. Data set K is a borderline case with marginally high $\hat{k} = 0.76$. For this data set, importance sampling will likely still work well given enough posterior samples.

Even in the case where the parameter posterior is not estimated reliably using neural networks, the two states are separated well enough that even the biased parameter posteriors do not have a big influence on the resulting classification; as shown in Figure 17.

4 Discussion

In this paper, we developed and evaluated a framework for amortized Bayesian mixture models based on deep learning. By decomposing mixture models into parameter posterior distributions and mixture membership distributions, we can represent these distributions through corresponding neural architectures. These neural networks can be trained simultaneously on the same training data generated from the Bayesian model. This allows flexible amortization over different design factors, such as the number of observational units. Once trained, the neural networks provide reliable estimates in a fraction of the time required by traditional methods such as MCMC.

We evaluated the proposed framework through three case studies, demonstrating its applicability. The computational faithfulness of the neural networks was rigorously assessed through simulations. We also extensively compared the accuracy of the neural network outputs against state-of-the-art MCMC results obtained using `Stan` (Carpenter et al. 2017). The first two case studies show a proof of concept on independent and dependent mixture models, though with limited applicability considering their simplistic nature. The third case study brings forth a more realistic scenario of using the framework in empirical context.

The methodology is implemented with the Python library `BayesFlow` (Radev et al. 2022), and is publicly accessible online at osf.io/7wvyk/.

Limitations & Future directions

Although the scope of the statistical models showcased in this article is relatively limited, we believe it demonstrates that the proposed framework has much broader potential. General benefits of ABI, speed during inference, fitting models with intractable likelihoods, might not be the only appealing characteristics in the context of mixture models. For example, some MCMC samplers might be prone to get stuck in isolated parameter regions for specific models (Stephens 1997; Swanson 2024; Celeux et al. 2000; Diebolt and Robert 1994; Marin et al. 2005). ABI might be a promising alternative to MCMC in such use cases.

Due to our aim to validate our results against MCMC, the parametrizations used in this article are inspired by standard practices in probabilistic programming languages such as `Stan` (Carpenter et al. 2017). However, translating these directly for ABI is not always straightforward. Priors convenient in `Stan` in particular or for MCMC in general may have significant downsides for ABI. Extremely flat priors, for instance, are often used for MCMC, but may hinder ABI by training the networks on too many unrealistic examples. Conversely, overly narrow or misspecified priors lead to ABI failure since the networks are never trained on relevant data. Another consideration is computational efficiency: density-based MCMC (e.g., Hamiltonian Monte Carlo) require priors whose density is fast to evaluate, whereas ABI benefits from priors that can be efficiently sampled from. This distinc-

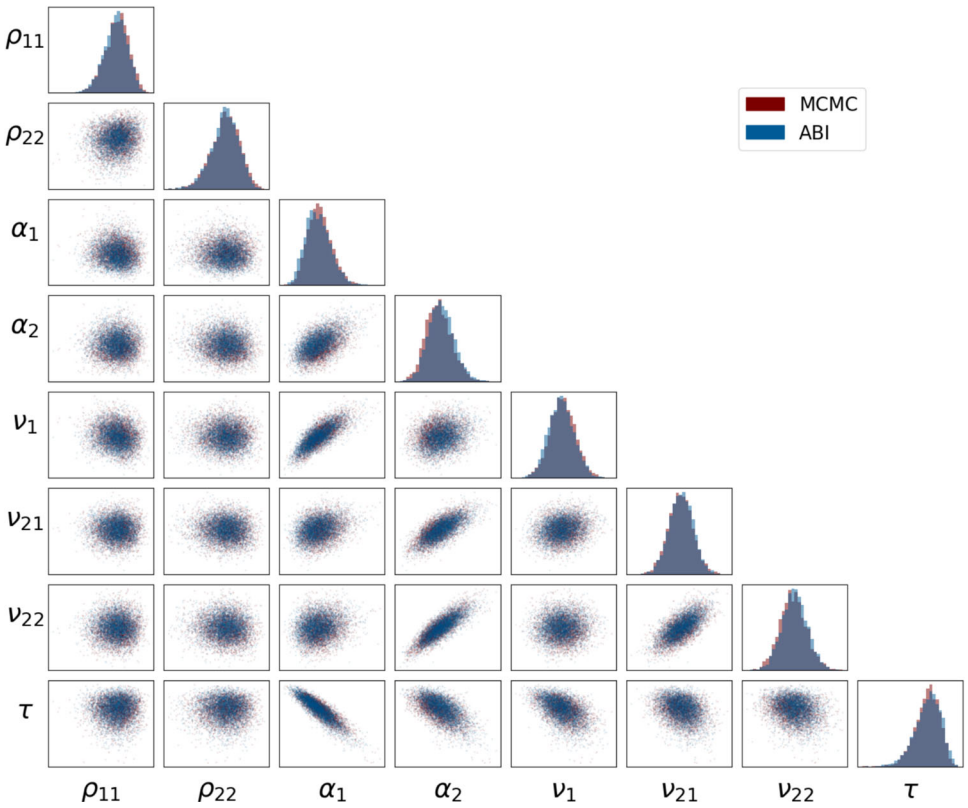


Fig. 14 *Case Study 3: Latent switches in cognitive processing.* Comparison of the parameter posterior samples between ABI and MCMC for participant C reported by Dutilh and Wagenmakesr, Visser, van der Maas (2011). The diagonals show the marginal parameter posteriors, whereas off-diagonal elements show the pairwise scatter plots to display dependencies between parameters. The two methods return nearly identical results, yielding almost perfectly overlapping distributions

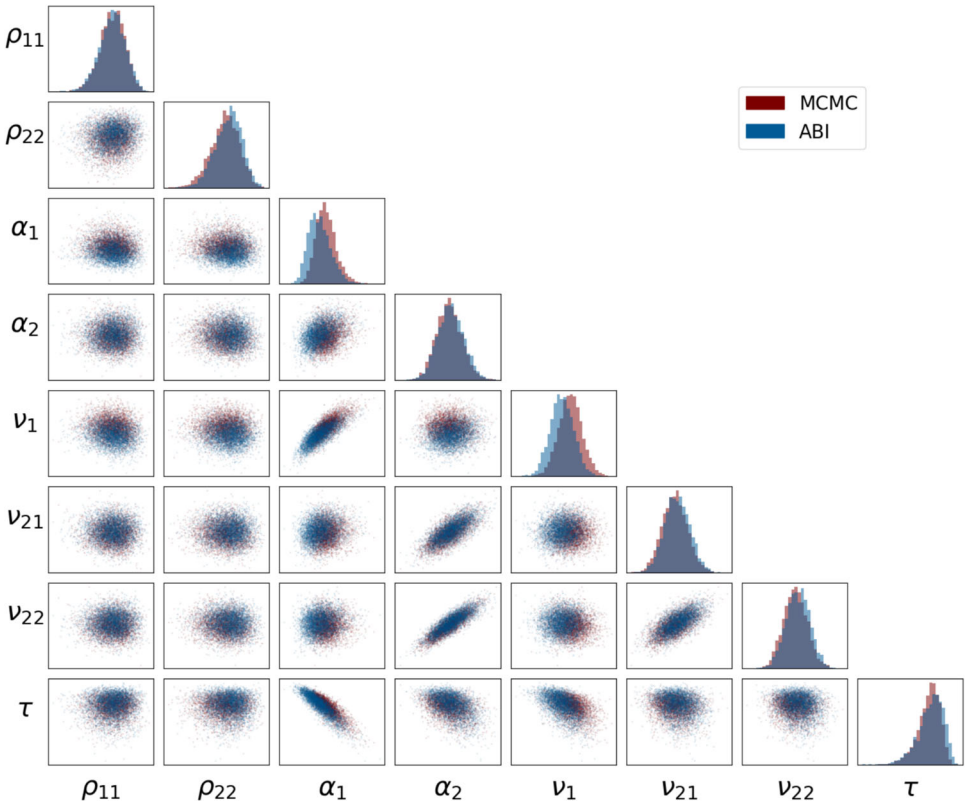


Fig. 15 Case Study 3: Latent switches in cognitive processing. Comparison of the parameter priors and posterior samples between ABI and MCMC for participant B reported by Dutilh and Wagenmakesr, Visser, van der Maas (2011). The diagonals show the marginal parameter posteriors, whereas off-diagonal elements show the pairwise scatter plots to display correlations between parameters. The ABI approximation is slightly biased but the estimates can be corrected by Pareto smoothed importance sampling

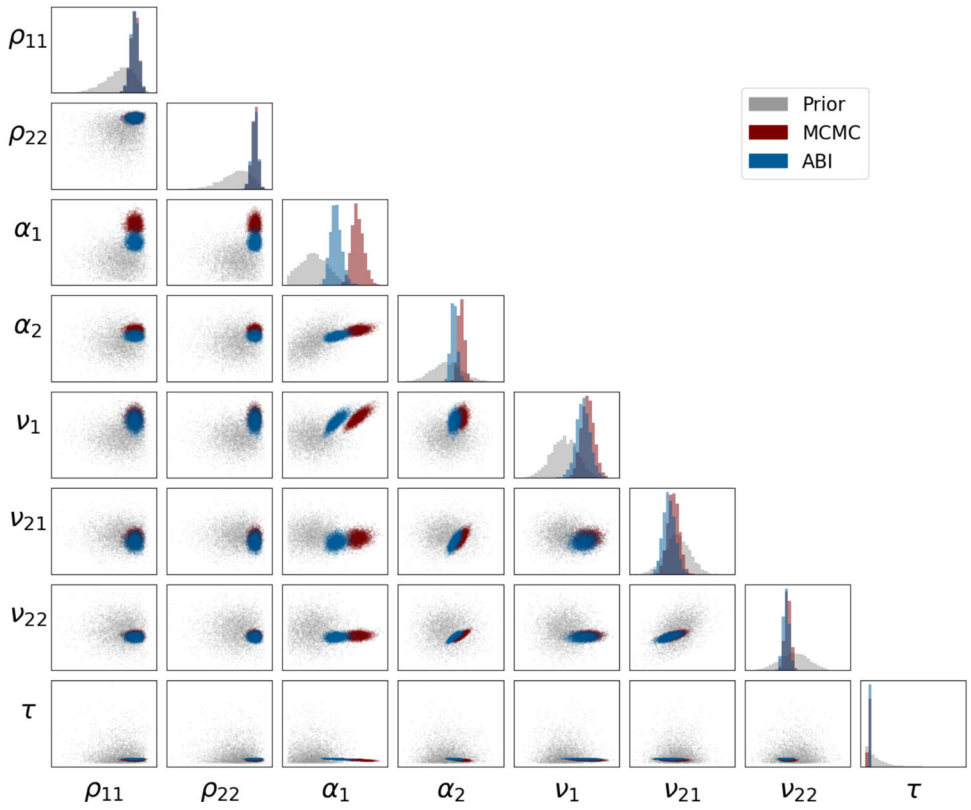


Fig. 16 Case Study 3: Latent switches in cognitive processing. Comparison of the parameter priors and posterior samples between ABI and MCMC for participant A reported by Dutilh and Wagenmakesr, Visser, van der Maas (2011). The diagonals show the marginal parameter posteriors, whereas off-diagonal elements show the pairwise scatter plots to display dependencies between parameters. The ABI approximation is severely biased and corrections using Pareto smoothed importance sampling will not be efficient

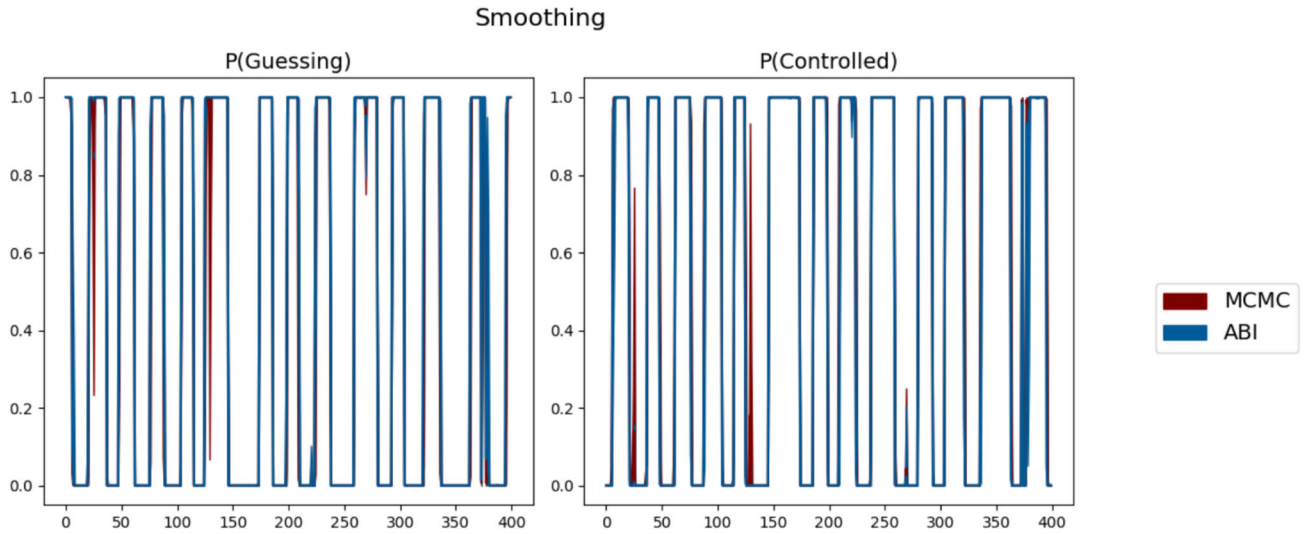


Fig. 17 *Case Study 3: Latent switches in cognitive processing.* Comparison between smoothing classification probabilities between ABI and MCMC for participant A reported by Dutilh and Wagenmakesr, Visser, van der Maas (2011). The two methods return nearly identical results, yielding almost perfectly overlapping lines

tion becomes relevant, for example, in our Gaussian mixture model. We adopted an order-restricted prior on component means, a parametrization common in `Stan`. While efficient for density evaluation, it necessitated rejection sampling for ABI. In our example, well-separated components kept rejection rates low, but with less separation or many more components, rejection sampling would become infeasible, and alternative prior specifications would be required.

One of the main strengths of ABI is inference speed. Across our examples, our implementation of ABI with `BayesFlow` consistently outperformed MCMC sampling with `Stan` by several orders of magnitude in inference time. This acceleration is important even when applying a model to a single dataset, since simulation-based validation procedures such as SBC (Talts et al. 2018; Modrák et al. 2023)

typically require fitting hundreds or thousands of datasets, which can become prohibitively expensive with MCMC.

However, the speed benefits apply primarily to the inference phase. *Total* model-fitting time may turn out to be substantially longer (Bürkner et al. 2023). Generating training data, training the neural networks, and tuning hyperparameters can demand considerable computational resources which add to the cost of ABI. Implementing ABI pipelines also entails developing and testing the simulator, designing and debugging neural architectures, and experimenting with different network configurations to achieve reliable inference, all of which contribute significant overhead. In our case studies, implementing the new neural factorization, debugging the simulator, and testing multiple network configurations to achieve satisfactory calibration required by far the most time and effort. Moreover, training and inference times depend strongly on network architecture. For example, normalizing flows are typically slower to train than flow-matching architectures (Lipman et al. 2022), whereas inference with normalizing flows tends to be faster, particularly on CPUs, than flow-matching. These observations underscore that, although ABI can offer substantial gains during inference, it does not necessarily outperform MCMC in overall efficiency in every use case.

Network architectures influence not only speed of training and inference, but also computational faithfulness of the networks. Inappropriate architectures (e.g., using permutation-invariant Deep Sets on time-series data) can distort the model structure, but even within appropriate architectural families, network accuracy is highly sensitive to hyperparameters such as the number of layers, hidden units, activation functions, etc. Simpler networks may underfit and fail to capture relevant patterns in the data; overly complex networks may overfit, reproducing noise from the training data and producing unstable outputs. We observed the latter phenomenon in our first case study, where a deep classification network (12 fully connected layers with ReLU activations) exhibited visibly "jittery" output, likely reflecting excessive capacity of the network (Novak et al. 2018). This setting was the result of our iterative experimentation rather than a principled search for the best hyperparameters. A more principled approach would be using some sort of tuning procedure; for example, Bayesian optimization (Snoek et al. 2012) or population-based training (Jaderberg et al. 2017).

Acknowledgements Paul Bürkner acknowledges support of the DFG Collaborative Research Center 391 (Spatio-Temporal Statistics for the Transition of Energy and Transport) – 520388526. Paul Bürkner further acknowledges support of the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projects 508399956 and 528702768.

Author Contributions Š.K. and P.B conceptualized the article. Š.K. implemented the framework and ran the case studies. Š.K. and P.B. wrote the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Data re-analyzed in this article is publicly available, and can be accessed through the code repository linked to from the paper.

Declarations

Competing interests The authors declare no competing interests.

References

Ambroise, C., Dang, M., Govaert, G.: Clustering of Spatial Data by the EM Algorithm. A. Soares, J. Gómez-Hernandez, and R. Froidevaux (Eds.), geoENV I — Geostatistics for Environmental Applications (pp. 493–504). Dordrecht: Springer Netherlands. (1997)

Anders, R., Alario, F.X., Van Maanen, L.: The shifted Wald distribution for response time data analysis. Psychol. Methods **21**(3), 309–327 (2016). https://doi.org/10.1037/met0000066

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., Köthe, U.: Guided Image Generation with Conditional Invertible Neural Networks. arXiv, (2019)

Arruda, J., Pandey, V., Sherry, C., Barroso, M., Intes, X., Hasenauer, J., Radev, S.T.: Compositional amortized inference for large-scale hierarchical bayesian models, (2025). arXiv:2505.14429 arXiv preprint

Baum, E.B.: On the capabilities of multilayer perceptrons. J. Complex. **4**(3), 193–215 (1988). https://doi.org/10.1016/0885-064X(88)90020-9

Boelts, J., Lueckmann, J.M., Gao, R., Macke, J.H.: Flexible and efficient simulation-based inference for models of decision-making. ELife **11**, 77220 (2022). https://doi.org/10.7554/eLife.77220

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., Cohen, J.D.: The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. Psychol. Rev. **113**(4), 700 (2006)

Brehmer, J.: Simulation-based inference in particle physics. Nature Reviews Physics **3**(5), 305–305 (2021). https://doi.org/10.1038/s42254-021-00305-6

Brehmer, J., Louppe, G., Pavez, J., Cranmer, K.: Mining gold from implicit models to improve likelihood-free inference. Proc. Natl. Acad. Sci. **117**(10), 5242–5249 (2020). https://doi.org/10.1073/pnas.1915980117

Bürkner, P.C., Scholz, M., Radev, S.T.: Some models are useful, but how do we know which ones? towards a unified bayesian model taxonomy. In: Statistic Surveys, vol. 17, pp. 216–310. (2023)

Bürkner, P.- C.: brms: An R package for Bayesian multilevel models Using Stan. Journal of Statistical Software, 80(1), 1–28, (2017). https://doi.org/10.18637/jss.v080.i01

Bürkner, P.C., Gabry, J., Vehtari, A.: Approximate leave-future-out cross-validation for Bayesian time series models. J. Stat. Comput. Simul. **90**(14), 2499–2523 (2020). https://doi.org/10.1080/00949655.2020.1783262

Bürkner, P.C., Gabry, J., Vehtari, A.: Efficient leave-one-out cross-validation for Bayesian non-factorized normal and Student-t models. Comput. Statistics **36**(2), 1243–1261 (2021). https://doi.org/10.1007/s00180-020-01045-4

Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Riddell, A.: Stan: A probabilistic programming language. J. Stat. Softw. **76**(1), 1–32 (2017). https://doi.org/10.18637/jss.v076.i01

Celeux, G., Hurn, M., Robert, C.P.: Computational and Inferential Difficulties with Mixture Posterior Distributions. J. Am. Stat. Assoc. **95**(451), 957–970 (2000). https://doi.org/10.1080/01621459.2000.10474285

Chen, Y., Zhang, D., Gutmann, M., Courville, A., Zhu, Z.: Neural Approximate Sufficient Statistics for Implicit Models. arXiv, (2021)

Chhikara, R., Folks, J.L.: The inverse Gaussian distribution: Theory, methodology, and applications (No. 95). Boca Raton London New York: CRC Press, Taylor & Francis Group (2019)

Cranmer, K., Brehmer, J., Louppe, G.: The frontier of simulation-based Inference. Proc. Natl. Acad. Sci. **117**(48), 30055–30062 (2020). https://doi.org/10.1073/pnas.1912789117

Dax, M., Wildberger, J., Buchholz, S., Green, S.R., Macke, J.H., Schölkopf, B.: Flow Matching for Scalable Simulation-Based Inference. arXiv, (2023)

Diebolt, J., Robert, C.P.: Estimation of Finite Mixture Distributions Through Bayesian Sampling. J. Roy. Stat. Soc.: Ser. B (Methodol.) **56**(2), 363–375 (1994). https://doi.org/10.1111/j.2517-6161.1994.tb01985.x

Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp, (2016). arXiv:1605.08803 arXiv preprint

Durkan, C., Bekasov, A., Murray, I., Papamakarios, G.: Neural spline flows. In: Advances in neural information processing systems, p. 32. (2019)

Dutilh, G., Wagenmakers, E.J., Visser, I., van der Maas, H.L.J.: A Phase Transition Model for the Speed-Accuracy Trade-Off in Response Time Experiments. Cogn. Sci. **35**(2), 211–250 (2011). https://doi.org/10.1111/j.1551-6709.2010.01147.x

Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)

Elsemüller, L., Olischläger, H., Schmitt, M., Bürkner, P.C., Köthe, U., Radev, S.T.: Sensitivity-Aware Amortized Bayesian Inference. arXiv, (2024)

Evans, N.J., Wagenmakers, E.J., et al.: Evidence accumulation models: Current limitations and future directions. The Quantitative Methods for Psychology **16**(2), 73–90 (2020)

Frühwirth-Schnatter, S.: Markov chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models. J. Am. Stat. Assoc. **96**(453), 194–209 (2001). https://doi.org/10.1198/016214501750333063

Frühwirth-Schnatter, S.: Finite mixture and Markov switching models. Springer, New York (2006)

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., Gelman, A.: Visualization in Bayesian Workflow. J. R. Stat. Soc. Ser. A Stat. Soc. **182**(2), 389–402 (2019). https://doi.org/10.1111/rssa.12378

Ganesalingam, S.: Classification and Mixture Approaches to Clustering via Maximum Likelihood. Appl. Stat. **38**(3), 455–466 (1989). https://doi.org/10.2307/2347733

Gelman, A., Meng, X.L., Stern, H.: Posterior Predictive Assessment of Model Fitness Via Realized Discrepancies. Stat. Sin. **6**(4), 733–760 (1996)

Gelman, A., Vehtari, A., Simpson, D., Margossian, C.C., Carpenter, B., Yao, Y., Modrák, M.: (2020). Bayesian Workflow. arXiv

Gers, F., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470) (Vol. 2, pp. 850–855 vol.2). (ISSN: 0537-9989) (1999)

Gershman, S., Goodman, N.: Amortized Inference in Probabilistic Reasoning. Proceedings of the Annual Meeting of the Cognitive Science Society, 36(36), (2014)

Gonçalves, P.J., Lueckmann, J. M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G. others Training deep neural density estimators to identify mechanistic models of neural dynamics. elife, 9, e56261, (2020)

Habermann, D., Schmitt, M., Kühmichel, L., Bulling, A., Radev, S.T., Bürkner, P.C.: Amortized Bayesian Multilevel Models, (2024). arXiv

Hadj-Amar, B., Jewson, J., Fiecas, M.: Bayesian Approximations to Hidden Semi-Markov Models for Telemetric Monitoring of Physical Activity. Bayesian Anal. **18**(2), 547–577 (2023). https://doi.org/10.1214/22-BA1318

Heinrich, L., Mishra-Sharma, S., Pollard, C., Windischhofer, P.: Hierarchical Neural Simulation-Based Inference Over Event Ensembles. arXiv, (2024)

Hermans, J., Delaunoy, A., Rozet, F., Wehenkel, A., Louppe, G.: A Crisis In Simulation-Based Inference? Beware, Your Posterior Approximations Can Be Unfaithful. Transactions on Machine Learning Research, , https://hdl.handle.net/2268/265148 (2022)

Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A.. others Population based training of neural networks. arXiv preprint arXiv:1711.09846, (2017)

Jasra, A., Holmes, C.C., Stephens, D.A.: Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. Stat. Sci. **20**(1), 50–67 (2005). https://doi.org/10.1214/088342305000000016

Karatzas, I., Shreve, S.: Brownian motion and stochastic calculus, vol. 113, springer (2014)

Kobyzev, I., Prince, S.J.D., Brubaker, M.A.: Normalizing Flows: An Introduction and Review of Current Methods. arXiv, (2020)

Kucharský, V., Tran, N.H., Veldkamp, K., Raijmakers, M., Visser, I.: Hidden Markov Models of Evidence Accumulation in Speeded Decision Tasks. Computational Brain & Behavior **4**(4), 416–441 (2021). https://doi.org/10.1007/s42113-021-00115-0

Kumar, R., Carroll, C., Hartikainen, A., Martin, O. (2019). Arviz a unified library for exploratory analysis of bayesian models in python. Journal of Open Source Software, 4(33), 1143, https://doi.org/10.21105/joss.01143

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the ieee (Vol. 86, pp. 2278–2324). IEEE (1998)

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. International conference on machine learning (pp. 3744–3753) (2019)

Lember, J., Gasbarra, D., Koloydenko, A., Kuljus, K.: Estimation of Viterbi path in Bayesian hidden Markov models. METRON **77**(2), 137–169 (2019). https://doi.org/10.1007/s40300-019-00152-7

Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling, (2022). arXiv:2210.02747 arXiv preprint

Liu, Y., Zhang, H.H., Wu, Y.: Hard or Soft Classification? Large-margin Unified Machines. J. Am. Stat. Assoc. **106**(493), 166–177 (2011). https://doi.org/10.1198/jasa.2011.tm10319

Lopez-Paz, D., Oquab, M.: Revisiting classifier two-sample tests, (2016). arXiv:1610.06545 arXiv preprint

Luce, R.D.: Response Times: Their Role in Inferring Elementary Mental Organization. Oxford University Press, Oxford, UK (1991)
Lueckmann, J.- M., Boelts, J., Greenberg, D., Goncalves, P., Macke, J.: Benchmarking Simulation-Based Inference. A. Banerjee and K. Fukumizu (Eds.), Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Vol. 130, pp. 343–351) (2021)
Marin, J.- M., Mengersen, K., Robert, C.P.: Bayesian Modelling and Inference on Mixtures of Distributions. D.K. Dey and C.R. Rao (Eds.), Handbook of Statistics (Vol. 25, pp. 459–507). Elsevier (2005)
Mark, C., Metzner, C., Lautscham, L., Strissel, P.L., Strick, R., Fabry, B.: Bayesian model selection for complex dynamic systems. Nat. Commun. **9**(1), 1803 (2018). https://doi.org/10.1038/s41467-018-04241-5
May, P.B., Finley, A.O., Dubayah, R.O.: A Spatial Mixture Model for Spaceborne Lidar Observations Over Mixed Forest and Non-forest Land Types. J. Agric. Biol. Environ. Stat. **29**(4), 671–694 (2024). https://doi.org/10.1007/s13253-024-00600-6
McLachlan, G.J.: The classification and mixture maximum likelihood approaches to cluster analysis. P.R. Krishnaiah and L.N. Kanal (Eds.), Handbook of Statistics (Vol. 2, pp. 199–208). Elsevier (1982)
McLachlan, G.J., Basford, K.E.: Mixture models: Inference and applications to clustering, vol. 84. Dekker, New York, N.Y (1988)
Miller, J.W., Harrison, M.T.: Mixture models with a prior on the number of components. J. Am. Stat. Assoc. **113**(521), 340–356 (2018)
Modrák, M., Moon, A.H., Kim, S., Bürkner, P., Huurre, N., Faltejsková, K., Vehtari, A.: Simulation-Based Calibration Checking for Bayesian Computation: The Choice of Test Quantities Shapes Sensitivity. Bayesian Anal. (2023). https://doi.org/10.1214/23-BA1404
Murtagh, F.: Multilayer perceptrons for classification and regression. Neurocomputing **2**(5), 183–197 (1991). https://doi.org/10.1016/0925-2312(91)90023-5
Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. arXiv preprint arXiv:1802.08760, ,
Papamakarios, G., Murray, I.: Fast ε-free inference of simulation models with bayesian conditional density estimation. Advances in neural information processing systems, 29, (2016)
Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. **22**(1), 57:2617-57:2680 (2021)
Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Proceedings of the IEEE, vol. 77, pp. 257–286. (1989)
Radev, S.T., Mertens, U.K., Voss, A., Ardizzone, L., Kothe, U.: BayesFlow: Learning Complex Stochastic Models With Invertible Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, vol. 33, pp. 1452–1466. (2022)
Ratcliff, R.: A theory of memory retrieval. Psychol. Rev. **85**(2), 59 (1978)
Ratcliff, R., McKoon, G.: The diffusion decision model: theory and data for two-choice decision tasks. Neural Comput. **20**(4), 873–922 (2008)
Ratcliff, R., Smith, P.L., Brown, S.D., McKoon, G.: Diffusion decision model: Current issues and history. Trends Cogn. Sci. **20**(4), 260–281 (2016)
Rezende, D., Mohamed, S.: Variational inference with normalizing flows. International conference on machine learning (pp. 1530–1538) (2015)
Ritchie, D., Horsfall, P., Goodman, N.D.: Deep Amortized Inference for Probabilistic Programs. arXiv, (2016)
Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386–408 (1958). https://doi.org/10.1037/h0042519
Rossum, G.V., Drake, F.L.: The Python language reference, Python Software Foundation (2010). (Release 3.0.1 [Repr.] ed.) (No. Pt. 2)
Samé, A.: Clustering Spatial Data via Mixture Models with Dynamic Weights. W. Lu and K.Q. Zhu (Eds.), Trends and Applications in Knowledge Discovery and Data Mining (pp. 128–138). Cham: Springer International Publishing (2020)
Schaaf, J.V., Jepma, M., Visser, I., Huizenga, H.M.: A hierarchical Bayesian approach to assess learning and guessing strategies in reinforcement learning. J. Math. Psychol. **93**, 102276 (2019). https://doi.org/10.1016/j.jmp.2019.102276
Schad, D.J., Betancourt, M., Vasishth, S.: Toward a principled Bayesian workflow in cognitive science. arXiv, (2019)
Schmitt, M., Bürkner, P.- C., Köthe, U., Radev, S.T.: Detecting Model Misspecification in Amortized Bayesian Inference with Neural Networks. U. Köthe and C. Rother (Eds.), Pattern Recognition (Vol. 14264, pp. 541–557). Cham: Springer Nature Switzerland. (Series Title: Lecture Notes in Computer Science) (2024)
Schmitt, M., Li, C., Vehtari, A., Acerbi, L., Bürkner, P.C., Radev, S.T.: Amortized Bayesian Workflow, (2024). Extended Abstract). arXiv
Schmitt, M., Pratz, V., Köthe, U., Bürkner, P.C., Radev, S.T.: Consistency Models for Scalable and Fast Simulation-Based Inference. arXiv, (2024)
Schröder, C., Macke, J.H.: Simultaneous identification of models and parameters of scientific simulators, (2023). arXiv:2305.15174 arXiv preprint
Schumacher, L., Bürkner, P.C., Voss, A., Köthe, U., Radev, S.T.: Neural superstatistics for Bayesian estimation of dynamic cognitive models. Sci. Rep. **13**(1), 13778 (2023). https://doi.org/10.1038/s41598-023-40278-3
Schumacher, L., Schnuerch, M., Voss, A., Radev, S.T.: Validation and Comparison of Non-stationary Cognitive Models: A Diffusion Model Application. Computational Brain & Behavior (2024). https://doi.org/10.1007/s42113-024-00218-4
Scrucca, L., Fraley, C., Murphy, T.B., Raftery, A.E.: Model-Based Clustering, Classification, and Density Estimation Using mclust in R. Chapman and Hall/CRC, New York (2023)
Sharrock, L., Simons, J., Liu, S., Beaumont, M.: Sequential Neural Score Estimation: Likelihood-Free Inference with Conditional Score Based Diffusion Models, (2024). arXiv
Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, p. 25. (2012)
Stephens, M., P., D.: Bayesian methods for mixtures of normal distributions, (1997)
Swanson, D.M.: Blocked gibbs sampling for improved convergence in finite mixture models, (2024). arXiv:2411.00371 arXiv preprint
Särkkä, S., Svensson, L.: Bayesian Filtering and Smoothing, 2nd edn. Cambridge University Press, Cambridge, UK (2023)
Talts, S., Betancourt, M., Simpson, D., Vehtari, A., Gelman, A.: Validating Bayesian inference algorithms with simulation-based calibration. arXiv, , https://doi.org/10.48550/arXiv.1804.06788 (2018)
Tillman, G., Van Zandt, T., Logan, G.D.: Sequential sampling models without random between-trial variability: the racing diffusion model of speeded decision making. Psychonomic Bulletin & Review **27**(5), 911–936 (2020). https://doi.org/10.3758/s13423-020-01719-6
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, vol. 30, (2017)
Vehtari, A., Gelman, A., Gabry, J.: Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat. Comput.

27(5), 1413–1432 (2017). https://doi.org/10.1007/s11222-016-9696-4

Vehtari, A., Simpson, D., Gelman, A., Yao, Y., Gabry, J.: Pareto Smoothed Importance Sampling. arXiv, (2024). arXiv:abs/1507.02646

Visser, I., Speekenbrink, M.: Mixture and Hidden Markov Models with R. Springer International Publishing, Cham (2022)

Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Inf. Theory **13**, 260–269 (1967)

von Krause, M., Radev, S.T., Voss, A.: Mental speed is high until age 60 as revealed by analysis of over a million participants. Nat. Hum. Behav. **6**(5), 700–708 (2022). https://doi.org/10.1038/s41562-021-01282-7

Wahba, G.: Soft and hard classification by reproducing kernel Hilbert space methods. Proc. Natl. Acad. Sci. **99**(26), 16524–16530 (2002). https://doi.org/10.1073/pnas.242574899

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep Sets. Advances in Neural Information Processing Systems, vol. 30, (2017)

Zavadskiy, G., Zantedeschi, D., Jank, W.: A functional Hidden Markov Model to incorporate dynamics into Bayesian optimal stopping problems: Helping physicians manage traumatic brain injuries. Decis. Support Syst. **177**, 114078 (2024). https://doi.org/10.1016/j.dss.2023.114078

Zeghal, J., Lanusse, F., Boucaud, A., Remy, B., Aubourg, E.: Neural Posterior Estimation with Differentiable Simulators. arXiv, (2022)

Zucchini, W., MacDonald, I.L., Langrock, R.: Hidden Markov Models for Time Series: An Introduction Using R, Second Edition, 2nd edn. Chapman and Hall/CRC, New York (2016)