FLEXIBLE AND EFFICIENT SIMULATION-BASED INFERENCE FOR MODELS OF DECISION-MAKING

Jan Boelts[1,2] **Jan-Matthis Lueckmann**[1] **Richard Gao**[1] **Jakob H. Macke**[1,3]

[1] Machine Learning in Science, Excellence Cluster Machine Learning, University of Tübingen
[2] Technical University of Munich
[3] Max Planck Institute for Intelligent Systems Tübingen

December 22, 2021

ABSTRACT

Identifying parameters of computational models that capture experimental data is a central task in cognitive neuroscience. Bayesian statistical inference aims to not only find a single configuration of best-fitting parameters, but to recover all model parameters that are consistent with the data and prior knowledge. Statistical inference methods usually require the ability to evaluate the likelihood of the model—however, for many models of interest in cognitive neuroscience, the associated likelihoods cannot be computed efficiently. Simulation-based inference (SBI) offers a solution to this problem by only requiring access to simulations produced by the model. Here, we provide an efficient SBI method for models of decision-making. Our approach, Mixed Neural Likelihood Estimation (MNLE), trains neural density estimators on model simulations to emulate the simulator. The likelihoods of the emulator can then be used to perform Bayesian parameter inference on experimental data using standard approximate inference methods like Markov Chain Monte Carlo sampling. While most neural likelihood estimation methods target continuous data, MNLE works with mixed data types, as typically obtained in decision-making experiments (e.g., binary decisions and associated continuous reaction times). We demonstrate MNLE on the classical drift-diffusion model (DDM) and compare its performance to a recently proposed method for SBI on DDMs, called likelihood approximation networks (LAN, Fengler et al. 2021). We show that MNLE is substantially more efficient than LANs, requiring up to six orders of magnitudes fewer model simulations to achieve comparable likelihood accuracy and evaluation time while providing the same level of flexibility. We include an implementation of our algorithm in the user-friendly open-source package `sbi`.

1 Introduction

Computational modeling is an essential part of the scientific process in cognitive neuroscience: Models are developed from prior knowledge and hypotheses, and compared to experimentally observed phenomena (Churchland and Sejnowski, 1988; McClelland, 2009). Computational models usually have free parameters which need to be tuned to find those model(s) that capture experimental data. This is often approached by searching for single best-fitting parameters using grid search or optimization methods. While this point-wise approach has been used successfully (e.g., Lee et al., 2016; Patil et al., 2016) it can be scientifically more informative to perform Bayesian inference over the model parameters: Bayesian inference takes into account prior knowledge, reveals *all* the parameters consistent with observed data, and thus can be used for quantifying uncertainty, hypothesis testing, and model selection (Lee, 2008; Shiffrin et al., 2008; Lee and Wagenmakers, 2014; Schad et al., 2021). Yet, as the complexity of models used in cognitive neuroscience increases, Bayesian inference becomes challenging for two reasons. First, for some commonly used models, computational evaluation of likelihoods is not possible, rendering standard approximate inference methods like Markov chain Monte Carlo (MCMC) inapplicable. Second, the kind of models and experimental paradigms in cognitive neuroscience often induce scenarios in which inference is repeated for varying numbers of experimental trials and changing hierarchical dependencies between model parameters (Lee, 2011). As such, fitting computational models with arbitrary hierarchical structures to experimental data often still requires idiosyncratic and complex inference algorithms.

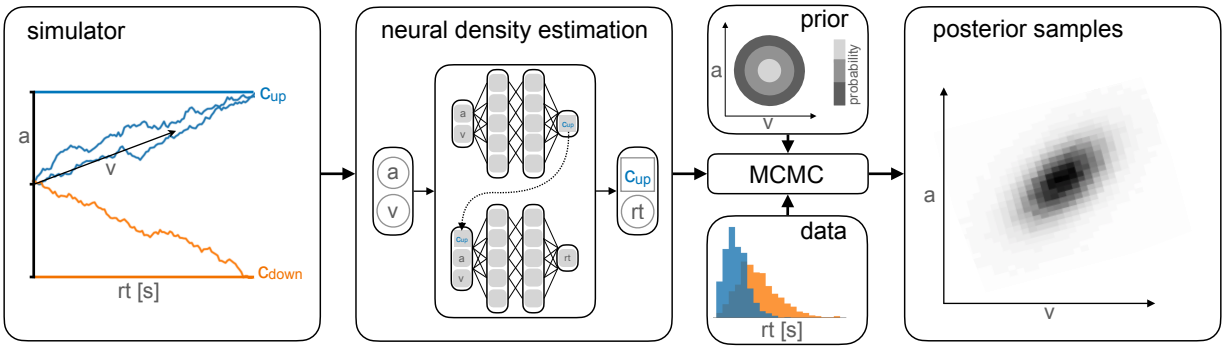

Figure 1: **Training a neural likelihood estimator on simulated data to perform parameter inference** Our goal is to perform statistical inference on models of decision-making for which likelihoods cannot be evaluated (here a drift-diffusion model for illustration, left). We train a neural density estimation network on synthetic data generated by the model, to provide access to (estimated) likelihoods. We use neural density estimators which account for the mixed data types of decision-making models (e.g. discrete valued choices and continuous valued reaction times, middle). These estimates can then be used for inference with MCMC, i.e., to obtain samples from the posterior over the parameters of the simulator given experimental data (right). The inference can be performed for many inference scenarios like varying number of trials or hierarchical inference without re-training the neural likelihood estimator.

Approximate Bayesian computation (ABC, Sisson et al., 2018) offers a solution to the first challenge by enabling Bayesian inference based on comparing simulated with experimental data, without the need to evaluate an explicit likelihood function. Accordingly, various ABC methods have been applied to and developed for models in cognitive neuroscience and related fields (Turner and Van Zandt, 2012, 2018; Palestro et al., 2018; Kangasrääsiö et al., 2019). However, ABC methods are limited regarding the second challenge because they become inefficient as the number of model parameters increases (Lueckmann et al., 2021) and require generating new simulations whenever the observed data or parameter dependencies change.

More recent approaches from the field simulation-based inference (SBI, Cranmer et al., 2020) have the potential to overcome these limitations by using machine learning algorithms such as neural networks. Recently, Fengler et al. (2021) presented an SBI-algorithm for a specific problem in cognitive neuroscience—inference for drift-diffusion models (DDM). They introduced a new approach, called likelihood approximation networks (LANs), which uses neural networks to predict log-likelihoods from data and parameters. The predicted likelihoods can subsequently be used to generate posterior samples using MCMC methods. LANs are trained in a three-step procedure. First, a set of N parameters is generated and for each of the N parameters the model is simulated M times. Second, for each of the N parameters, empirical likelihood targets are estimated from the M model simulations using kernel density estimation (KDE) or empirical histograms. Third, a training dataset consisting of parameters, data points and empirical likelihood targets is constructed by augmenting the initial set of N parameters by a factor of 1000: for each parameter, 1000 data points and empirical likelihood targets are generated from the learned KDE. Finally, supervised learning is used to train a neural network to predict log-likelihoods, by minimizing a loss function (the Huber loss) between the network-predicted log-likelihoods and the (log of) the empirically estimated likelihoods. Overall, LANs require a large number of model simulations such that the histogram-probability of each possible observed data and for each possible combination of input parameters, can be accurately estimated—$N \cdot M$ model simulations, e.g., $1.5 \cdot 10^6 \cdot 10^5$ (150 billion) for the examples used in Fengler et al. (2021). The extremely high number of model simulations will make it infeasible for most users to run this training themselves, so that there would need to be a repository from which users can download pre-trained LANs. This restricts the application of LANs to a small set of canonical models like drift-diffusion models, and prohibits customization and iteration of models by users. In addition, the high simulation requirement limits this approach to models whose parameters and observations are sufficiently low-dimensional for histograms to be sampled densely.

To overcome these limitations, we propose an alternative approach called Mixed Neural Likelihood Estimation (MNLE). MNLE builds on recent advances in probabilistic machine learning, and in particular on the framework of neural likelihood estimation (Papamakarios et al., 2019b; Lueckmann et al., 2019) but is designed to specifically capture the mixed data types arising in models of decision-making, e.g., discrete choices and continuous reaction times. Neural likelihood estimation has its origin in classical synthetic likelihood (SL) approaches (Wood, 2010; Drovandi et al., 2018). SL assumes the likelihood of the simulation-based model to be Gaussian (so that its moments can be estimated from model simulations) and then uses MCMC methods for inference. This approach and various extensions of it have

been widely used (Price et al., 2018; Ong et al., 2018; An et al., 2019; Priddle et al., 2021)—but inherently they need multiple model simulations for each parameter in the MCMC chain to estimate the associated likelihood.

Neural likelihood approaches instead perform *conditional* density estimation, i.e., they train a neural network to predict the parameters of the approximate likelihood conditioned on the model parameters (e.g. Papamakarios et al., 2019b; Lueckmann et al., 2019). By using a conditional density estimator, it is possible to exploit continuity across different model parameters, rather than having to learn a separate density for each individual parameter as in classical SL. Recent advances in conditional density estimation (such as normalizing flows (Papamakarios et al., 2019a)) further allow lifting the parametric assumptions of classical SL methods and learning flexible conditional density estimators which are able to model a wide range of densities, as well as highly nonlinear dependencies on the conditioning variable. In addition, the neural likelihood estimator yields estimates of the probability density which are guaranteed to be non-negative and normalized, and which can be both sampled and evaluated, acting as a probabilistic emulator of the simulator (Lueckmann et al., 2019).

Our approach, MNLE, uses neural likelihood estimation to learn an emulator of the simulator. The training phase is a simple two-step procedure: first, a training dataset of N parameters $\boldsymbol{\theta}$ is sampled from a proposal distribution and corresponding model simulations $\mathbf{x}$ are generated and second, the N parameter-data pairs $(\boldsymbol{\theta}, \mathbf{x})$ are directly used to train a conditional neural likelihood estimator to estimate $p(\mathbf{x}|\boldsymbol{\theta})$. Like for LANs, the proposal distribution for the training data can be *any* distribution over $\boldsymbol{\theta}$, and should be chosen to cover all parameter-values one expects to encounter in empirical data. Here, we choose to use the prior that is used in the inference phase as the proposal distribution. To account for mixed data types, we learn the likelihood estimator as a mixed model composed of one neural density estimator for categorical data and one for continuous data, conditioned on the categorical data. This separation allows us to choose the appropriate neural density estimator for each data type, e.g., a Bernoulli model for the categorical data and a normalizing flow (Papamakarios et al., 2019a) for the continuous data. The resulting joined density estimator gives access to the likelihood, which enables inference via MCMC methods, and it acts as an emulator that can generate synthetic data. See Fig.1 for an illustration of our approach, and section 4 for details.

Both LANs and MNLEs allow for flexible inference scenarios common in cognitive neuroscience, e.g., varying number of trials with same underlying experimental conditions or hierarchical inference, and need to be trained only once. However, there is a key difference between the two approaches. LANs use feed-forward neural networks to perform regression from model parameters to empirical likelihood targets obtained from KDE. MNLE learns the likelihood directly by performing conditional density estimation on the simulated data without requiring likelihood targets. This makes MNLE by design more simulation efficient than LANs—we will demonstrate empirically that it can learn likelihood-estimators which are as good, or better, than those of LAN, but using a factor of 1.000.000 fewer simulations than those reported for LANs (Fengler et al., 2021). Moreover, MNLE results in a density estimator that is guaranteed to correspond to a valid probability distribution and can also act as an emulator that can be sampled to generate synthetic data without running the simulator. The simulation-efficiency of MNLEs allows users to explore and iterate on their own models without generating a massive training dataset, rather than being restricted to canonical models for which pre-trained networks have been provided by a central resource. To facilitate this process, we implemented our method as an extension to an open-source toolbox for SBI methods (Tejero-Cantero et al., 2020), and provide tutorials for user-friendly access.

2 Results

2.1 Evaluating the performance of mixed neural likelihood estimation (MNLE) on the drift-diffusion model

We first demonstrate the efficiency and performance of MLNEs on a classical model of decision-making, the drift-diffusion model (DDM, Ratcliff and McKoon, 2008). The DDM is an influential phenomenological model of a two-alternative perceptual decision-making task. It simulates the evolution of an internal decision variable that integrates sensory evidence until one of two decision boundaries is reached and a choice is made (Fig.1, left). The decision variable is modeled with a stochastic differential equation which, in the "simple" DDM version (as used in Fengler et al., 2021), has four parameters: the drift rate v, boundary separation a, the starting point w of the decision variable, and the non-decision time τ. Given these four parameters $\boldsymbol{\theta} = (v, a, w, \tau)$, a single simulation of the DDM returns data $\mathbf{x}$ containing a choice $c \in \{0, 1\}$ and the corresponding reaction time in seconds $rt \in (\tau, \infty)$: $\mathbf{x} = (c, rt)$.

This version of the DDM is the ideal candidate for comparing the performance of different inference methods because the likelihood of an observation given the parameters, $L(\mathbf{x}|\boldsymbol{\theta})$, *can* be calculated analytically (Navarro and Fuss, 2009, in contrast to more complicated versions of the DDM, e.g., Ratcliff and Rouder (1998); Usher and McClelland (2001); Reynolds and Rhodes (2009)). We evaluated MNLE's performance, and compared against the publicly available pre-trained LAN neural networks (Fengler et al., 2021), based on the analytical likelihoods and the inferred posteriors of the DDM.

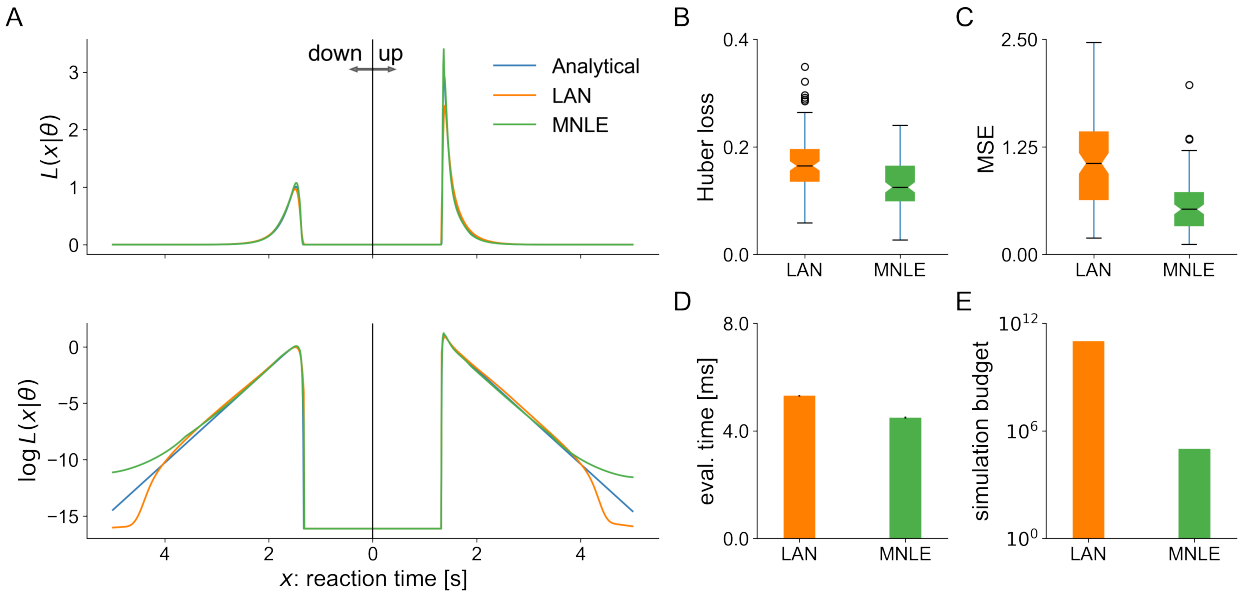

Figure 2: **MNLE estimates accurate likelihoods for the drift-diffusion model.** The classical drift-diffusion model (DDM) simulates reaction times and choices of a two-alternative decision task and has an analytical likelihood which can be used for benchmarking likelihood approximations. **A)** Example likelihood for a fixed parameter $\boldsymbol{\theta}$ over a range of reaction times (reaction times for down- and up-choices shown towards the left and right, respectively). **B)** Huber loss between analytical and estimated log-likelihoods shown for 100 simulated data calculated over 1000 test parameters sampled from the prior. Box plot notches indicate 95% confidence intervals around the median (black line). **C)** Same as in B but using mean squared error. **D)** Evaluation time for a batch of observed data and parameters as used during inference given a 100-trial DDM observation and MCMC sampling with ten parallel chains. Shown as mean over 100 repetitions, black bars indicate standard error of the mean. **E)** Number of simulations from the model required for training. LAN results reproduced using pre-trained LANs from Fengler et al. (2021). Note the usage of log-scale.

2.2 MNLE learns accurate likelihoods with a fraction of the simulation budget

First, we evaluated the quality of likelihood approximations of MNLE, and compared it to that of a LAN. Both MNLEs and LANs were able to accurately approximate the likelihoods for both decisions and a wide range of reaction times (see Fig.2A for an example, and section 4.3 for details of the comparison), although some deviations become visible in the log-domain, for extremely unlikely reaction times (e.g. for likelihoods less than 10^{-8}).

To quantitatively evaluate the quality of approximation, we calculated the Huber loss and mean-squared error (MSE) between the true and approximated log-likelihoods (Fig.2B,C). The metrics were calculated as averages over (approximate) likelihoods of a fixed observation given 1000 parameters sampled from the prior, repeated for 100 observations simulated from the DDM. We found that MNLEs were at least as accurate as LANs (median Huber loss, lower is better, MNLE: 0.12, LAN: 0.16). On 85 out of the 100 comparisons, MNLE had a smaller Huber loss than LAN. This pairwise comparison is significant under the binomial test ($p < 10^{-12}$), but note that these are simulated data and therefore the p-value can be arbitrarily inflated by increasing the number of comparisons. We note that the Huber loss is the loss which is directly optimized by LANs, and thus this comparison should in theory favor LANs over alternative approaches. When using the MSE for comparison, we similarly found that MNLEs achieved an error which was as small, or smaller, than that of LANs (median MSE, MNLE: 0.52, LAN: 1.06; Fig.2C). In this case, MNLEs achieved a smaller MSE than LANs for 89 out of 100 comparisons ($p < 10^{-15}$, binomial test).

When using the learned likelihood estimators for inference with MCMC methods, their evaluation speed is critical because MCMC often requires thousands of likelihood evaluations. We found that evaluating MNLE for a batch of 100 trials and ten model parameters (as used during MCMC) took 4.34 ± 0.04ms, compared to 5.2 ± 0.03ms for LANs (mean over 100 repetitions $\pm$ standard error of the mean), i.e., MNLE did not incur a larger computational foot-print at evaluation time (Fig.2D, see section 4.3 for details of computing infrastructure and parameter settings).

Finally, we compared the number of model simulations required for training: For this example, MNLE was trained using 10^5 simulations. In contrast, for the LAN results in Fengler et al. (2021) which we used for these comparisons,

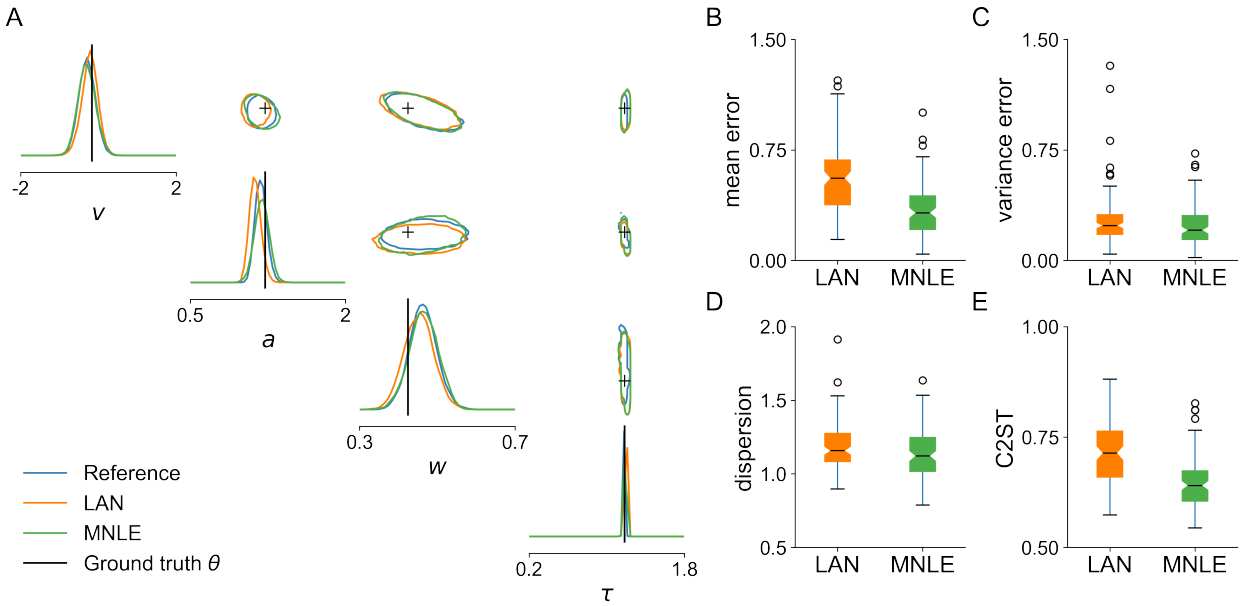

Figure 3: **Posterior accuracy for the drift-diffusion model.** Posteriors given a 100-trial i.i.d. observation, obtained with MCMC using analytical (i.e., reference) likelihoods, or those approximated using LANs and MNLE, respectively. **A)** Posterior samples shown as a corner-plot, i.e., one-dimension marginal histograms (diagonal) and all pair-wise two-dimensional marginals (upper triangle). **B)** Absolute difference in posterior sample mean normalized by reference posterior standard deviation (zero is best). **C)** Absolute difference in posterior variances normalized by reference posterior variance (zero is best). **D)** Ratio of approximate and reference posterior variance (one is best). **E)** Classification 2-sample test (C2ST) accuracy between approximate (LANs, MNLE) and reference posterior samples (0.5 is best). All box plots show metrics calculated from 100 repetitions with different observations; box plot notches represent the 95% confidence interval around the median (black line).

more than 10^{11} simulations were used (they used 1.5×10^6 different parameter settings for training and simulated 10^5 times for each of them to obtain empirical likelihood-targets).

Thus, in summary, we found that MNLE matched the performance of LANs on the accuracy metrics (Fig.2B,C), was equally fast to evaluate (Fig.2D), but using up to six orders of magnitudes fewer simulations during training (Fig.2E).

2.3 MNLE enables accurate flexible posterior inference with MCMC

In the previous section, we showed that both methods were similarly accurate in approximating likelihoods. To investigate whether these likelihood-estimates were accurate enough to support accurate parameter inference, we evaluated the quality of the resulting posteriors, using a framework for benchmarking SBI performance (Lueckmann et al., 2021). We used the analytical likelihoods of the simple DDM model to obtain reference posteriors using MCMC sampling, for 100 different observations. Each observation consisted of 100 i.i.d. trials simulated with the same parameter sampled from the prior (see Fig.3A for an example, details in section 4). We found that MNLEs approximated the posterior mean more accurately than LANs (median relative mean error MNLE: 0.32, LANs: 0.56), and was more accurate in 76 out of 100 comparisons ($p < 10^{-6}$; see Fig.3B). In terms of the posterior variance, MNLE performed on par with LANs (median relative variance error MNLE: 0.21, LAN: 0.24), being more accurate in 58 of 100 pairwise comparisons ($p = 0.13$). Additionally, we used 'dispersion', i.e., the ratio of estimated and reference posterior variance, to quantify the accuracy of recovering the variance and found that both approaches slightly overestimated the posterior variances (median dispersion MNLE: 1.12, median LAN: 1.16; Fig.3C and D). Overall, we found both MNLE and LAN posterior moments to be very close to the moments of the reference posteriors in absolute terms (median absolute mean difference MNLE: 0.02, LANs: 0.03; median absolute variance difference MNLE: 0.0003, LANs: 0.0002; Suppl. Fig.A.1C).

Finally, we used the classifier 2-sample test (C2ST, Lopez-Paz and Oquab, 2017; Lueckmann et al., 2021) to quantify the similarity between the estimated and reference posterior distributions. The C2ST is defined to be the error-rate of a classification algorithm which aims to classify whether samples belong to the true or the estimated posterior.

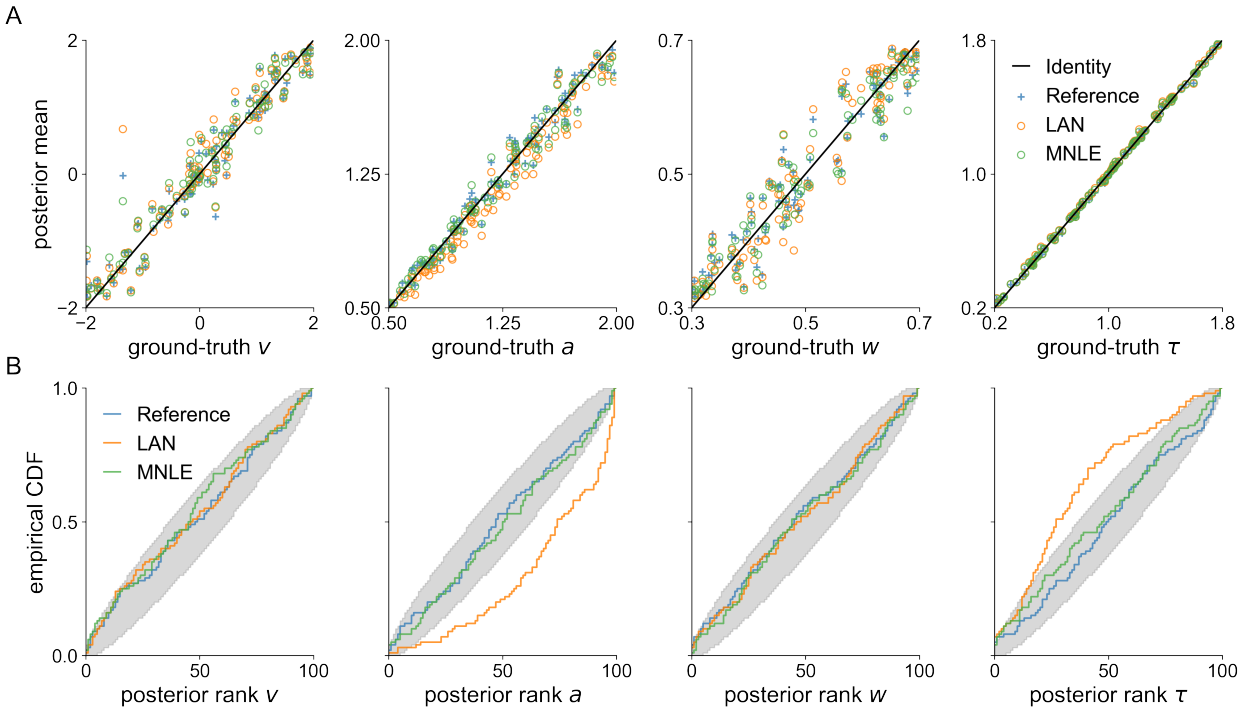

Figure 4: **Parameter recovery and posterior uncertainty calibration for the DDM. A)** Underlying ground-truth DDM parameters plotted against the sample mean of posterior samples inferred with the analytical likelihoods (reference, blue crosses), LAN (orange circles) and MNLE (green circles), for 100 different observations. Markers close to diagonal indicate good recovery of ground-truth parameters; circles on top of blue reference crosses indicate accurate posterior means. **B)** Simulation-based calibration results showing empirical cumulative density functions (CDF) of the ground-truth parameters ranked under the inferred posteriors calculated from 100 different observations. A well-calibrated posterior must have uniformly distributed ranks, as indicated by the area shaded gray. Shown for reference posteriors (blue), LAN posteriors (orange) and MNLE posterior (green), and for each parameters separately (v, a, w and τ).

Thus, it ranges from 0.5 (no difference between the distributions, the classifier is at chance level), to 1.0 (the classifier can perfectly distinguish the two distributions). We note that the C2ST is a highly sensitive measure of discrepancy between two multivariate-distributions—e.g. if the two distributions differ in *any* dimension, the C2ST will be close to 1 even if all other dimensions match perfectly. We found that neither of the two approaches was able to achieve perfect approximations, but that MNLE achieved lower C2ST scores (median C2ST score MNLEs: 0.64, LAN: 0.71, lower score on 81 out of 100 comparisons, $p < 10^{-9}$; Fig.3E). In summary, MNLE achieves more accurate recovery of posterior means, similar recovery of posterior variances, and overall more similar posteriors (as quantified by C2ST).

2.4 MNLE posteriors have uncertainties which are well-calibrated

For practical applications of inference, it is often desirable to know how well an inference procedure can recover the ground-truth parameters, and whether the uncertainty-estimates are well-calibrated, (Cook et al., 2006), i.e., that the *uncertainty* estimates of the posterior are balanced, and neither over-confident nor under-confident. For the DDM, we found that the posteriors inferred with MNLE and LANs recovered the ground-truth parameters accurately (in terms of posterior means, Fig.4A)– in fact, parameter recovery was similarly accurate to using the 'true' analytical likelihoods, indicating that much of the residual error is due to stochasticity of the observations, and not the inaccuracy of the likelihood approximations.

To assess posterior calibration we used simulation-based calibration (SBC, Talts et al., 2018). The basic idea of SBC is that, if one repeats the inference with many simulated observations and averages the corresponding posteriors, then, in a well-calibrated inference method, these should converge to the prior distribution, and be neither more broad nor more narrow. SBC results indicated that MNLE posteriors were as well-calibrated as the reference posteriors, i.e., the ranks of the ground-truth parameters under the inferred posteriors were uniformly distributed (Fig.4B)– thus, on this example, MNLE inferences would likely be of similar quality compared to using the analytical likelihoods. LANs, too,

appeared to recover most of the ground-truth parameters well, however, SBC detected a systematic underestimation of the parameter a and overestimation of the parameter τ (Fig.4B, see the deviation below and above the desired uniform distribution of ranks, respectively).

The results so far (i.e., Fig.3,4) indicate that both LANs and MNLE lead to similar parameter recovery, but only MNLE leads to posteriors which were well-calibrated for all parameters. These results were obtained using a scenario with 100 i.i.d. trials. When increasing the number of trials (here, to 1000 trials), posteriors become very concentrated around the ground-truth value. In that case, while the posteriors overall identified the ground-truth parameter value very well, even small deviations in the posteriors can have large effects on the posterior metrics (Suppl. Fig.A.1). This effect was also detected by SBC, showing systematic biases for some parameters (Suppl. Fig.A.2). For MNLE, we found that these biases were smaller, and furthermore that it was possible to mitigate this effect by inferring the posterior using ensembles, e.g., by combining samples inferred with five MNLEs trained with identical settings but different random initialization (see section A.5 for details). Overall, SBC provides a important tool for testing posterior coverage—we note that it would also be applicable in situations where the model in question has an intractable likelihood and reference posteriors are not available.

3 Discussion

Statistical inference for computational models in cognitive neuroscience can be challenging because models often do not have tractable likelihood functions. The recently proposed LAN-method (Fengler et al., 2021) performs SBI for a subset of such models (DDMs) by training neural networks with model simulations to approximate the intractable likelihood. However, LANs require large amounts of training data, restricting its usage to canonical models. We proposed an alternative approached called mixed neural likelihood estimation (MNLE), a synthetic neural likelihood method which is tailored to the data-types encountered in many models of decision-making.

Our comparison on a tractable example problem used in Fengler et al. (2021) showed that MNLE performed on par with LANs using six orders of magnitude fewer model simulations for training. While Fengler et al. (2021) discuss that LANs were not optimized for simulation efficiency and that it might be possible to reduce the required model simulations, we emphasize that the difference in simulation-efficiency is due to an inherent property of LANs: For each parameter in the training data, LANs require empirical likelihood targets that have to be estimated by building histograms or kernel density estimates from thousands of simulations. MNLE, instead, performs conditional density estimation without the need of likelihood targets and can work with only one simulation per parameter.

There exist a number of approaches with corresponding software packages for estimating parameters of cognitive neuroscience models, and DDMs in particular. However, these toolboxes either only estimate single best-fitting parameters (Voss and Voss, 2007; Wagenmakers et al., 2007; Chandrasekaran and Hawkins, 2019; Heathcote et al., 2019; Shinn et al., 2020) or, if they perform fully Bayesian inference, are restricted to variants of the DDM for which the likelihood can be evaluated (Wiecki et al., 2013, HDDM). While the restriction of the HDDM toolbox to likelihood-based inference could be lifted by adding LANs to it, the restriction to DDMs for which LANs can be pre-trained remains. In contrast, MNLE can in principle be applied to any simulation-based model with intractable likelihoods and mixed data output.

Several extensions to classical SL approaches have addressed the problem of a bias in the likelihood approximation due to the strong parametric assumptions, i.e., Gaussianity, the use of summary statistics, or finite-sample biases (Price et al., 2018; Ong et al., 2018; van Opheusden et al., 2020). MNLE builds on flexible neural likelihood estimators, e.g., normalizing flows, and does not require summary statistics for a low-dimensional simulator like the DDM, so would be less susceptible to these first two biases. It could be subject to biases resulting from the estimation of the log-likelihoods from a finite number of simulations. In our numerical experiments, and for the simulation-budgets we used, we did not observe biased inference results. We speculate that the ability of neural density estimators to pool data across multiple parameter settings (rather than using only data from a specific parameter set, like in classical synthetic likelihood methods) mitigates finite-sample effects.

MNLE is a SBI method which uses conditional density estimators to estimate likelihoods. Alternatives to neural likelihood estimation include neural posterior estimation (NPE, Papamakarios and Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019, which uses conditional density estimation to learn the posterior directly) and neural ratio estimation (NRE, Hermans et al., 2020, which uses classification to approximate the likelihood-to-evidence ratio to then use MCMC for inference). These approaches could in principle be applied here as well, however, they are not as well suited for the flexible inference scenarios common in decision-making models as MNLE: NPE by design does not allow for flexible inference scenarios but needs to be retrained because the posterior changes with changing number of trials or changing hierarchical inference setting, and NRE, performing ratio- and not density estimation, would not provide an emulator of the simulator. Regarding future research directions, MNLE has the potential to become more

simulation-efficient by using weight sharing between the discrete and the continuous neural density estimators (rather than to use separate neural networks, as we did here). Moreover, for high-dimensional inference problems in which MCMC might struggle, MNLE could be used in conjunction with variational inference as recently proposed by Wiqvist et al. (2021) and Anonymous (2022). Finally, using its emulator property, MNLE could be applied in an active learning setting for highly expensive simulators in which new simulations are chosen adaptively according to a acquisition function in a Bayesian optimization framework (Gutmann and Corander, 2016; Lueckmann et al., 2019; Järvenpää et al., 2019) .

In summary, MNLE enables flexible and efficient inference of parameters of models in cognitive neuroscience with intractable likelihoods. The training efficiency and flexibility of the neural density estimators used overcomes the limitations of LANs (Fengler et al., 2021). Critically, these features enables researchers to develop customized models of decision-making, not just apply existing models to new data. We implemented our approach as an extension to a public `sbi` python package to make it accessible for practitioners with detailed documentation and examples.

4 Methods

4.1 Mixed neural likelihood estimation

Mixed neural likelihood estimation (MNLE) extends the framework of neural likelihood estimation (Papamakarios et al., 2019b; Lueckmann et al., 2019) to be applicable to simulation-based models with mixed data types. It learns a parametric model $q_\psi(\mathbf{x}|\boldsymbol{\theta})$ of the intractable likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ defined implicitly by the simulation-based model. The parameters ψ are learned with training data $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ comprised of model parameters $\boldsymbol{\theta}_n$ and their corresponding data simulated from the model $\mathbf{x}_n|\boldsymbol{\theta}_n \sim p(\mathbf{x}|\boldsymbol{\theta}_n)$. The parameters are sampled from a proposal distribution over parameters $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta})$. The proposal distribution could be any distribution, but it determines the parameter regions for which the density estimator will be good in estimating likelihoods. Thus, the prior, or a distribution that contains the support of the prior (or even all priors which one expects to use in the future) constitutes a useful choice. After training, the emulator can be used to generate synthetic data $\mathbf{x}|\boldsymbol{\theta} \sim q_\psi(\mathbf{x}|\boldsymbol{\theta})$ given parameters and to evaluate the synthetic likelihood $q_\psi(\mathbf{x}|\boldsymbol{\theta})$ given experimentally observed data, so that posterior samples can be obtained via

$$p(\boldsymbol{\theta}|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}),$$

through approximate inference with MCMC. Importantly, the training is amortized, i.e., the emulator can be applied to new experimental data without retraining (running MCMC is still required).

We tailored MNLE to simulation-based models that return mixed data, e.g., in form of reaction times rt and (usually categorical) choices c as for the DDM. Conditional density estimation with normalizing flows has been proposed for continuous random variables (Papamakarios et al., 2019a), or discrete random variables (Tran et al., 2019), but not for mixed data. Our solution for estimating the likelihood of mixed data is to simply factorize the likelihood into continuous and discrete variables,

$$p(rt, c|\boldsymbol{\theta}) = p(rt|\boldsymbol{\theta}, c)\; p(c|\boldsymbol{\theta}),$$

and use two separate neural likelihood estimators: A discrete one q_{ψ_c} to estimate $p(c|\boldsymbol{\theta})$ and a continuous one $q_{\psi_{rt}}$ to estimate $p(r|\boldsymbol{\theta}, c)$. We defined q_{ψ_c} to be a Bernoulli model and use a neural network learn the Bernoulli probability ρ given parameters $\boldsymbol{\theta}$. For $q_{\psi_{rt}}$ we used a conditional neural spline flow (Durkan et al., 2019) to learn the density of rt given a parameter $\boldsymbol{\theta}$ and choice c. The two estimators are trained separately using the same training data (see section 4.5 for details on neural network architecture and training). After training, the full neural likelihood can be constructed by multiplying the likelihood estimates q_{ψ_c} and $q_{\psi_{rt}}$ back together:

$$q_{\psi_c,\psi_{rt}}(rt, c|\boldsymbol{\theta}) = q_{\psi_c}(c|\boldsymbol{\theta})\; q_{\psi_{rt}}(rt|c, \boldsymbol{\theta}).$$

Note that, as the second estimator $q_{\psi_{rt}}(r|c, \boldsymbol{\theta})$ is conditioned on the choice c, our likelihood-model can account for statistical dependencies between choices and reaction times. The neural likelihood can then be used to approximate the intractable likelihood defined by the simulator, e.g., for inference with MCMC. Additionally, it can be used to sample synthetic data given model parameters, without running the simulator (see section A.4).

4.2 Relation to LAN

Neural likelihood estimation can be much more simulation efficient than previous approaches because it exploits continuity across the parameters by making the density estimation conditional. Fengler et al. (2021), too, aim to exploit continuity across the parameter space by training a neural network to predict the value of the likelihood function from parameters $\boldsymbol{\theta}$ and data $\mathbf{x}$. However, the difference to neural likelihood estimation is that they do not use the

neural network for density estimation directly, but instead do classical neural network-based regression on likelihood targets. Crucially, the likelihood targets first have to obtained for each parameter in the training data set. To do so, one has to perform density estimation using KDE (as proposed by Turner et al., 2015) or empirical histograms for *every* parameter separately. Once trained, LANs do indeed exploit the continuity across the parameter space (they can predict log-likelihoods given unseen data and parameters), however, they do so at a very simulation high cost: For a training data set of N parameters, they perform N times KDE based on M simulations each[1], resulting is an overall simulation budget of $N \cdot M$ (N =1.5 million and M =100,000 for "pointwise" LAN approach).

4.3 Details of the numerical comparison

The comparison between MNLE and LAN is based on the drift-diffusion model (DDM). The DDM simulates a decision variable X as a stochastic differential equation with parameters $\boldsymbol{\theta} = (v, a, w, \tau)$:

$$\mathrm{d}X_{t+\tau} = v\mathrm{dt} + \mathrm{d}W, \quad X_\tau = w, \tag{1}$$

where W a Wiener noise process. The priors over the parameters are defined to be uniform: $v \sim \mathcal{U}(-2, 2)$ is the drift, $a \sim \mathcal{U}(0.5, 2)$ the boundary separation, $w \sim \mathcal{U}(0.3, 0.7)$ the initial offset, $\tau \sim \mathcal{U}(0.2, 1.8)$ the non-decision time. A single simulation from the model returns a choice $c \in \{0, 1\}$ and the corresponding reaction time in seconds $rt \in (\tau, \infty)$.

For this version of the DDM the likelihood of an observation (c, rt) given parameters $\boldsymbol{\theta}$, $l(c, rt|\boldsymbol{\theta})$, can be calculated analytically (Navarro and Fuss, 2009). To simulate the DDM and calculate analytical likelihoods we used the approach and the implementation proposed by Drugowitsch (2016). We numerically confirmed that the simulations and the analytical likelihoods match those obtained from the research code provided by Fengler et al. (2021).

To run LANs, we downloaded the neural network weights of the pre-trained models from the repository mentioned in Fengler et al. (2021) (to our knowledge, the implementation of LANs as part of the HDDM toolbox indicated in Fengler et al. (2021) has not been made publicly available yet). The budget of training simulations used for the LANs was 1.5×10^{11} (1.5 million training data points, each obtained from KDE based on 10^5 simulations). We only considered the approach based on training a multilayer-perceptron (MLP) on single-trial likelihoods ("pointwise approach", Fengler et al., 2021).

To run MNLE, we extended the implementation of neural likelihood estimation in the `sbi` toolbox (Tejero-Cantero et al., 2020). All comparisons were performed on a single AMD Ryzen Threadripper 1920X 12-Core processor with 2.2GHz. All the code is publicly available (see Section A.1).

4.4 Flexible Bayesian inference with MCMC

Once the MNLE is trained, it can be used for MCMC to obtain posterior samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathbf{x})$ given experimentally observed data $\mathbf{x}$. To sample from posteriors via MCMC, we transformed the parameters to unconstrained space, used slice sampling (Neal, 2003), and initialized ten parallel chains using sequential importance sampling (Papamakarios et al., 2019b), all as implemented in the `sbi` toolbox. We ran MCMC with identical settings for MNLE and LAN.

Importantly, performing MNLE and then using MCMC to obtain posterior samples allows for flexible inference scenarios because the form of $\mathbf{x}$ is not fixed. For example, when the model produces trial-based data that satisfy the i.i.d. assumption, e.g., a set of reaction times and choices $X = \{rt, c\}_{i=1}^N$ in a drift-diffusion model, then MNLE allows to perform inference given varying numbers of trials. This is achieved by training MNLE based on single-trial likelihoods (only once) and then combining multiple trials into the joint likelihood only when running MCMC:

$$p(\boldsymbol{\theta}|X) \propto \prod_{i=1}^{N} q(rt_i, c_i|\boldsymbol{\theta})\; p(\boldsymbol{\theta}).$$

Similarly, one can use the neural likelihood to perform hierarchical inference with MCMC, all without the need for retraining (see Hermans et al., 2020; Fengler et al., 2021, for examples).

4.5 Neural network architecture, training and hyperparameters

Architecture For the architecture of the Bernoulli model we chose a feed-forward neural network that takes parameters $\boldsymbol{\theta}$ as input and predicts the Bernoulli probability ρ of the corresponding choices. For the normalizing flow we used the neural spline flow architecture (NSF, Durkan et al., 2019). NSFs use a standard normal base distribution and transform

[1]For models with categorical output, i.e., all decision-making models, KDE is performed separately for each choice.

it using several modules of monotonic rational-quadratic splines whose parameters are learned by invertible neural networks. Using an unbounded base distribution for modeling data with bounded support, e.g., reaction time data $rt \in (0, \infty)$, can be challenging. To account for this, we tested two approaches: We either transformed the reaction time data to logarithmic space to obtain an unbounded support ($\log rt \in (-\infty, \infty)$), or we used a log-normal base distribution with rectified (instead of linear) tails for the splines (see Durkan et al., 2019, for details and section A.2 for the final architecture settings)

Training The neural network parameters ψ_c and ψ_{rt} were trained using the maximum likelihood loss and Adam (Kingma and Ba, 2015). As proposal distribution for the training dataset we used the prior over DDM parameters. Given a training data set of parameters, choices and reaction times $\{\boldsymbol{\theta}_i, (c_i, rt_i)\}_{i=1}^N$ with $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$; $(c_i, rt_i) \sim \text{DDM}(\boldsymbol{\theta}_i)$, we minimized the negative log-probability of the model, e.g., using the same training data, we trained the Bernoulli choice model by minimizing

$$-\frac{1}{N}\sum_{i=1}^{N} \log q_{\psi_c}(c_i|\boldsymbol{\theta}_i),$$

and the neural spline flow by minimizing

$$-\frac{1}{N}\sum_{i=1}^{N} \log q_{\psi_{rt}}(rt|c_i, \boldsymbol{\theta}_i).$$

Training was performed with code and training hyperparameter settings provided in the `sbi` toolbox (Tejero-Cantero et al., 2020).

Hyperparameters MNLE requires a number of hyperparameter choices regarding the neural network architectures, e.g., number of hidden layers, number of hidden units, number of stacked NSF transforms, kind of base distribution, among others (Durkan et al., 2019). With our implementation building on the `sbi` package we based our hyperparameter choices on the default settings provided there. This resulted in likelihood accuracy similar to LAN, but longer evaluation times due to the complexity of the underlying normalizing flow architecture.

To reduce evaluation time of MNLE, we further adapted the architecture to the example model (DDM). In particular, we run a cross-validation of the hyperparameters relevant for evaluation time, i.e., number of hidden layers, hidden units, NSF transforms, spline bins, and selected those that were optimal in terms of Huber loss and mean-squared error between the approximate and the *analytical* likelihoods, as well as evaluation time. This resulted in an architecture with performance *and* evaluation time similar to LANs (more details in appendix A.2). The cross-validation relied on access to the analytical likelihoods which is usually not given in practice, e.g., for simulators with intractable likelihoods. However, we note that in cases without access to analytical likelihoods a similar cross-validation can be performed using quality measures other than the difference to the analytical likelihood, e.g., by comparing the observed data with synthetic data and synthetic likelihoods provided by MNLE.

5 Acknowledgments

We thank Luigi Acerbi, Michael Deistler, and Ingeborg Wenger for discussions and comments on the manuscript. We also acknowledge and thank the Python (Van Rossum and Drake Jr, 1995) and Julia (Bezanson et al., 2017) communities for developing the tools enabling this work, including `DifferentialEquations.jl` (Rackauckas and Nie, 2017), `DiffModels.jl` (Drugowitsch, 2016), `NumPy` (Harris et al., 2020), `pandas` (pandas development team, 2020), `Pyro` (Bingham et al., 2019), `PyTorch` (Paszke et al., 2019), `sbi` (Tejero-Cantero et al., 2020), `sbibm` (Lueckmann et al., 2021) and `Scikit-learn` (Pedregosa et al., 2011).

This work was supported by the German Research Foundation (DFG; SFB 1233 PN 276693517; SPP 2041; Germany's Excellence Strategy MLCoE – EXC number 2064/1 PN 390727645), the German Federal Ministry of Education and Research (BMBF; project ADIMEM, FKZ 01IS18052 A-D; Tübingen AI Center, FKZ: 01IS18039A), and the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101030918 (to R.G.).

References

Ziwen An, Leah F South, David J Nott, and Christopher C Drovandi. Accelerating bayesian synthetic likelihood with the graphical lasso. *Journal of Computational and Graphical Statistics*, 28(2):471–475, 2019.

Anonymous. Variational methods for simulation-based inference. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=kZOUYdhqkNY. under review.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(1):973–978, 2019. ISSN 1532-4435.

Chandramouli Chandrasekaran and Guy E Hawkins. Chartr: An r toolbox for modeling choices and response times in decision-making tasks. *Journal of neuroscience methods*, 328:108432, 2019.

Patricia S Churchland and Terrence J Sejnowski. Perspectives on cognitive neuroscience. *Science*, 242(4879):741–745, 1988.

Samantha R Cook, Andrew Gelman, and Donald B Rubin. Validation of software for bayesian models using posterior quantiles. *Journal of Computational and Graphical Statistics*, 15(3):675–692, 2006.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020. doi: 10.1073/pnas.1912789117.

Christopher C Drovandi, Clara Grazian, Kerrie Mengersen, and Christian Robert. Approximating the likelihood in abc. *Handbook of approximate bayesian computation*, pages 321–368, 2018.

Jan Drugowitsch. Fast and accurate monte carlo sampling of first-passage times from wiener diffusion models. *Scientific reports*, 6(1):1–13, 2016.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 32:7511–7522, 2019.

Alexander Fengler, Lakshmi N Govindarajan, Tony Chen, and Michael J Frank. Likelihood approximation networks (lans) for fast inference of simulation models in cognitive neuroscience. *Elife*, 10:e65074, 2021.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2404–2414. PMLR, 2019.

Michael U Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *The Journal of Machine Learning Research*, 17(1):4256–4302, 2016.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825): 357–362, 2020.

Andrew Heathcote, Yi-Shin Lin, Angus Reynolds, Luke Strickland, Matthew Gretton, and Dora Matzke. Dynamic models of choice. *Behavior research methods*, 51(2):961–985, 2019.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free mcmc with approximate likelihood ratios. In *Proceedings of the 37th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR, 2020.

Marko Järvenpää, Michael U Gutmann, Arijus Pleska, Aki Vehtari, Pekka Marttinen, et al. Efficient acquisition rules for model-based approximate bayesian computation. *Bayesian Analysis*, 14(2):595–622, 2019.

Antti Kangasrääsiö, Jussi PP Jokinen, Antti Oulasvirta, Andrew Howes, and Samuel Kaski. Parameter inference for computational cognitive models with approximate bayesian computation. *Cognitive science*, 43(6):e12738, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015.

Hee Seung Lee, Shawn Betts, and John R Anderson. Learning problem-solving rules as search through a hypothesis space. *Cognitive science*, 40(5):1036–1079, 2016.

Michael D Lee. Three case studies in the bayesian analysis of cognitive models. *Psychonomic Bulletin & Review*, 15 (1):1–15, 2008.

Michael D Lee. How cognitive modeling can benefit from hierarchical bayesian models. *Journal of Mathematical Psychology*, 55(1):1–7, 2011.

Michael D Lee and Eric-Jan Wagenmakers. *Bayesian cognitive modeling: A practical course*. Cambridge university press, 2014.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *5th International Conference on Learning Representations, ICLR*, 2017.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 30, 2017.

Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. In *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96 of *Proceedings of Machine Learning Research*, pages 32–53. PMLR, 2019.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 343–351. PMLR, 13–15 Apr 2021.

James L McClelland. The place of modeling in cognitive science. *Topics in Cognitive Science*, 1(1):11–38, 2009.

Daniel J Navarro and Ian G Fuss. Fast and accurate calculations for first-passage times in wiener diffusion models. *Journal of mathematical psychology*, 53(4):222–230, 2009.

Radford M Neal. Slice sampling. *Annals of Statistics*, pages 705–741, 2003.

Victor MH Ong, David J Nott, Minh-Ngoc Tran, Scott A Sisson, and Christopher C Drovandi. Variational bayes with synthetic likelihood. *Statistics and Computing*, 28(4):971–988, 2018.

James J Palestro, Per B Sederberg, Adam F Osth, Trisha Van Zandt, and Brandon M Turner. *Likelihood-free methods for cognitive science*. Springer, 2018.

The pandas development team. pandas-dev/pandas: Pandas, February 2020. URL `https://doi.org/10.5281/zenodo.3509134`.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems 29*, pages 1028–1036. Curran Associates, Inc., 2016.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019a.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 837–848. PMLR, 2019b.

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

Umesh Patil, Sandra Hanne, Frank Burchert, Ria De Bleser, and Shravan Vasishth. A computational evaluation of sentence processing deficits in aphasia. *Cognitive Science*, 40(1):5–50, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Leah F Price, Christopher C Drovandi, Anthony Lee, and David J Nott. Bayesian synthetic likelihood. *Journal of Computational and Graphical Statistics*, 27(1):1–11, 2018.

Jacob W Priddle, Scott A Sisson, David T Frazier, Ian Turner, and Christopher Drovandi. Efficient bayesian synthetic likelihood with whitening transformations. *Journal of Computational and Graphical Statistics*, pages 1–14, 2021.

Christopher Rackauckas and Qing Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *The Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.151.

Roger Ratcliff and Gail McKoon. The diffusion decision model: theory and data for two-choice decision tasks. *Neural computation*, 20(4):873–922, 2008.

Roger Ratcliff and Jeffrey N Rouder. Modeling response times for two-choice decisions. *Psychological science*, 9(5): 347–356, 1998.

Andy M Reynolds and Christopher J Rhodes. The lévy flight paradigm: random search patterns and mechanisms. *Ecology*, 90(4):877–887, 2009.

Daniel J Schad, Michael Betancourt, and Shravan Vasishth. Toward a principled bayesian workflow in cognitive science. *Psychological methods*, 26(1):103, 2021.

Richard M Shiffrin, Michael D Lee, Woojae Kim, and Eric-Jan Wagenmakers. A survey of model evaluation approaches with a tutorial on hierarchical bayesian methods. *Cognitive Science*, 32(8):1248–1284, 2008.

Maxwell Shinn, Norman H Lam, and John D Murray. A flexible framework for simulating and fitting generalized drift-diffusion models. *ELife*, 9:e56938, 2020.

S. A. Sisson, Fan Y., and Beaumont M. A. Overview of abc. In *Handbook of Approximate Bayesian Computation*, chapter 1. CRC Press, Taylor & Francis Group, 2018.

Sean Talts, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*, 2018.

Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J. Gonçalves, David S. Greenberg, and Jakob H. Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020. doi: 10.21105/joss.02505. URL `https://doi.org/10.21105/joss.02505`.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *Advances in Neural Information Processing Systems*, 32:14719–14728, 2019.

Brandon M Turner and Trisha Van Zandt. A tutorial on approximate bayesian computation. *Journal of Mathematical Psychology*, 56(2):69–85, 2012.

Brandon M Turner and Trisha Van Zandt. Approximating bayesian inference through model simulation. *Trends in cognitive sciences*, 22(9):826–840, 2018.

Brandon M Turner, Leendert Van Maanen, and Birte U Forstmann. Informing cognitive abstractions through neuroimaging: the neural drift diffusion model. *Psychological review*, 122(2):312, 2015.

Marius Usher and James L McClelland. The time course of perceptual choice: the leaky, competing accumulator model. *Psychological review*, 108(3):550, 2001.

Bas van Opheusden, Luigi Acerbi, and Wei Ji Ma. Unbiased and efficient log-likelihood estimation with inverse binomial sampling. *PLoS computational biology*, 16(12):e1008483, 2020.

Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

Andreas Voss and Jochen Voss. Fast-dm: A free program for efficient diffusion model analysis. *Behavior research methods*, 39(4):767–775, 2007.

Eric-Jan Wagenmakers, Han LJ Van Der Maas, and Raoul PPP Grasman. An ez-diffusion model for response time and accuracy. *Psychonomic bulletin & review*, 14(1):3–22, 2007.

Thomas V Wiecki, Imri Sofer, and Michael J Frank. Hddm: Hierarchical bayesian estimation of the drift-diffusion model in python. *Frontiers in neuroinformatics*, 7:14, 2013.

Samuel Wiqvist, Jes Frellsen, and Umberto Picchini. Sequential neural posterior and likelihood approximation. *arXiv preprint arXiv:2102.06522*, 2021.

Simon N Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.

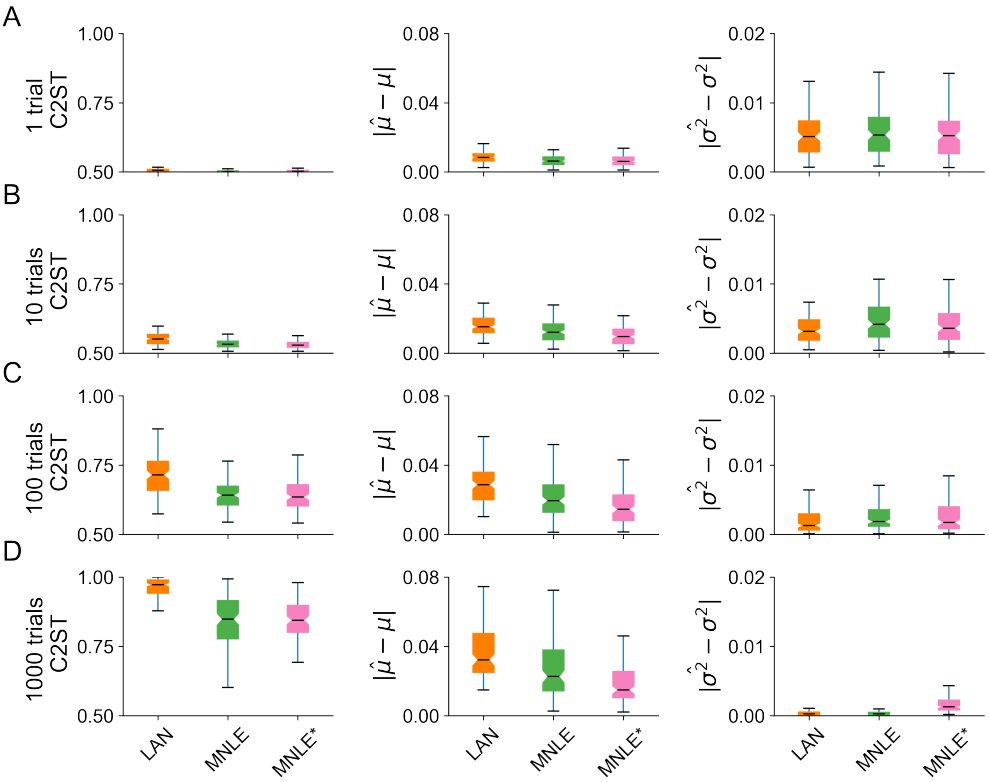

Figure A.1: **Posterior accuracy metrics for different numbers of observed trials.** Posterior metrics C2ST, absolute posterior mean error, and absolute error in posterior variance are shown for different number of observed trials in panels A, B, C, and D respectively. They were calculated from 10.000 posterior samples and with respect to the reference posterior, for LAN (orange), MNLE (green), and an ensemble of five MLNEs (MNLE*, purple).

A Appendix

A.1 Code availability

We implemented MNLE as part of the open source package for SBI, `sbi` at `https://github.com/mackelab/sbi`. Code for running MNLE and reproducing the results presented here can be found at `https://github.com/mackelab/mnle-for-ddms`.

A.2 Architecture and training hyperparameters

For the Bernoulli neural network we used three hidden layers with ten units each and sigmoid activation functions. For the neural spline flow architecture (Durkan et al., 2019) we transformed the reaction time data to the log-domain, used a standard normal base distribution, two spline transforms with five bins each and conditioning networks with three hidden layers and ten hidden units each, and rectified linear unit activation functions. The neural network training was performed using the *sbi* package with the following settings: learning rate 0.0005; training batch size 100; 10% of training data as validation data, stop training after 20 epochs without validation loss improvement.

A.3 Supplementary figures

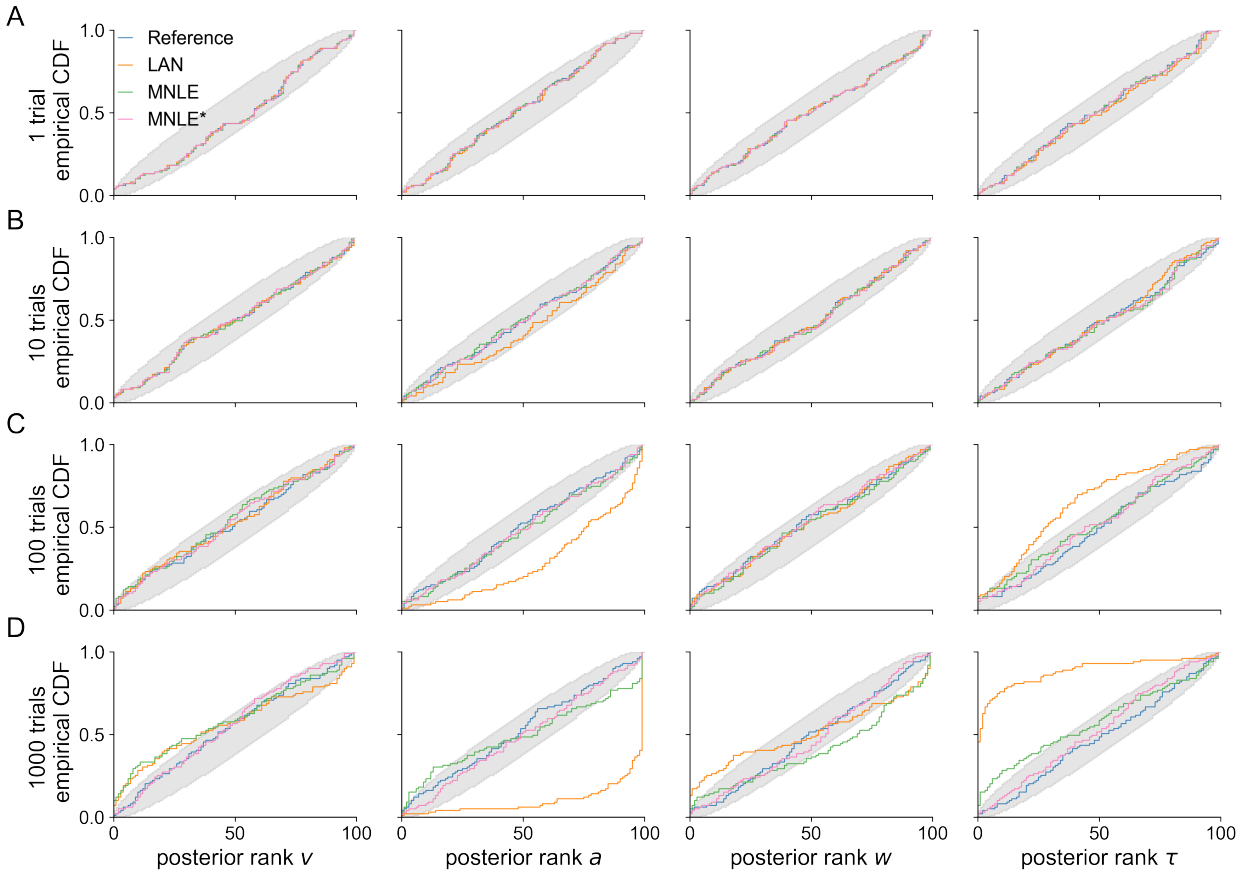

Figure A.2: **Simulation-based calibration results for different numbers of observed trials.** SBC results in form empirical conditional density functions of the ranks of ground-truth parameters under the approximate posterior samples. We compared posterior samples based on analytical likelihoods (blue), LAN (orange), MNLE (green), and an ensemble of five MLNEs (MNLE*, purple); for each of the four parameters of the DDM and for 1, 10, 100, 1000 observed trials (panel A, B, C, and D, respectively). Grey areas show random deviations expected under uniformly distributed ranks (ideal case).

A.4 MNLE provides an accurate emulator of the DDM

Being based on the neural likelihood estimation framework, MNLE naturally returns such an emulator of the simulator that can be sampled to generate synthetic data without running the simulator. We found that the synthetic data generated by MNLE accurately matched the data we obtained by running the DDM simulator (Fig.A.3). This has several potential benefits: it can help with evaluating the performance of the density estimator, it enables almost instantaneous data generation (one forward pass in the neural network) even if the simulator is computationally expensive, and it gives full access to the internals of the emulator, e.g., to gradients w.r.t. to data or parameters.

There is variant of the LAN approach which allows for sampling synthetic data as well: In the "Histogram-approach" (Fengler et al., 2021) LANs are trained with a convolutional neural network (CNN) architecture using likelihood targets in form of two-dimensional empirical histograms. The output of the CNN is a probability distribution over a discretized version of the data space which can, in principle, be sampled to generate synthetic DDM choices and reaction times. However, the accuracy of this emulator property of CNN-LANs is limited by the number of bins used to approximate the continuous data space (e.g., 512 bins for the examples shown in Fengler et al. (2021)).

A.5 Parameter recovery and posterior coverage

When testing methods for fitting parameters of computational models to experimental data, one common approach is to test their ability to recover ground-truth parameters from simulated data. For example, one would obtain posterior

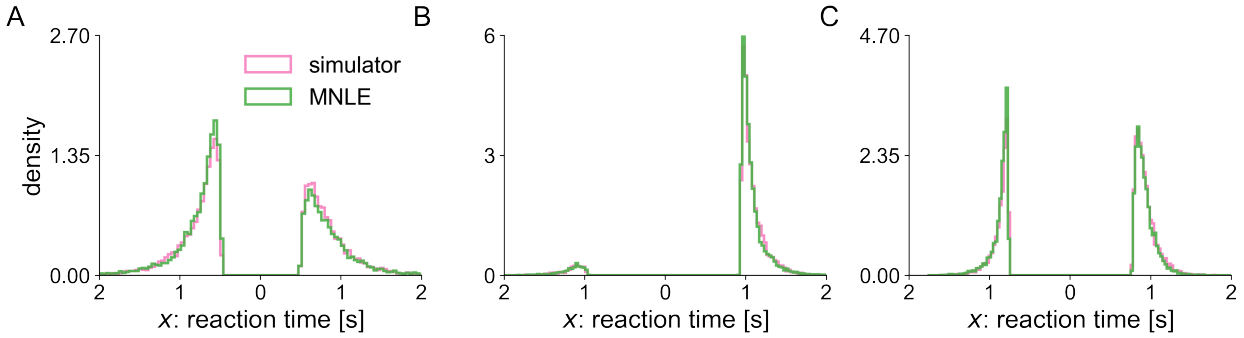

Figure A.3: **Comparison of simulated DDM data and synthetic data sampled from the MNLE emulator.** Histograms of reaction times from 1000 i.i.d. trials generated from three different parameters sampled from the prior (panel A, B, C) using the original DDM simulator (purple) and the emulator obtained from MNLE (green). “Down" choices are shown to the left of zero and “up" choices to the right of zero.

samples given simulated data for which one knows the ground-truth parameters, and then compare the mean or the mode of the posterior with the ground-truth parameter (see e.g. Fengler et al., 2021).

While it is true that for an increasing number of i.i.d. trials the posteriors should becomes narrower around the ground-truth parameters, it is not the case that the posterior is generally “centered” on the ground-truth parameters. When only little information is available, e.g., only a small number of trials was observed, the resulting posterior should reflect this uncertainty and should be rather broad. Additionally, irrespective of the variance of the posterior, when one repeats the inference for many different simulated data points, the positions of the ground-truth parameters in percentiles of the inferred posteriors should cover the entire range of percentiles (Talts et al., 2018). Thus, it is often more informative to check the coverage of the inferred posteriors instead of checking for parameter recovery by comparing posterior means with ground-truth parameters.

To demonstrate this, we compared parameter recovery and posterior coverage for inference in the DDM. In the main paper we ran inference with LANs and MNLE for 100 different 100-trial observations. Additionally, we tested parameter recovery and coverage for observed data comprised of 1, 10, and 1000 trials. For parameter recovery, we calculated sample means from 10.000 MCMC samples obtained using LANs and MNLE and compared them to the underlying true parameters.

We found that single-trial observations, the large uncertainty in the parameter estimate was reflected in large posterior variance so that posterior means did not recover ground-truth parameters (Fig.A.4A, note that MNLE and LAN posterior means actually matched the reference posterior means). For ten or more observed trials both methods give good recovery performance showing high correlation between ground-truth parameters and sample means (Fig.A.4, $R > 0.99$).

Additionally, we used simulation-based calibration (SBC, Talts et al., 2018) to check the calibration properties of LANs and MNLE. According to SBC, it is a necessary condition for a valid inference method that samples from the data-averaged posterior are distributed according to the prior (or equivalently that the ground-truth parameters ranked under the approximated posterior have uniform ranks). Our results showed that the posteriors obtained from analytical likelihoods were well calibrated (as expected). For MNLE and LAN we found that posteriors were well calibrated for small numbers of trials (<100), while larger number of trials some systematic biases occur: For MNLE, the drift parameter v which tends to be over-estimated, and for LAN, posteriors are less well calibrated for the boundary parameter a (under-estimation) and the non-decision time τ (over-estimation), confirming the visual tendencies in parameter recovery from above (Fig.A.4; see Talts et al. (2018) section 4.2 for a guide on visual interpretation of SBC results).

We also calculate posterior metrics for all four numbers of trials (1,10,100, and 1000) and found that posterior accuracy metrics also decrease for increasing number of observed i.i.d. trials (Fig.A.1). However, we note that the posteriors given large number of trials tend to have very small variances so that metrics like C2ST and dispersion show decreased performance where the absolute error is in fact relatively small (Fig.A.1D).

There might be two possible explanation for the systematic biases for larger trial counts detected by SBC (Fig.A.2). First, for large trial counts the posteriors tend to have very low variance such that small deviations in the approximate posteriors lead to large effects in the SBC analysis and the other inference quality metrics. Second, both methods are trained to predict single-trial likelihoods. As a consequence, small errors in the the likelihood predictions can amplify

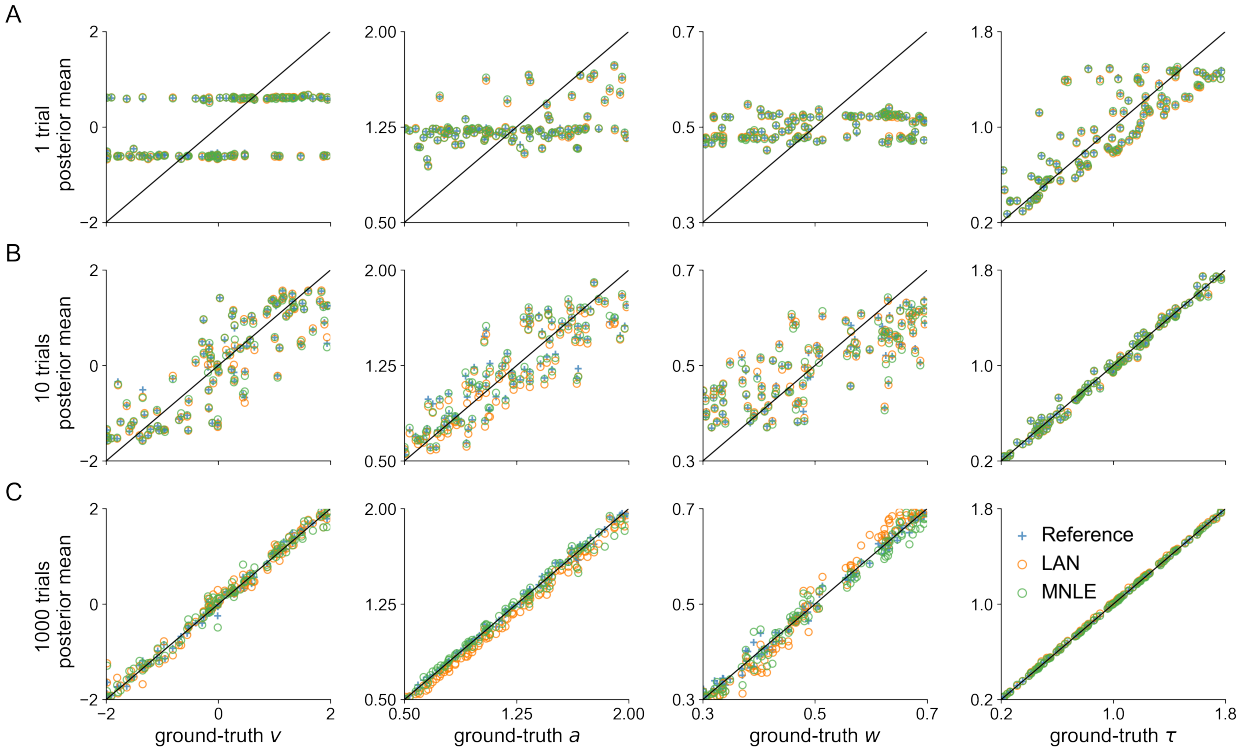

Figure A.4: **Parameter recovery for different number of i.i.d. DDM trials.** True underlying DDM parameters plotted against posterior sample means for 1, 10, and 1000 of observed i.i.d. trial(s) (in rows) and for the four DDM parameters v, a, w and τ (in columns). Calculated from 10.000 posterior samples obtained with MCMC using the reference (blue), LAN (orange) and the MNLE (green) likelihoods. Black line shows the identity function indicating perfect recovery.

when they are multiplied over all observed trials. In line with this hypothesis we observed that posteriors inferred with an ensemble of five MNLEs (MNLE*, colored purple in Fig A.2) show better calibration, C2ST score and bias.