Encoder-Decoder Neural Architectures for Fast Amortized Inference of Cognitive Process Models

Alexander Fengler (alexander_fengler@brown.edu)
Department of Cognitive, Linguistic and Psychological Sciences, 190 Thayer St
Providence, RI 02912 USA

Lakshmi Narasimhan Govindarajan (lakshmi_govindarajan@brown.edu)
Department of Cognitive, Linguistic and Psychological Sciences, 190 Thayer St
Providence, RI 02912 USA

Michael J. Frank (michael_frank@brown.edu)
Department of Cognitive, Linguistic and Psychological Sciences
Carney Institute for Brain Science
190 Thayer St
Providence, RI 02912 USA

Abstract

Computational cognitive modeling offers a principled interpretation of the functional demands of cognitive systems and affords quantitative fits to behavioral/brain data. Typically, cognitive modelers are interested in the fit of a model with parameters estimated using maximum likelihood or Bayesian methods. However, the set of models with known likelihoods is dramatically smaller than the set of plausible generative models. For all but some standard models (e.g., the drift-diffusion model), lack of closed-form likelihoods typically prevents using traditional Bayesian inference methods. Employing likelihood-free methods is a workaround in practice. However, the computational complexity of these methods is a bottleneck, since it requires many simulations for each proposed parameter set in posterior sampling schemes. Here, we propose a method that learns an approximate likelihood over the parameter space of interest by encapsulation into a convolutional neural network, affording fast parallel posterior sampling downstream after a one-off simulation cost is incurred for training.

Keywords: Likelihood-free inference; Approximate Bayesian Computation; ABC; Importance Sampling; Bayesian Inference; Neural Networks; GPU; Parallel Computing; Cognitive Process Models.

Introduction

Computational modeling has gained traction in cognitive neuroscience in part because it can guide principled interpretations of functional demands of cognitive systems and offers tractable quantitative fits of brain-behavior relationships. Bayesian parameter estimation allows one to infer posterior distributions over likely parameters (instead of just point estimates), including their uncertainty, while also assessing how variations in one parameter alter inference over another.

Traditionally, posterior sampling for such models has demanded analytical likelihood functions: given a model and a set of parameters, the likelihood of any data point needs to be analytically solvable. This constraint limits the application of Bayesian analysis to a relatively small subset of cognitive models chosen for so-defined convenience instead of theoretical interest. Consequently, model comparison exercises are hampered, as many plausible likelihood-free models were effectively *untestable*. During the last decade and a half, however, algorithms have prominently developed which allow posterior sampling over parameters of models that are defined only as simulators (Sisson, Fan, & Beaumont, 2018). Early applications of these methods can be found in Biology and Ecology, notably as an approach towards posterior inference, for the famous Lotka-Volterra predator-prey model (Beaumont, 2010). These approximate Bayesian computation (ABC) methods now enjoy successful application across the sciences, for example, in physics (Akeret, Refregier, Amara, Seehars, & Hasner, 2015), and have crucially found their way into cognitive science (B. Turner & Van Zandt, 2018).

Traditional ABC approaches rely on summary statistics of the data produced by simulations. The distance between these summary statistics and summary statistics computed from empirical observations are then compared with help of a distance function $d(S_{obs}, S_{sim})$. Two shortcomings result from such a starting point. First, ABC approaches generally require many simulations and hence can be computationally expensive (in some cases prohibitive). Second, aside from the fact that the *right* low dimensional summary is usually unknown, using low dimensional summaries of simulated data as approximate sufficient statistics restricts expressiveness and usually entails information loss with respect to model parameters. Worse, the appropriate multi-dimensional distance function $d(x, y)$ is a hyperparameter of choice, which is a priori unknown. In choosing this hyperparameter, a failure to take into account potential non-linear relationships between the summary statistics can lead to geometric distortion of the posterior distribution.

In the context of computational cognitive modeling, B. Turner and Sederberg (2014) developed an ABC (ABC-KDE in the following) method that is independent of summary statistics and instead relies on kernel density approximations of likelihood functions produced from model simulations. The benefit of this approach is that, while approximate with respect to the likelihood function, it does not

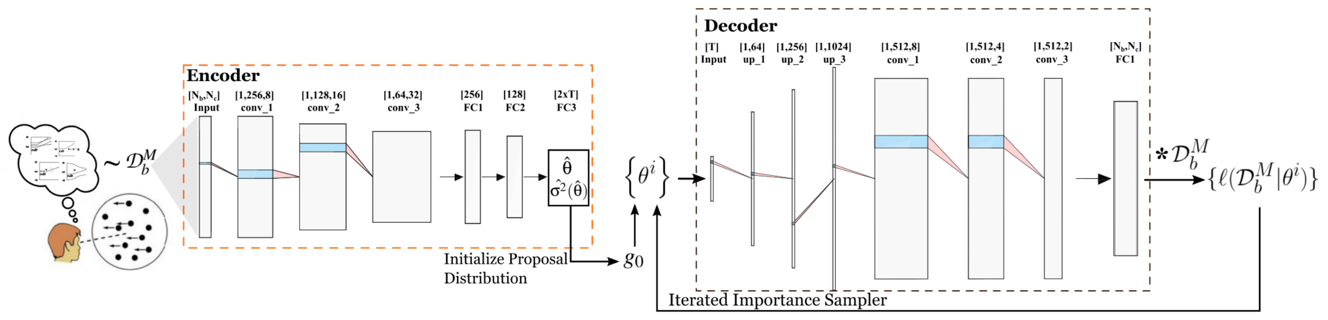


Figure 1: Illustration of a use case of our Encoder-Decoder based Importance sampling algorithm for estimating posteriors on the generative parameters of a stochastic model. Both the Encoder and Decoder are instantiated as CNNs. *conv_x*, *up_x*, *FC_x* refer to convolutional, upsampling and fully-connected layers. Symbol notations adhere to definitions in the main text.

make any obvious compromise on the posterior geometry concerning parameter trade-offs. What this approach does not overcome, however, is the reliance on an excessive amount of model simulations at inference, exacerbated by the sequential nature of the Markov chain Monte Carlo (MCMC) methods applied for posterior inference. Even recent refinements (Holmes, 2015) do not overcome this issue.

The basic idea of the ABC-KDE method has opened doors (N. J. Evans, Trueblood, & Holmes, 2019), but the excessive computational burden has hampered widespread application. For example, typically tens of thousands of simulations are needed to generate a realistic empirical likelihood for a single set of parameters (B. Turner & Sederberg, 2014). These empirical likelihood functions are then used in a Markov Chain (DE-MCMC is popular, (Ter Braak, 2006)) which itself needs to be run for tens of thousands of steps. The simulation runs are then simply discarded. Here we propose the use of neural networks for amortized inference, which aims at progressive reuse of previously incurred computations.

Similar ideas have directly inspired two recent proposals. Mestdagh, Verdonck, Meers, and Tuerlinckx (2019) propose an infrastructure for databases of pre-computed summary statistics. While such an initiative could substantially facilitate inference for popular models, any ABC method operating over such a database will suffer from the shortcomings concerning operation on summary statistics. Radev, Mertens, Voss, and Koethe (2019) offers an ambitious end-to-end approach to inference. The essence of their approach lies in the utilization of fully convolutional neural networks to learn approximate multivariate normal posterior distributions directly from data. Specifically, Radev et al. (2019) adopt a heteroscedastic loss function to estimate anisotropic variances in their parameter estimates. This approach is rapid for obtaining approximate maximum likelihood estimates and impressive due to its aspiration towards full encapsulation of posterior inference problems in a simple computational step. While this is a laudable goal, it suffers from two main shortcomings. First, the primary purpose of full posterior inference is to become aware of parameter relationships, for which posterior *covariances* are indispensable. Radev et al. (2019) however only consider (an estimate of) marginal posterior variances; independence between parameters is implicitly assumed. Because covariances are not estimated, the resulting marginal variances are potentially distorted when parameters are not independent (which is nearly always the case for cognitive models). A second limitation is that the posterior variances are only strictly interpretable as such if the network has been trained on the same dataset sizes which is used during inference. Third, the learned posterior inference is specific to a single inference scenario. Consequently, the method cannot easily be extended to critical application scenarios like hierarchical estimation, or scenarios in which a subset of parameters is assumed fixed while others vary across experimental conditions (or continuously as a function of neural regressors).

Here, we propose an end-to-end algorithm that combines the primary use of empirical likelihood distributions with the idea of amortized inference using neural networks and fast but honest posterior sampling, without reliance on summary statistics. Our approach achieves approximately a 100-1000 fold speed increase when compared to the application of ABC-KDE methods for equivalent problems.

Section 2 introduces the components of our analysis pipeline with an emphasis on the generality of the approach. In section 3, we provide details on data generation and network training. Section 4 presents the results of our computational experiments. In section 5, we end with a discussion of our approach, possible extensions, and options for deployment to the community.

Methods

Two main desiderata guided our algorithm development. First, we wanted to maximally reuse computation incurred by model simulations by learning a transformation that can compute a joint likelihood from large data-sets quickly. Second, we strive to exploit parallel computing infrastructure to enable fast posterior sampling across the parameter space. Together these guiding principles should produce a procedure that massively speeds up posterior sampling while only paying an initial simulation cost once.

Concerning principle one, in brief, we will use convolutional neural networks (CNNs) to approximate discretized global likelihood functions (called *Decoder or Reverse Networks* below). Evaluation of the likelihood of a dataset $\mathcal{D}$ under a parameter set θ can be computed as the sum of elementwise multiplication of the log probability of each discrete outcome with the number of observed cases. We note that the cost of computing the likelihood is now independent of the size of the dataset.

Generally, one can then perform posterior inference via MCMC, replacing the (analytical or KDE) likelihood function by evaluating a CNN for parameter inputs. While some preliminary success with this approach (Fengler & Frank, 2019) has been shown previously, because MCMC is inherently sequential, it misses out on the potential to exploit the power of parallel computing infrastructure. Thus as per our second desiderata, we aim for a parallel sampling scheme, for which we chose a minimally revised version of iterated importance sampling based on adaptable Gaussian mixture proposals (Cappé, Douc, Guillin, Marin, & Robert, 2007; Wraith et al., 2009). Such a scheme affords the potential to leverage parallel batch processing in neural networks, as explained later. Importance sampling relies on the idea that for any probability distribution of interest f, we may produce samples from f by using an *importance weighted resampling* of samples from another (the *proposal*) distribution g. The method is driven by the basic equality,

$$\int_{\Omega} f(x)dx = \int_{\Omega} \frac{f(x)}{g(x)} g(x)dx$$

A sample $\mathbf{x} = \{x_0, ..., x_n\} \sim g$, gets assigned weights $\mathbf{w} = \{w_o = \frac{f(x_0)}{g(x_0)}, ... w_n = \frac{f(x_n)}{g(x_n)}\}$. Multinomial resampling from $\mathbf{x}$, according to $\mathbf{w}$, produces an approximation to sampling from f directly. Importance sampling is exact in the limit, however finite sample approximation error is strongly dependent on how closely g matches f to begin with. Iterative importance sampling adds the idea of improving the distribution g over time to reduce approximation error introduced by having finite samples (consistency of importance sampling relies on infinite samples).

However, an important issue for importance sampling is that the initial proposal distribution must be reasonable to kick-start the iterative update process successfully. That is, the sequence of proposal distributions $\{g_i\}_{i=0}^{N}$ is crucially dependent on getting the very first proposal g_0 roughly correct. Precisely, g_0 needs to cover regions of high likelihood concerning the target of interest, and should do so in a way that is somewhat, but not excessively, overdispersed as compared to the target, where the target in our case is a posterior distribution.

For this purpose, we obtain an initial estimate of the posterior distribution by slightly modifying and re-purposing recent neural network approaches to end-to-end approximate posterior inference (Radev et al., 2019; Kendall & Gal, 2017; Jiang, Wu, Zheng, & Wong, 2017). Convenient for us, the approach of Radev et al. (2019) produces overdispersed approximate posteriors (in part because of an embedded assumption of the posterior being isotropic Gaussian), which enables us to utilize it as an "Encoder network" to initialize our importance sampling scheme. The complete processing pipeline and algorithm is pictorially illustrated in Fig. 1, leaving us with the *three* main components: the *Encoder Network*, the *Decoder Network* and the *Importance sampler*, each of which is elaborated below.

Test-bed

Our method applies to models that have a reasonably constrained output (data) space D_{out}, such that binning the output space down to approximately $500 \times dim(D_{out})$ maintains a manageable discretization-error.

As an initial test-bed, we choose some variations of a prominent class of cognitive process models known as sequential sampling models (SSMs) or evidence accumulation models (EAMs): The most widely known example being the *ratcliff drift-diffusion model* (DDM). This class of models is frequently used across domains in the experimental cognitive and neuroscience literature, to jointly model choice and response time distribution data. The underlying parameters are, in turn, linked to neural processes (Ratcliff, Smith, Brown, & McKoon, 2016; N. Evans & Wagenmakers, 2019; Frank et al., 2015; B. M. Turner, van Maanen, & Forstmann, 2015). The links can be explored via hierarchical models parameter in which neural processes connect to parameters in cognitive models via e.g., a regression function. To illustrate the potential of our method, we choose four specific variants of such EAMs that are frequently used in research but for which analytic likelihood functions are not available: *(1)* A *simple DDM model* in which the decision bound collapses according to the CDF of a *Weibull distribution*, *(2)* the *Full-DDM model (with inter-trial variability in 3 parameters)* (Ratcliff & McKoon, 2008), *(3)* the *Ornstein-Uhlenbeck* model (as in the leaky competing accumulator or LCA) and *(4)* a *Four Choice Race Model*.

The data-generating process of all four models relies on an underlying diffusion, a *Brownian Motion W* with a drift term v. All models have a parameter that expresses the starting point bias $w \in [0, 1]$; a non-decision time $ndt \in [0, NDT_{max}]$ capturing time taken for perception and motor output; and an initial distance to one (or more) decision bound(s), labeled a. Abstractly, the driving diffusion for all our models can be described by a *stochastic differential equation* of the form,

$$dX_t = A(t, X_t)dt + dW_t$$

We make precise the specific discriminatory attributes of the four models below.
The *Ornstein-Uhlenbeck model* adds a position dependency expressed in an excitation/inhibition parameter g. The *Full-DDM* instead treats the starting point w, the drift v and

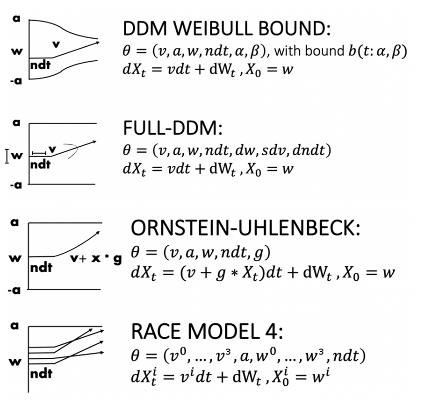

a
w
ndt
-a
v
DDM WEIBULL BOUND:
$\theta = (v, a, w, ndt, \alpha, \beta)$, with bound $b(t: \alpha, \beta)$
$dX_t = vdt + dW_t, X_0 = w$
FULL-DDM:
$\theta = (v, a, w, ndt, dw, sdv, dndt)$
$dX_t = vdt + dW_t, X_0 = w$
v+ x * g
ORNSTEIN-UHLENBECK:
$\theta = (v, a, w, ndt, g)$
$dX_t = (v + g * X_t)dt + dW_t, X_0 = w$
RACE MODEL 4:
$\theta = (v^0, \dots, v^3, a, w^0, \dots, w^3, ndt)$
$dX_t^i = v^i dt + dW_t, X_0^i = w^i$

Figure 2: Graphic illustration of the stochastic forward models used as a test bed for posterior inference.

the non-decision time *ndt* as trial-by-trial random variables emerging from a *uniform*, *normal*, and again *uniform* distribution respectively (adding the $dw, sdv, dndt$ parameters). The *DDM-Weibull* model, adds a parameterized boundary function to the model. The bound is defined as,

$$b(t|\alpha, \beta, a) = a * \exp -(\frac{t}{\beta})^{\alpha}.$$

Finally the *Four Choice Race Model*, in contrast to the three previous models, treats each decision option as a separate particle replicating the drift and starting point parameters four times. Decisions in this model correspond to the time-point at which the *first* of the four particles hits the boundary a. Fig. 2, gives a graphic depiction of the models.

Implementation Details

Synthetic data Generation We follow a uniform approach to universal data generation for all models presented. For each model, we generate $N = 3M$ sets of parameters drawn from uniform distributions. For each set of parameters Θ^M, we then generate $100K$ simulations from the stochastic model M and bin the output into b bins for each discrete choice option c, where the bins divided the $t \in [0, T_{max} = 10s]$ evenly up to the last bin which collected any observations $t > T_{max}$. The so-generated binned simulations were stored in a data-array D. For all numerical experiments we choose $b = 256$. For each model mentioned above, we therefore generate a data-set $\mathcal{D}_b^M = \{\theta_i, D_i\}_{i=0}^{3M}$, upon which training of the below mentioned networks was based.

Decoder (Reverse) Model The decoder network encapsulates the essence of the relevant stochastic model under consideration. It maps the vector of proposed generative parameters Θ to the likelihood, i.e. $P(\mathcal{D}_b^M|\Theta)$. By implementing the decoder as a CNN we gain two crucial properties. First, we can leverage batch processing of input and generate likelihoods for a large number of input parameter sets in parallel. Second, the output of the CNN is a global approximation of the likelihood for a single parameter set. Evaluation for a given data-set amounts to element-wise multiplication of the CNN output layer with bin-wise event counts. Speed of valuation is therefore independent of dataset-size. Parallelizability is also crucial for the employment of the Iterated importance sampler.

The network is trained by minimizing the *KL-divergence* between observed and generated data histograms (defined in Eq. 1). To create a higher dimensional embedding of the parameter θ, we employ a sequence of upsampling layers followed by convolutions with 1×5 kernels and a final fully connected layer resulting in an output of dimensionality $[N_b, N_c]$.

$$\mathcal{D}(\hat{P}(x|\Theta) \| P(x|\Theta)) = \sum_{i=0}^{b} \left[\hat{P}(x_i|\Theta) \; log \frac{\hat{P}(x_i|\Theta)}{P(x_i|\Theta)} \right] \quad (1)$$

Encoder (Forward) Model The encoder network, as described in Fig. 1, serves to provide a good initialization for our Importance Sampling procedure. The network maps a data set $\mathcal{D}_b^M$ (in our case a likelihood from a stochastic model M discretized into b number of bins) to the generative parameters of M, $\Theta = \{\theta_i : i \in [0, T]\}$. The readout node contains the predicted means and variances of the stochastic model parameters. As per Radev et al. (2019) training is accomplished via minimization of the *heteroscedastic* loss function defined in Eq. 2.

$$\mathcal{L}(\hat{\theta}, \hat{\sigma^2}(\hat{\theta}), \theta) = \frac{1}{T} \sum_{t=0}^{T-1} \left[\frac{(\theta_t - \hat{\theta}_t)^2}{2\hat{\sigma}_t^2(\hat{\theta})} + \frac{1}{2} log(\hat{\sigma}_t^2(\hat{\theta})) \right] \quad (2)$$

For both encoder and decoder networks, we use the adaptive learning rate based gradient descent algorithm Adam (Kingma & Ba, 2015), initialized with a learning rate of $1e-4$. We implemented both the encoder and decoder models in Tensorflow (Abadi et al., 2015).

Importance sampling Given a data-set of observations D^{obs}, we use iterative adaptive importance sampling based on Gaussian mixtures. Adjustments of the proposal distribution across iterations utilize the mixture update equations as derived in (Cappé et al., 2007; Wraith et al., 2009). The basic idea behind iterative importance sampling is to use an adaptive importance distribution, which upon good initialization, will converge to the target sufficiently to allow for efficient importance sampling at the last iteration.

As suggested by Cappé et al. (2007), to monitor convergence, we use the normalized perplexity statistic, defined at every iteration k as the exponential of the Shannon entropy of normalized importance weights divided by the total sample size, $\exp(H^{k,N})/N$, where $H^{k,N} = -\sum_{i=1}^{N} \bar{\omega}_{k,i} \log \bar{\omega}_{k,i}$. If the proposal distribution converges to the target, this statistic converges to 1. It is crucial to generate an initial proposal that is sufficiently close to the target to allow successful adaptation over time. We generate the initial mixture distribution g_0 in the following way. We apply the *encoder network* to D^{obs}

to get a point estimate for the generative parameters $\hat{\theta}^{obs}$, as well as parameter-wise variance terms $\hat{\sigma}^2_{\theta}$. The first mixture component, g_0^1, is then initialized as the isotropic multivariate Gaussian implied by the *encoder network* output. To generate the rest of the mixture components, we use the variance estimates $\hat{\sigma}^2(\hat{\theta}^{obs})$ gleaned from the *Encoder Model* and apply component-wise random perturbations from the centers of g_0^1. We reapply $\hat{\sigma}^2(\hat{\theta}^{obs})$, to generate a set of dispersed isotropic Gaussians around the g_0^1 as mixture components $g_0^2, ..., g_0^M$. In principle, other procedures for initialization are possible which do not themselves involve neural networks. One alternative is to replace the encoder network with another optimization technique such as Maximum Likelihood Estimation (MLE). The flip side of using MLE however, is that it yields point estimates. While, in principle, one can use a numerically approximated Hessian matrix around the MLE solution to initialize the covariance structure of our mixture distributions, this method is potentially unstable, especially if the MLE procedure does not, in fact, find a local minimum (the MLE might for example not sit in a locally convex cone).

To help with convergence, we use an annealing factor $\gamma = 2^{z-t}$, $z \in 1, 2, 4, ..., N$, where z is a parameter of the sampler. Hence, the first iteration operates on a target $f(x)^{\frac{1}{\gamma}}$. We decrease from γ_0 by dividing by 2 for each iteration that improves the complexity measure $\exp(H^{k,N})/N$, until we reach $\gamma_i = 1$. The importance sampler thus has three *hyperparameters*. The annealing factor z, the number of mixture components M and the number of samples we draw at each iteration i, N_i.

Results

Parameter Recovery Study Every dataset used in our parameter recovery studies consisted of $N = 1024$ data points. Model parameters that governed each dataset are chosen uniformly at random across the valid parameter range, albeit slightly truncated to lie within the parameter ranges used for network training. We note that model identifiability can deteriorate parameter recovery as defined here, even if sampling from the posterior was entirely successful. For this reason, we separately show posterior predictive distributions as well. As can be seen in Fig 3, these posterior predictive plots produce highly accurate results. Moreover, our covariance plots show that we can capture such relationships between parameters in the posterior distribution. Fig 3, summarizes the results of the parameter recovery study.

Speed and Efficiency of Sampling We report some statistics concerning the speed and efficiency of the importance sampler. Generally, across all models considered, the importance sampler converges within 20 iterations. At each iteration, we draw $200K$ importance samples; hence a total of up to $4M$ samples is run within approximately $300s$. As a rough estimate, this implies that we are processing one sample in approximately $75\mu s$. Since we intend here first and foremost to compare to traditional ABC methods, a reasonable benchmark would be to consider the time it takes to produce and evaluate, at point of inference, the empirical likelihood that is embedded in our *decoder network*. This implies drawing on the order of at least $30K$ (B. Turner & Sederberg, 2014) simulations for each proposed set of parameters in a posterior sampling scheme. To get a precise estimates of the time savings our method promises, we repeated 5000 runs of $30K$ simulations each from a *Full DDM* model. We used highly optimized simulators for this purpose, and ran this experiment on a top of the line 2019 MacBook Pro Laptop. For each run, the set of parameters was sampled uniformly from the domain our CNNs were trained on. The average time for $30K$ simulations was estimated at $600ms$. Treating computation and evaluation of the KDE for all samples as free, this already implies a speedup of approximately 4 orders of magnitude that our method promises.

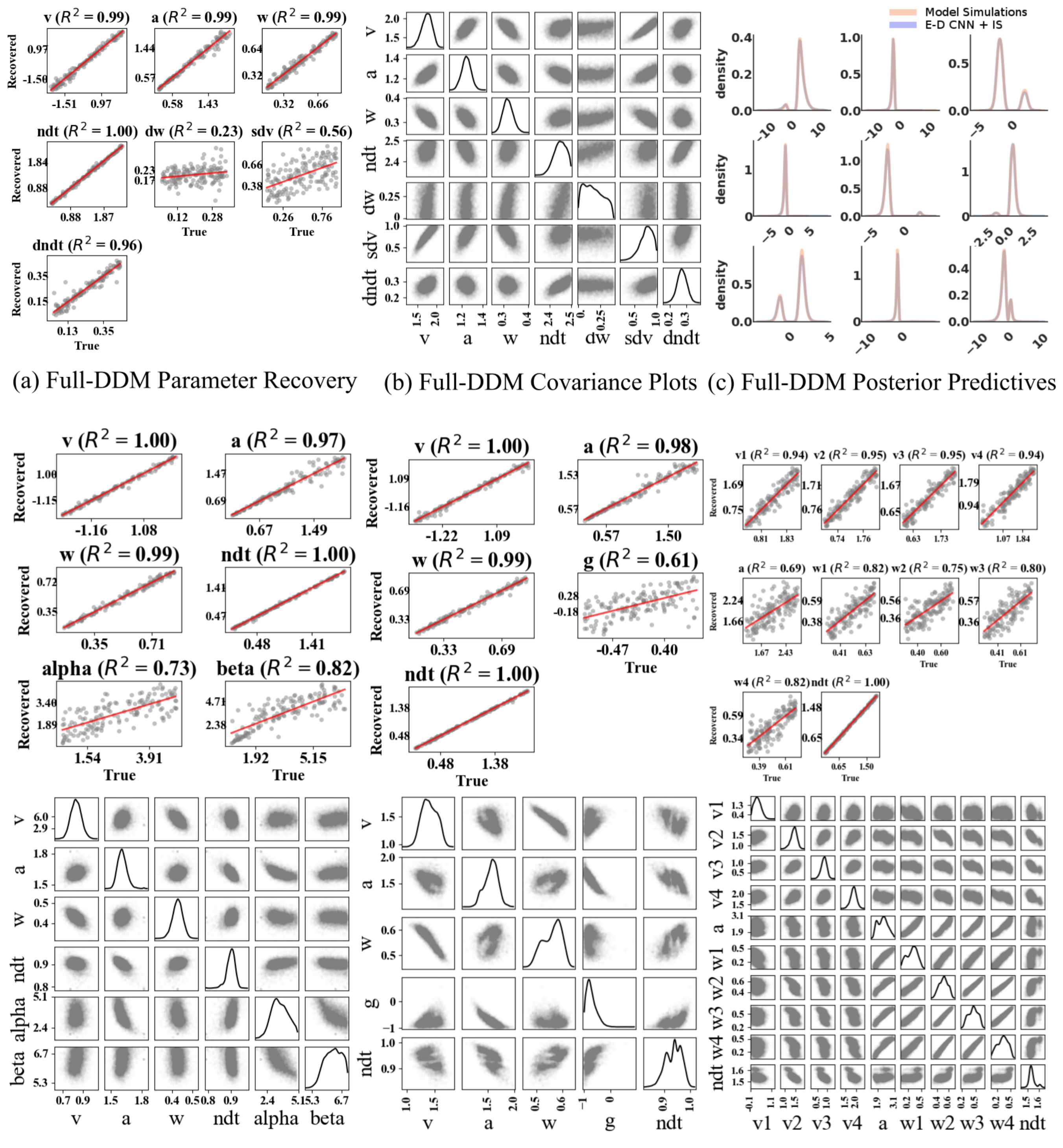
Model	N_{eff}	Run time (in mins)
Weibull	114K± 31.3K	7.7± 4.5
Full DDM	109K± 19.6K	6.5± 2.8
Ornstein–Uhlenbeck	139K± 24K	3.8± 2.2
Race Model 4	112K± 30K	9 ± 3.9

Table 1: Importance sampler statistics. N_{eff} is the effective sample size defined as $\frac{1}{\Sigma_{i=1}^N \omega_i^2}$. Run time here refers to the total time consumed until importance sampler convergence.

Discussion and Future Work

We present an end-to-end pipeline for amortized Bayesian inference. As a test bed, we consider evidence accumulation models of varying complexity. Our method crucially relied on two guiding principles. *(1)* The ability to reuse computations resulting from forward model simulations to the greatest extent possible, in other words maximally amortize our inference cost. We do so by learning from model simulations a global likelihood approximation encapsulated in a CNN that allows us to go from parameters directly to first passage likelihood distributions in one forward pass. *(2)* We want to do so in a way that allows us to to leverage advances in parallel computing infrastructure for rapid inference algorithms. For this purpose we utilize the capacity for parallel batch-processing to power extremely fast sampling rounds in an annealed iterated importance sampling approach to posterior sampling.

We show parameter recovery results and posterior predictive checks for four different evidence accumulation models, all notoriously difficult to estimate with traditional methods. Application across models shows that the algorithm is quite robust, with fine-tuning necessary only at the stage of importance sampling, where the choice of the number of components and an annealing factor γ can have an impact on the performance and convergence. We emphasize that we employed only minimal to no fine-tuning overall, and importantly used consistent sampler settings for all numerical experiments shown. While the number of choice options af-



(a) Full-DDM Parameter Recovery (b) Full-DDM Covariance Plots (c) Full-DDM Posterior Predictives

(d) Weibull (e) Ornstein (f) Race Model (4 Choices)

Figure 3: For each of the four models we show *parameter recovery* and an example *covariance plot* for a single randomly sampled parameter set. In addition, for the Full-DDM model we also show 9 *posterior predictive plots* from randomly sampled parameter sets.

fected the speed of inference (by virtue of its impact on the network architecture, since the last layer needs to accommodate a larger number of (t, c) states), we remain well below reported inference times that apply to the ABC-KDE methods for equivalent models. For all models shown, we reliably get good/sufficient convergence of the importance sampling distribution to the target. Marginal and covariance plots (Fig. 3) show that we can capture complex posterior shapes implied by non-linear parameter trade-offs.

There are multiple ways in which we plan to extend and build on the work presented. First, a major goal is to make the discussed procedures available to other researchers as a Python package. We aim to cover two basic use cases: On the one hand, users will be able to use their (or a cloud-based) GPU infrastructure. On the other hand, we will exploit the newly emerging dedicated hardware for low-cost, fast inference with pre-trained neural networks, (e.g., neural compute stick, Intel ®), to allow users an extremely cheap ($\sim$ 100$) entry point for utilizing the methods presented here, even on a personal budget laptop machine. Moreover, users will be able to upload their simulators, from which the encoder and decoder networks can be automatically trained. Second, exploiting the literature on *model distillation* (Hinton, Vinyals, & Dean, 2015), we aim to incorporate architecture minimization into our training pipeline, to ensure maximal speed at inference. Lastly, we aim to generalize the current approach to allow for hierarchical estimation of parameters.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Akeret, J., Refregier, A., Amara, A., Seehars, S., & Hasner, C. (2015). Approximate bayesian computation for forward modeling in cosmology. *Journal of Cosmology and Astroparticle Physics*.

Beaumont, M. A. (2010, December). Approximate Bayesian Computation in Evolution and Ecology. *Annual Review of Ecology, Evolution, and Systematics*, *41*(1), 379–406.

Cappé, O., Douc, R., Guillin, A., Marin, J.-M., & Robert, C. P. (2007). Adaptive importance sampling in general mixture classes. Statistics and Computing 18, 4 (2008) 447-459. doi: 10.1007/s11222-008-9059-x

Evans, N., & Wagenmakers, E. (2019). Evidence accumulation models: Current limitations and future directions. In *psyarxiv preprint.* Retrieved from `https://doi.org/10.31234/osf.io/74df9`

Evans, N. J., Trueblood, J., & Holmes, R. (2019). A parameter recovery assessment of time-variant models of decision-making. *Behavior Research Methods*.

Fengler, A., & Frank, M. (2019). Abc-nn: Approximate bayesian computation with neural networks to learn likelihood functions for efficient parameter estimation.. Retrieved from `https://doi.org/10.32470/CCN.2019.1361-0`

Frank, M. J., Gagne, C., Nyhus, E., Masters, S., Wiecki, T. V., Cavanagh, J. F., & Badre, D. (2015). fmri and eeg predictors of dynamic decision parameters during human reinforcement learning. *Journal of Neuroscience*, *35*(2), 485–494. Retrieved from `https://www.jneurosci.org/content/35/2/485` doi: 10.1523/JNEUROSCI.2036-14.2015

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. In *arxiv preprint.* Retrieved from `https://arxiv.org/abs/1503.02531`

Holmes, R. (2015). A practical guide to the probability density approximation (pda) with improved implementation and error characterization. *Journal of Mathematical Psychology*, *68*, 13-24.

Jiang, B., Wu, T., Zheng, C., & Wong, W. (2017). Learning summary statistics for approximate bayesian computation via deep neural networks. *Statistica Sinica*, *27*, 1595 - 1618.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision. In *arxiv preprint.* Retrieved from `https://arxiv.org/abs/1703.04977`

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Iclr conference paper.* ICLR. Retrieved from `https://arxiv.org/abs/1412.6980`

Mestdagh, M., Verdonck, S., Meers, K., & Tuerlinckx, F. (2019). Prepaid parameter estimation without likelihoods. *PLoS Computational Biology*, *15*(9).

Radev, S., Mertens, U., Voss, A., & Koethe, U. (2019). Towards end-to-end likelihood-free inference with convolutional neural networks. *British Journal of Mathematical and Statistical Psychology*, 23-43.

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, *20*(4), 873 - 922.

Ratcliff, R., Smith, P., Brown, S., & McKoon, G. (2016). Diffusion decision model: Current issue and history. *Trends in Cognitive Sciences*, *20*(4), 260–281.

Sisson, S., Fan, Y., & Beaumont, M. (2018). *Handbook of approximate bayesian computation*. CRP Press.

Ter Braak, C. J. (2006). A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces. *Statistical Computing*, *16*, 239-249.

Turner, B., & Sederberg, P. (2014). A generalized, likelihood-free method for posterior estimation. *Psychological Bulletin*, *21*(2), 227–250.

Turner, B., & Van Zandt, T. (2018). Approximating bayesian inference through model simulation. *Trends in Cognitive Sciences*, *22*(9), 826–840.

Turner, B. M., van Maanen, L., & Forstmann, B. U. (2015). Informing cognitive abstractions through neuroimaging: the neural drift diffusion model. *Psychological Review*, *122*(2), 312 - 336.

Wraith, D., Kilbinger, M., Benabed, K., Cappé, O., Cardoso, J.-F., Fort, G., ... Robert, C. P. (2009). Estimation of cosmological parameters using adaptive importance sampling. *Physical Review*. doi: 10.1103/PhysRevD.80.023507