ADAVI: Automatic Dual Amortized Variational Inference Applied To Pyramidal Bayesian Models

Louis Rouillard, Demian Wassermann

ADAVI: Automatic Dual Amortized Variational Inference Applied To Pyramidal Bayesian Models

Louis Rouillard
Université Paris-Saclay, Inria, CEA
Palaiseau, 91120, France
louis.rouillard-odera@inria.fr

Demian Wassermann
Université Paris-Saclay, Inria, CEA
Palaiseau, 91120, France
demian.wassermann@inria.fr

Abstract

Frequently, population studies feature pyramidally-organized data represented using Hierarchical Bayesian Models (HBM) enriched with plates. These models can become prohibitively large in settings such as neuroimaging, where a sample is composed of a functional MRI signal measured on 64 thousand brain locations, across 4 measurement sessions, and at least tens of subjects. Even a reduced example on a specific cortical region of 300 brain locations features around 1 million parameters, hampering the usage of modern density estimation techniques such as Simulation-Based Inference (SBI). To infer parameter posterior distributions in this challenging class of problems, we designed a novel methodology that automatically produces a variational family dual to a target HBM. This variatonal family, represented as a neural network, consists in the combination of an attention-based hierarchical encoder feeding summary statistics to a set of normalizing flows. Our automatically-derived neural network exploits exchangeability in the plate-enriched HBM and factorizes its parameter space. The resulting architecture reduces by orders of magnitude its parameterization with respect to that of a typical SBI representation, while maintaining expressivity. Our method performs inference on the specified HBM in an amortized setup: once trained, it can readily be applied to a new data sample to compute the parameters' full posterior. We demonstrate the capability of our method on simulated data, as well as a challenging high-dimensional brain parcellation experiment. We also open up several questions that lie at the intersection between SBI techniques and structured Variational Inference.

1 Introduction

Inference aims at obtaining the posterior distribution $p(\theta|X)$ of latent model parameters θ given the observed data X. In setups such as neuroimaging, featuring Hierarchical Bayesian Models (HBM) representing large population studies [4; 21], the dimensionality of θ can go over the million. This dimensionality hinders the usage of modern, normalizing flows based techniques [14; 20; 26; 27]. Indeed, the parametrization of the these techniques usually scales quadratically with the size of the parameter space [e.g. 9; 13; 25]. This in turn can lead to very complex, problem-specific derivations: for instance Kong et al. [21] rely on a manually-derived Expectation Maximization (EM) technique. This analytical complexity constitutes a strong barrier to entry, and limits the wide and fruitful usage of Bayesian modelling in that field. Our first aim is to meet that experimental need: how can we derive a technique both automatic and efficient in the context of very large, hierarchically-organised data?

We take inspiration from the field of Variational Inference (VI) [3; 41], in which such an analytical barrier to entry is also prominent. In VI, the experimenter posits a variational family $\mathcal{Q}$ so as to

Preprint. Under review.

approximate $q(\theta) \approx p(\theta|X)$. In practice, deriving an expressive, yet computationally attractive variational family can be challenging [3]. This triggered a trend towards the derivation of automatic VI techniques readily applicable to a problem [1; 22; 29]. We follow that logic and present a methodology that automatically derives a variational family $\mathcal{Q}$.

Contrary to traditional VI, we aim at deriving a variational family $\mathcal{Q}$ in the context of amortized inference [7; 31]. This means that, once an initial training overhead has been "paid" for, our technique could readily be applied to a new data point. Amortized inference is an active area of research in the context of Variational Auto Encoders (VAE) [17; 19; 32; 39]. It is also a natural possibility for normalizing flows (NF) [see 28; 31, close to our work], our technology of choice. We note the focus of modern inference techniques in that field is rather, placed on the derivation of *sequential* estimation, refining a posteriors (and loosing amortization) across several rounds of simulation [7; 14; 27; 34]. With all those VAE and NF-based methods, we share the idea of encoding observed data through a neural network to condition the density estimation of a stochastic block.

Yet, due to the very large parameter spaces presented above, our target applications aren't amenable to the generic black-box techniques described in Cranmer et al. [7]. We therefore differentiate ourselves in our will to exploit the structure of the problem not only through the design of an adapted encoder, but down to the very architecture of our density estimator. More specifically, we focus on the inference problem for Hierarchical Bayesian Models (HBMs) [11]. The idea to condition the architecture of a density estimator by an analysis of the dependency structure of an HBM has been studied in [37; 38], in the form of the masking of a single normalizing flow. To our knowledge, we introduce the first architecture that proposes instead to hierarchically combine separate flows. Considering a different field, our static analysis of a generative model can also be associated with structured VI [1; 16; 30]. But though we share the same philosophy, our working principles are rather orthogonal: structured VI usually aims at exploiting model structure to augment the expressivity of a variational family, whereas we aim at reducing its parametrization.

Our objective is therefore to derive an automatic methodology that takes as input a generative HBM and generates a *dual* variational family able to perform amortized parameter inference. This variational famimy exploits the exchangeability in the HBM to reduce its parametrization by orders of magnitude compared to generic methods [14; 27]. Consequently, our method can be applied in the context of large, pyramidally-structured data, a challenging setup in which generic methods would fail. Our general scheme is visible in fig. 1, a figure that we will explain thorough the course of the next section [1].

2 Methods

2.1 Pyramidal Bayesian Models

We are interested in practical experimental setups, such as population studies, modelled using HBMs [4; 21]. These models feature independent sampling from a common conditional distribution at multiple levels, translating the statistical notion of *exchangeability* [11], and the graphical notion of *plates* [12]. Our methodology aims at taking advantage of plates in HBMs. We further refine the sub-class of models we specialize upon as *pyramidal* models, represented by Direct Acyclic Graphs (DAG) with vertices $\mathcal{V}$, edges $\mathcal{E}$, and plates $\mathcal{P}$:

$$\begin{aligned} \mathcal{V} &= \{\theta_i\}_{i=0\dots L} & \mathcal{P} &= \{\mathbf{P}_p\}_{p=0\dots P} \\ \mathcal{E} &\subseteq \mathcal{V}\times\mathcal{V} & \mathrm{Card} &= \{\mathbf{P}_p \to \#\mathbf{P}_p\}_{p=0\dots P} \\ \mathrm{plates} &= \{\theta_i \to (\mathbf{P}_p,\dots\mathbf{P}_q)\}_{(i,p\dots q)\text{ where i belongs to plates } p\dots q} \end{aligned} \quad (1)$$

where $\mathrm{Card}(\mathbf{P})$ represents the number of exchangeable elements in the plate $\mathbf{P}$. Pyramidal graphs are DAG with the 3 following differentiating properties: first, we consider a stacking of the plates $\mathbf{P}_0,\dots,\mathbf{P}_P$, where an decreasing index describes further subdivisions of the nodes. This means that a plate $\mathbf{P}_p$ is included in every plate $\{\mathbf{P}_q\}_{q>p}$. Equivalently, if the vertex θ_i belongs to the plate $\mathbf{P}_p$, then it must belong to the plates $\{\mathbf{P}_q\}_{q>p}$; second, each plate $\mathbf{P}_p$ is associated to a fixed number of draws $\mathrm{Card}(\mathbf{P}_p)$; third, a vertex θ_i can only have its parents belonging to plate of larger rank than his. The obtained graph follows a typical pyramidal structure, as seen in Figure 1.

[1] code available in https://github.com/NeuroLang/adavi

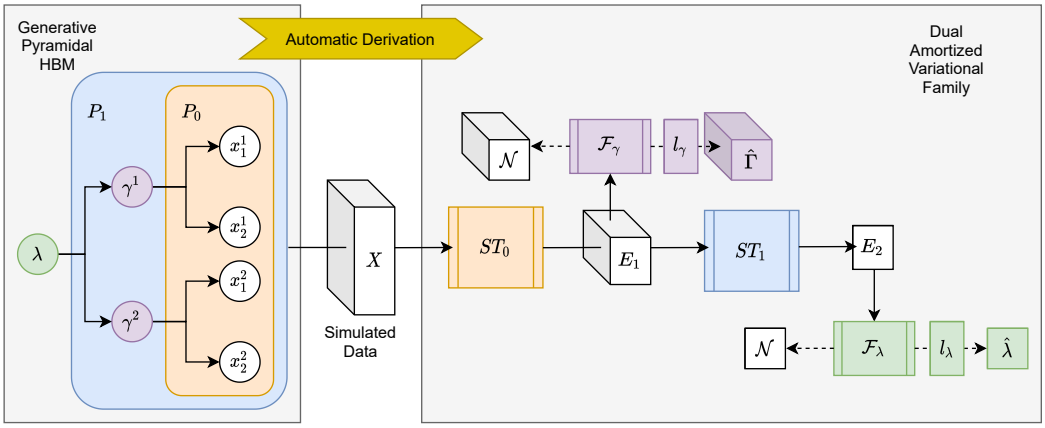

Figure 1: Automatic Dual Amortized Variational Inference (ADAVI) working principle. On the left is a generative HBM. It features 2 latent RV λ and $\Gamma = [\gamma^1, \gamma^2]$, the observed $X = [[x_1^1, x_2^1], [x_1^2, x_2^2]]$, and 2 plates P_0, P_1 of cardinality 2. We analyse automatically the structure of the HBM to produce its *dual* amortized variational family (on the right).Notice how the resulting architecture processes data X through 2 successive set transformers to produce encodings E aggregating summary statistics at different hierarchies. Those encodings are then used to condition normalizing flows producing the variational distributions for each latent RV.

This class of graphs is sufficiently expressive to represent many practical examples (see 8 schools and medical survey examples in Gelman et al. [11]). Our methodology is designed to process generative models whose dependency structure follow a pyramidal graph. As such, we cast our observed data X as a realization of the RV θ_0, that we expect to belong to the plate $\mathbf{P}_0$. The RVs $\{\theta_i\}_{i=1\dots L}$ therefore constitute the latent parameters on which we wish to perform inference.

A Pyramidal Bayesian Model can be summarized by the following descriptors:

$$\begin{aligned} &\text{Hier} = \{\theta_i \mapsto h_i = \min_p\{p \ : \ \mathbf{P}_p \in \text{plates}(\theta_i)\}\}_{i=0\dots L} \quad \text{Shape} = \{\theta_i \mapsto \mathcal{S}_i^e\}_{i=0\dots L} \\ &\text{Link} = \{\theta_i \mapsto (l_i : S_i^l \to S_i^e)\}_{i=0\dots L}, \end{aligned} \tag{2}$$

where: Hier describes the *hierarchy* h of an RV θ, i.e. the smallest rank for the plates it belongs to, or equivalently the level of the pyramidal graph it is placed at (X is of hierarchy 0); Shape maps a RV to its *event shape* $\mathcal{S}^e$, the latter potentially being high rank; Link maps a RV to its *event space link function* l [6]. l is a diffeomorphism between the constrained, potentially high-rank event space $\mathbb{Z}^{\mathcal{S}^e}$ in which a RV lives and a flattened unbounded $\mathbb{R}^{S^l}$ space. Note that we potentially have $\prod_{\text{all dimensions}}(\mathcal{S}^l) \neq \prod_{\text{all dimensions}}(\mathcal{S}^e)$. For instance, $\mathbb{Z}^{\mathcal{S}^e}$ is the unit sphere for a Von Mises-Fisher distribution and the simplex for a Dirichlet distribution).

In addition to the the random variables $\mathcal{V}$, the plates $\mathcal{P}$ and the plate cardinalities Card defined in eq. (1), these descriptors are the only inputs necessary for our derivation. Those descriptors can be readily obtained from a static analysis of a generative model, especially when the latter is expressed in a modern probabilistic programming framework [2; 8].

2.2 Function mapping and parameter sharing

We first introduce the notations for function mapping, a concept that we leverage thorough our entire architecture to reduce its parametrization. Given a function $f : F \to G$, we notate the *mapping* of f over a tensor of shape $(D_1 \times \dots \times D_K)$ of values in F as $\overrightarrow{f}^{D_1 \times \dots \times D_K}$.

Our architecture is based upon the mapping of functions across plates to reduce its parametrization. Considering that every independent draw from a plate consists in an equivalent problem, we train a common function to solve this common problem. Normalizing-flow based density estimators for a parameters space of size D have a parametrization in $\mathcal{O}(D^2)$ [e.g. 9; 13; 25]. For a RV θ, of batch shape $\text{Card}(\mathbf{P}_P) \times \dots \times \text{Card}(\mathbf{P}_h) \times S^e$, this would result in $\mathcal{O}(\text{Card}(\mathbf{P}_P)^2 \times \dots \times \text{Card}(\mathbf{P}_h)^2 \times S^{e^2})$ parameters. But since we train a common density estimator on the event space of shape S^e and

map its usage across all plates $\mathbf{P}_P, \ldots, \mathbf{P}_h$, we effectively reduce this parametrization to $\mathcal{O}(S^{e\,2})$. This is a fundamental property of our approach, leveraged through the rest of this section.

2.3 Automatic derivation of a dual amortized variational family

In this section, we derive our main methodological contribution. We aim at obtaining posterior distributions for a generative model described using equations 2. For this purpose, we construct a family of variational distribution $\mathcal{Q}$ *dual* to the model. This architecture consists in the combination of 2 items. First, a Hierachical Encoder that aggregates summary statistics from the data, exploiting the exchangeability property introduced by plates. Second, a set of conditional density estimators.

For our encoder, our goal is to learn a function HE that takes as an input the observed data X and successively exploits the permutation invariance across plates $\mathbf{P}_0, \ldots \mathbf{P}_P$. In doing so, HE produces encodings E at different hierarchy levels. To build HE, we leverage *SetTransformers* [24]: an attention-based architecture based on the working principle of *DeepSets* [40]. We use *SetTransformers* to derive encodings *across* a given plate, *mapping* them on all larger-rank plates. Specifically we define, for our observed data, the initial encoding E_0 as:

$$e_0 = \prod_{\text{all dimensions}} S_X^l \qquad E_0 = l_0^{-1}(X) \in \mathbb{R}^{\text{Card}(\mathbf{P}_P)\times\ldots\times\text{Card}(\mathbf{P}_0)\times e_0} \tag{3}$$

and an iterative encoding for every plate p, $p = 1 \ldots P$, using the set transformer ST_{p-1} as:

$$E_p = \overrightarrow{\text{ST}}_{p-1}^{\text{Card}(\mathbf{P}_P)\times\ldots\times\text{Card}(\mathbf{P}_p)}(E_{p-1}) \in \mathbb{R}^{\text{Card}(\mathbf{P}_P)\times\ldots\times\text{Card}(\mathbf{P}_p)\times e_p} \tag{4}$$

Note that the application of the set transformer ST_{p-1} produces a contraction of plate $\mathbf{P}_{p-1}$. We apply the mapping operation defined in section 2.2 on ST_{p-1}. As a consequence, the contraction operation will be performed in *parallel* for all components exchangeable across a plate of higher rank than $p-1$.

We now will use the encodings E, gathering hierarchical summary statistics on the data X, to condition the inference on the parameters θ. An encoding will condition the inference for the parameters sharing its hierarchy. Put formally, the encodings $\{E_p\}_{p=1\ldots P-1}$ will respectively condition the density estimators for the posterior distribution of parameters $\{\{\theta_i : h_i = p\}\}_{p=1\ldots P-1}$.

A conditional density estimator is a 2-step flow from a latent space onto the event space in which a RV lives. We represent every RV as a standard normal distribution in the latent space $\mathbb{R}^{S^l}$. First, this latent distribution is reparametrized by a conditional *normalizing flow* $\mathcal{F}$ into a distribution of more complex density in the space $\mathbb{R}^{S^l}$.Second, the obtained latent distribution is projected onto the event space by the application of the link function l.

The flow $\mathcal{F}$ is a diffeomorphism in $\text{Diff}(\mathbb{R}^{S^l})$ conditioned by the encoding E_h. $\mathcal{F}$ could be more or less expressive, opening up the possibility of sophisticated posteriors when leveraging the normalizing flow literature [20; 26]. Note that the normalizing flow $\mathcal{F}$ is *mapped* (see section 2.2) on plates of larger rank than the hierarchy of θ.

Put more formally, for every RV θ_i we have:

$$\begin{aligned}
&\mathcal{B}_i = \text{Card}(\mathbf{P}_P) \times \ldots \times \text{Card}(\mathbf{P}_{h_i}) && \text{NFlow} : \mathbb{R}^{e_{h_i}} \to \text{Diff}(\mathbb{R}^{S_i^l}) \\
&\mathcal{S}_i^b = \mathcal{B}_i \times S_i^l && \mathcal{F}_i(x; E_{h_i}) = \overrightarrow{\text{NFlow}}^{\mathcal{B}_i}(E_{h_i})(x) \\
&u_i \sim \mathcal{N}\left(\vec{0}_{\mathcal{S}_i^b}, \text{Id}_{S_i^b}\right) && \tilde{\theta}_i = l_i(\mathcal{F}_i(u_i; E_{h_i})) \sim q_i(\theta_i; E_{h_i})
\end{aligned} \tag{5}$$

Where q_i is the conditional density estimator for the posterior distribution $p(\theta_i|X)$. A visualization of the complete architecture can be seen in Figure 1.

2.4 Variational distribution and training

Given the encodings E_p provided by the hierarchical encoder HE, and the resulting conditioned density estimators q_i, we define our parametric amortized variational distribution as a *mean field approximation* [3] as follow:

$$\begin{aligned}
(E_1, \ldots E_P) &= \text{HE}_\chi(X) \\
q_{\chi,\Phi}(\theta|X) = q_\Phi(\theta; \text{HE}_\chi(X)) &= \prod_{i=1\ldots L} q_i(\theta_i; E_{h_i}, \Phi)
\end{aligned} \tag{6}$$

Where we can group parameters as $\Psi = (\chi, \Phi)$. Our objective is to have $q_\Psi(\theta|X) \approx p(\theta|X)$. Following the terminology introduced in Papamakarios et al. [26], we obtained the best experimental results using a *reverse* KL divergence (see comparative experiments in supplemental material). We therefore benefit from our access to a target joint density $p(X, \theta)$. The reverse KL loss is an amortized version of the classical ELBO expression [3]. To train our full architecture, we only need to have access to a dataset $\{X^m\}_{m=1\dots M}$ of points drawn from the generative HBM of interest. Indeed, the θ^m points are sampled from the variational distribution (full derivation available in supplemental material):

$$\begin{aligned}\Psi^\star &= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathrm{KL}(q_\Psi(\theta|X)||p(\theta|X)] \\ &\approx \arg\min_\Psi \frac{1}{M} \times \sum_{\substack{X^m\sim p(X) \\ \theta^m\sim q_\Psi(\theta|X^m)}} \log q_\Psi(\theta^m|X^m) - \log p(X^m, \theta^m)\end{aligned} \tag{7}$$

Note that, as can be seen in supplemental material, we also studied the usage of the *forward* KL divergence, more common for SBI techniques that assume that the target density p is only *implicitly* defined by a simulator [7].

3 Experiments

All experiments were performed on a computational cluster with 40 Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz CPUs (32Gb RAM per CPU), and a single Tesla V100 (32Gb) GPU. We implemented our architecture on GPU using *Tensorflow Probability* [8], and SBI methods on CPU using the SBI Python library [33].

3.1 Gaussian random effects

As a first experiment, we want to assess the performance of our method on a simple experiment, for which theoretical results are available. We use the following model:

$$\begin{aligned} G,\ D,\ N = 3,\ 2,\ 50 \qquad & \sigma_\mu,\ \sigma_g,\ \sigma_x = 1.0,\ 0.2,\ 0.05 \\ \mu \sim N(\vec{0}_D, \sigma_\mu) \quad & \mu^1,\ \dots\ , \mu^G|\mu \sim N(\mu, \sigma_g) \\ x_n^g|\mu^g \sim N(\mu^g, \sigma_x) & \end{aligned} \tag{8}$$

where we try to recover the posterior distributions of the parameters $\mu, \mu^1, \dots, \mu^G$ given the observed data $X = [x_1^1, \dots x_N^G]$. Put more simply, given N points in each group g we wish to estimate the group mean μ^g, and given all group means we wish to estimate the population mean μ. For this problem, we note the presence of an analytical ground truth posterior derived in the supplemental material.

In this section we use a Hierarchical encoder featuring 2 set transformers (encoding size of 16; split in 4 attention heads of size 4; encoder with 2 SAB blocks; decoder with 1 seed vector). For the conditional density estimators, we use simple affine flow with triangular scale. Note that we therefore forgo the usage of a more expressive normalizing flow. This is to obtain Gaussian posteriors that can be analytically compared to the theoretical ones (see supplemental material). We use the Adam optimizer (10^{-3} learning rate), and minibatch gradient descent (batch size 32, 32 θ^m draws per X^m) on a dataset of $20,000$ points (θ^m, X^m). Convergence took 10 epochs (under a minute).

We compare our technique with a state-of-the-art amortized density estimation technique: NPE-C [14]. NPE-C was implemented using 3 Masked Autoregressive Flow (MAF) [25] blocks, each with 32 hidden units. To provide a fair point of comparison, we plugged as embedder to the density estimator a stacking of a Set transformer [24] contracting the N plate (all embedding size 8: 2 SABs with 4 heads, followed by a PMA with seed size 1 and a linear unit), and concatenated its output with the one of a similar Set Transformer contracting the G plate. We carefully designed the architecture of that embedder so as to provide a way for the density estimator to estimate per-group statistics, and across-group statistics. The amortized training took 207 epochs over the same dataset as the one used to train our methods, and more than 3 hours (2d of CPU time). In terms of parametrization, we also note that, as it flattens the parameter space, the NPE-C has $\mathcal{O}(G^2D^2)$ parameters whereas we have $\mathcal{O}(D^2)$ (see section 2.2).

Results can be observed in Figure 2. Both method fit well to the theoretical ground truth in this example.

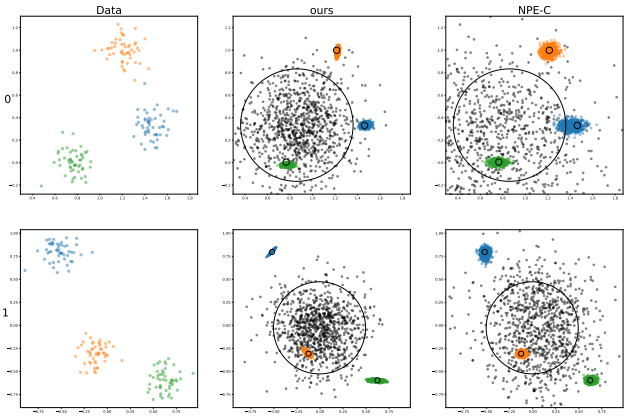

Figure 2: Graphical results on the Gaussian random effects. Rows represent 2 different data points. Left column represents data, with colors representing 3 different groups. Other columns represent posterior samples for μ (black) and μ^1, μ^2, μ^3. Second column represents our results, and third column represents NPE-C results. Associated with those posterior samples are analytical solutions (thin black circles), centered on the analytical MAP point, and whose radius correspond to 2 times the standard deviation of the analytical posterior: 95 % of the draws from a posterior should fall within the corresponding circle. Both our technique and NPE-C manage to recover the correct posteriors.

3.2 Gaussian mixture with random effects

As a second experiment we want to test our architecture on a more challenging, yet interpretable experiment. We consider the following model:

$$\begin{aligned} &G,\ L,\ D,\ N = 3,\ 3,\ 2,\ 50 \qquad \kappa,\ \sigma_\mu,\ \sigma_g,\ \sigma_x = 1,\ 1.0,\ 0.2,\ 0.05 \\ &\mu_1, \ \dots\ , \mu_L \sim N(\vec{0}_D, \sigma_\mu) \qquad \mu_l^1, \ \dots\ , \mu_l^G | \mu_l \sim N(\mu_l, \sigma_g) \end{aligned} \tag{9}$$

$$\begin{aligned} &\pi^1, \ \dots\ , \pi^G \in [0,1]^L \sim \text{Dirichlet}([\kappa] \times L) \\ &x_n^g | \pi^g, [\mu_1^g, \ \dots\ , \mu_L^g] \sim \text{Mixture}(\pi^g, [N(\mu_1^g, \sigma_x), \dots, N(\mu_L^g, \sigma_x)]) \end{aligned} \tag{10}$$

Though of low dimensionality, this model features a mixture of distributions, a well-known difficult setup for inference [18]. At the heart of the complexity of this example is the so-called *label switching* problem [18] (further analysed in supplemental material). The problem is even further complexified by the random effects introduced in the G groups. Yet this problem is not unnecessarily complex as it can be seen as a simplification of our next target experiment (see section 3.3).

We use a Hierarchical encoder featuring 2 set transformers (encoding size of 32; split in 4 attention heads of size 8; encoder with 2 SAB blocks; decoder with 1 seed vector). For the conditional density estimators, we use a flow comprised of an affine block with diagonal scale, followed by a MAF with hidden dimensions $[32, 32, 32]$. We use the Adam optimizer (10^{-3} learning rate), minibatch gradient descent (batch size 32, 32 θ^m draws per X^m) over a training dataset of size $20,000$. Training took 30 epochs, and under 3 minutes.

We compare our technique with NPE-C, with the same architecture as described in 3.1. Training took 296 epochs and more than 9h (6d 22h of CPU time). To provided a fair point of comparison -even if we position ourselves in the amortized setup- we also ran the NPE-C in its original, most efficient sequential setting (SNPE-C). In the SNPE-C setup, we specialize the posterior for the example at hand across several rounds of simulation. Training was run over 5 rounds and 4h (349, 50, 45, 31, 99 epochs respectively, total CPU time 2d 22h for a dataset of an effective size of 5000). We also acknowledge here the existence of different SBI techniques, namely SNRE-B [34] and SNLE-A [27], analysed in supplemental materials, that did not provide consistent results on this problem.

Results can be seen in Figure 3. We note that no technique recovers the complexity of the theoretical posterior. NPE-C and SNPE-C's performances are very degraded, and samples point in a broad

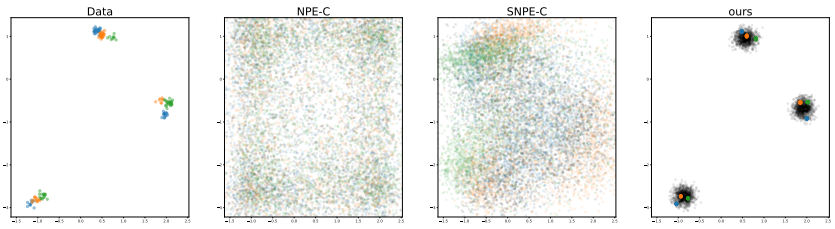

Figure 3: Graphical representation of the inference results of various methods on the Gaussian mixture with random effects example. First column represents a data point. All other columns represent the posterior samples for population mixture components $\mu_1, \ldots, \mu_3$ (black) and group mixture components $\mu_1^1, \ldots, \mu_3^1, \mu_1^2, \ldots, \mu_3^2, \mu_1^3, \ldots, \mu_3^3$. Second column represents the results of an amortized NPE-C. Third column represents the result of a non-amortized SNPE-C, specialized for the given data point. Last column represent our amortized technique. We obtain results of greater experimental value.

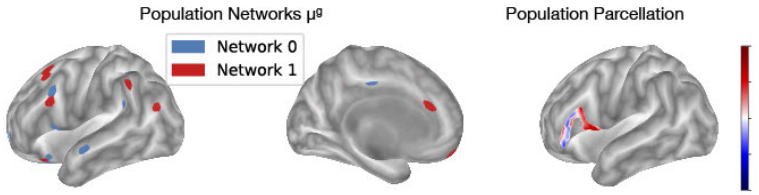

Figure 4: Results for our neuroimaging experiment in section 3.3. Networks show the top 1% connected components. Network 0 (in blue) agrees with current knowledge in semantic/phonologic processing while network 1 (in red) agrees with current networks known in language production [15; 42]. Our soft parcellation, where coloring lightens as the cortical point is less probably associated with one of the networks, also agrees with current knowledge where more posterior parts are involved in language production while more anterior ones in semantic/phonological processing [15; 42].

domain centered in between the 3 blobs of points. Our method doesn't truly recover the theoretical posterior (see supplemental material for details) but gives results close to the intuition one forms of the problem. By exploiting the structure of the HBM, we therefore obtain results of greater experimental value: contrary to NE-C, our results can be readily used to estimate the components of the Gaussian mixture.

3.3 Neuroimaging: modelling multi-scale variability in Broca's area functional parcellation

To show the efficiency of our method in a high-dimensional neuroimaging context, we consider the model proposed by Kong et al. [21]. This features a HBM to parcel the human brain cortex in terms of its functional MRI (fMRI)-based connectivity simultaneously at the population and individual level. This approach's main goal, achieved through a complex model-specific EM algorithm, is to address the classical problem in neuroscience of estimating population commonalities along with individual characteristics. Here, we study this model, for which ADAVI provides automatically an efficient fitting strategy, in the context of parcelling the region of left inferior frontal gyrus (IFG) composed by pars orbitalis and triangularis into $L = 2$ functionally-distinct regions, an open problem in neuroscience [see e.g. 15].

As Kong et al. [21], we consider a population of S=30 subjects, each with $T = 2$ acquisition sessions, from the Human Connectome Project dataset [36]. The fMRI connectivity between a cortical point and the rest of the brain, split in $D = 1,483$ regions, is represented as a vector of length D with each component quantifying the temporal correlation of blood-oxygenation between the point and a region. A main hypothesis of Kong et al. [21], and the fMRI field, is that the fMRI connectivity of points belonging to the same parcel share a similar connectivity pattern or correlation vector. Following Kong et al. [21], we represent D-dimensional correlation vectors as RVs on the positive quadrant of the D-dimensional unit-sphere. We do this efficiently assuming they have a $\mathcal{L}$-normal distribution,

or Gaussian under the transformation of the link function $\mathcal{L}(x) = \sqrt{\text{SoftmaxCentered}(x)}$ [8]. The resulting model, hierarchical with three levels: session, subject, and population, is expressed as the HBM:

$$\begin{aligned}
N, T, S, D, L = 314, 4, 30, 1483, 2 \qquad & \epsilon^- = \sigma^- = \kappa^- = -10 \\
\pi = 2 \qquad & \epsilon^+ = \sigma^+ = \kappa^+ = -8 \\
\mathcal{L}^{-1}(\mu_l^g) \sim N(\vec{0}_{D-1}, \Sigma_g) \qquad & \text{Log}(\epsilon_l) \sim \text{Uniform}_L(\epsilon^-, \epsilon^+) \\
\mathcal{L}^{-1}(\mu_l^s)|\mu_l^g,\ \epsilon_l \sim N(\mathcal{L}^{-1}(\mu_l^g), \epsilon_l) \qquad & \text{Log}(\sigma_l) \sim \text{Uniform}_L(\sigma^-, \sigma^+) \\
\mathcal{L}^{-1}(\mu_l^{s,t})|\mu_l^s,\ \sigma_l \sim N(\mathcal{L}^{-1}(\mu_l^s), \sigma_l) \qquad & \text{Log}(\kappa) \sim \text{Uniform}_1(\kappa^-, \kappa^+) \\
\Pi \sim \text{Dirichlet}([\pi] \times L) &
\end{aligned} \tag{11}$$

Where parameters μ represent the nested connectivity vectors at the population (g), subject (s), and session (t) level. Thus, the connectivity signal at a given vertex n is given by the mixture:

$$\mathcal{L}^{-1}(X_n^{s,t})|[\mu_1^{s,t},\ \ldots\ , \mu_L^{s,t}], \kappa, \Pi \sim \text{Mixture}(\Pi, [N(\mathcal{L}^{-1}(\mu_1^{s,t}), \kappa),\ \ldots\ , N(\mathcal{L}^{-1}(\mu_L^{s,t}), \kappa)]), \tag{12}$$

where the prior $N_{D-1}(\vec{0}_{D-1}, \Sigma_g)$ is the result of a data pre-processing step described in supplemental material.

Our aim is therefore to identify $L = 2$ functional networks that would produce a functional parcellation of the studied IFG section. In this setting, the parameters θ are the networks μ. For these, we will perform full-posterior inference using our automatically derived architecture instead of the complex EM computation, derived in Kong et al. [21].

Our method, ADAVI, derives from eqs. (11) and (12) a Hierarchical encoder featuring 3 set transformers (encoding size of 1024; split in 4 attention heads of size 128; encoder with 1 SAB blocks; decoder with 1 seed vector). For the conditional density estimators, we use a flow comprised of an affine block with diagonal scale, followed by a MAF with hidden dimensions $[1024]$. We use the Adam optimizer (10^{-3} learning rate), stochastic gradient descent (batch size 1, 4 θ^m draws per X^m). We further used a warm-up strategy for our training, for a total training time of 2h, as further described in supplemental material.

The results of our experiment are shown in fig. 4. Where, in agreement with the literature, the two predominant networks splitting the region composed by pars triangularis and opercularis are related to semantic/phonologic processing (0, in blue) and language production (1, in red) [15; 42]. Our inference of the full posterior enabled us to compute the confidence interval for all μ parameters, which was at least one order of magnitude smaller than the values of μ, pointing to a high confidence level in the estimation of our model. Furthermore, the full posterior also provided us with the means to produce a soft parcellation, accounting for data uncertainty more accurately than with hard clustering.

4 Conclusion and Discussion

4.1 Benefits of amortization and extension to a sequential version

Our main comparison point in this work has been amortized, normalizing-flows based inference techniques [14; 26; 34]. But our contribution is actually rather orthogonal to those: we propose a principled and automated way to combine several density estimators in a hierarchical structure. As such, our methods could be applied to a different class of estimators such as VAEs [19]. SBI techniques are meta-algorithms, in the sense that they leverage an arbitrary normalizing flow technology for inference. Similarly, we could leverage the SBI techniques and extend our work into a sequential version through the reparameterization of our conditional estimators q_i (see section 2.3). Ultimately, our method is not meant as an alternative to SBI, but a complement to it for the pyramidal class of problems described in section 2.1.

We choose to posit ourselves as an amortized technique. Yet, in our target experiment [21] (see section 3.3), the inference is performed on a specific data point. Since we also have access to a differentiable density function, techniques in the line of Automatic Differentiation VI [22] could appear as a more natural option. Our experimental experience on that example however makes us put

forth the value that can be obtained from sharing learning across multiple examples, as amortization entitles [7]. Specifically, we encountered less issues related to local minima of the loss, a canonical issue for VI [3]. We would therefore argue against the intuition that a (locally) amortized technique is necessarily wasteful in the context of a single data point.

What's more, in both our Gaussian RE (section 3.1) and Gaussian mixture (section 3.2), given our very fast convergence, we argue that we benefit from amortization without too much of a computational overhead. Yet, the cost to pay for amortization can become prohibitive in high dimensions, as exponentially more training examples can be necessary to estimate densities properly [10]. Specializing for a local parameter regime could therefore make us benefit from amortization without too steep an upfront training cost.

4.2 Leveraging structure in density estimation

SBI techniques have also been presented as *likelihood-free methods* [see naming in 14; 34], buidling upon the conception of a simulator as a black box [a concept explicited in 7]. Following the general line of thinking of Bronstein et al. [5], we argue that there is much value to gain from analysing the structure and geometry of a problem. In the SBI setup, this structure can be exploited through learnable data embedders [28]. We go one step beyond and also use the problem structure to shape our density estimator: we factorize the parameter space of a problem into smaller components, and share network parametrization across tasks we know to be equivalent (see section 2.3).

Contrary to the notion of black box, we argue that experimenters oftentimes can identify properties such as exchangeability in their experiments [11]. The presence of such properties is also not tied to the explicit modelling of a problem as a HBM: Zaheer et al. [40] rather describe this property as an permutation invariance present in the studied data. As a consequence, though our derivation is based on HBMs, we believe that the working principle of our method could be applied to a broader class of simulators featuring exchangeability. Our reliance on HBMs is in fact only tied to our usage of the revers KL loss (see section 2.4), a readily modifiable implementation detail. Beyond the notion of plates, some VI literature is based on the complex and automated analysis of the data flow inside a generative model [16; 35]. This kind of analysis could potentially automatically identify, beyond plates, desirable properties that could be then leveraged for efficient inference.

A main limitation in our work is that our amortized posterior distribution is ultimately akin to a mean field approximation of the problem [3]. Though computationally attractive, this limits the expressivity of our variational family [16; 30]. We ponder the possibility to leverage VI techniques such as the one derived by Ranganath et al. [30] and their variational objectives for structured populations of normalizing flows such as ours.

In this work, we restrict o urselves to the pyramidal class of Bayesian networks (see section 2.1. Going further, this class of models could be extended to cover more and more use-cases. This bottom-up approach stands at opposite ends from the generic approach of SBI techniques [7]. But, as our target experiment in 3.3 demonstrates, we argue that in the long run this bottom-up approach could result in more scalable and efficient architectures, applicable to challenging setups such as neuroimaging.

4.3 Conclusion

For the delineated yet expressive class of pyramidal Bayesian models, we have introduced a potent, automatically derived architecture able to perform amortized parameter inference. This architecture leverages the mapping of functions across plates: through a Hierarchical Encoder it conditions a network of normalizing flows that stand as a variational family *dual* to the forward HBM. To demonstrate the expressivity and scalability of our method, we successfully applied it to a challenging neuroimaging setup. Our work stands as an original attempt to leverage exchangeability in a generative model, and presents a general framework that could be extended to different forms of inference (VAEs, Sequential inference) and structures (beyond the sole notion of plates).

Acknowledgments and Disclosure of Funding

This work was partially supported by the ERC-StG NeuroLang ID:757672.

We would like to warmly thank Dr. Thomas Yeo and Dr. Ru Kong (CBIG) who made pre-processed HCP functional connectivity data available to us.

We also would like to thank Dr. Majd Abdallah (Inria) for his insights and perspectives regarding our functional connectivity results.

References

[1] Luca Ambrogioni, Kate Lin, Emily Fertig, Sharad Vikram, Max Hinne, Dave Moore, and Marcel van Gerven. Automatic structured variational inference. page 10.

[2] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019. URL `http://jmlr.org/papers/v20/18-403.html`.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.2017.1285773. URL `http://arxiv.org/abs/1601.00670`. arXiv: 1601.00670.

[4] Anna K Bonkhoff, Jae-Sung Lim, Hee-Joon Bae, Nick A Weaver, Hugo J Kuijf, J Matthijs Biesbroek, Natalia S Rost, and Danilo Bzdok. Generative lesion pattern decomposition of cognitive impairment after stroke. *Brain Communications*, 05 2021. ISSN 2632-1297. doi:10.1093/braincomms/fcab110. URL `https://doi.org/10.1093/braincomms/fcab110`. fcab110.

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, July 2017. ISSN 1053-5888, 1558-0792. doi:10.1109/MSP.2017.2693418. URL `http://arxiv.org/abs/1611.08097`. arXiv: 1611.08097.

[6] Shayle R Searle Charles E McCulloch. *Generalized, Linear, and Mixed Models*. Wiley-Interscience, January 2001. ISBN 978-0-471-19364-7 0-471-19364-X.

[7] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, page 201912789, May 2020. ISSN 0027-8424, 1091-6490. doi:10.1073/pnas.1912789117. URL `http://www.pnas.org/lookup/doi/10.1073/pnas.1912789117`.

[8] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow Distributions. *arXiv:1711.10604 [cs, stat]*, November 2017. URL `http://arxiv.org/abs/1711.10604`. arXiv: 1711.10604.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv:1605.08803 [cs, stat]*, February 2017. URL `http://arxiv.org/abs/1605.08803`. arXiv: 1605.08803.

[10] David Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. pages 1–32. American Mathematical Society.

[11] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. edition, 2004.

[12] W. R. Gilks, A. Thomas, and D. J. Spiegelhalter. A Language and Program for Complex Bayesian Modelling. *The Statistician*, 43(1):169, 1994. ISSN 00390526. doi:10.2307/2348941. URL `https://www.jstor.org/stable/10.2307/2348941?origin=crossref`.

[13] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *arXiv:1810.01367 [cs, stat]*, October 2018. URL `http://arxiv.org/abs/1810.01367`. arXiv: 1810.01367.

[14] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic Posterior Transformation for Likelihood-Free Inference. *arXiv:1905.07488 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1905.07488. arXiv: 1905.07488.

[15] Stefan Heim, Simon B. Eickhoff, Anja K. Ischebeck, Angela D. Friederici, Klaas E. Stephan, and Katrin Amunts. Effective connectivity of the left BA 44, BA 45, and inferior temporal gyrus during lexical and phonological decisions identified with DCM. *Human Brain Mapping*, 30(2):392–402, February 2009. ISSN 10659471. doi:10.1002/hbm.20512.

[16] Matthew D. Hoffman and David M. Blei. Structured Stochastic Variational Inference. *arXiv:1404.4114 [cs]*, November 2014. URL http://arxiv.org/abs/1404.4114. arXiv: 1404.4114.

[17] Ekaterina Iakovleva, Jakob Verbeek, and Karteek Alahari. Meta-Learning with Shared Amortized Variational Inference. page 13.

[18] A. Jasra, C. C. Holmes, and D. A. Stephens. Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. *Statistical Science*, 20 (1), February 2005. ISSN 0883-4237. doi:10.1214/088342305000000016. URL https://projecteuclid.org/journals/statistical-science/volume-20/issue-1/Markov-Chain-Monte-Carlo-Methods-and-the-Label-Switching-Problem/10.1214/088342305000000016.full.

[19] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv: 1312.6114.

[20] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *arXiv:1908.09257 [cs, stat]*, June 2020. doi:10.1109/TPAMI.2020.2992934. URL http://arxiv.org/abs/1908.09257. arXiv: 1908.09257.

[21] Ru Kong, Jingwei Li, Csaba Orban, Mert R Sabuncu, Hesheng Liu, Alexander Schaefer, Nanbo Sun, Xi-Nian Zuo, Avram J Holmes, Simon B Eickhoff, and B T Thomas Yeo. Spatial Topography of Individual-Specific Cortical Networks Predicts Human Cognition, Personality, and Emotion. page 19, 2018.

[22] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic Differentiation Variational Inference. *arXiv:1603.00788 [cs, stat]*, March 2016. URL http://arxiv.org/abs/1603.00788. arXiv: 1603.00788.

[23] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. ISSN 0047-259X. doi:https://doi.org/10.1016/S0047-259X(03)00096-4. URL https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

[24] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/lee19d.html.

[25] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv:1705.07057 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1705.07057. arXiv: 1705.07057.

[26] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv:1912.02762 [cs, stat]*, December 2019. URL http://arxiv.org/abs/1912.02762. arXiv: 1912.02762.

[27] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows. *arXiv:1805.07226 [cs, stat]*, January 2019. URL http://arxiv.org/abs/1805.07226. arXiv: 1805.07226.

[28] Stefan T. Radev, Ulf K. Mertens, Andreass Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *arXiv:2003.06281 [cs, stat]*, April 2020. URL http://arxiv.org/abs/2003.06281. arXiv: 2003.06281.

[29] Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black Box Variational Inference. *arXiv:1401.0118 [cs, stat]*, December 2013. URL http://arxiv.org/abs/1401.0118. arXiv: 1401.0118.

[30] Rajesh Ranganath, Dustin Tran, and David M. Blei. Hierarchical Variational Models. *arXiv:1511.02386 [cs, stat]*, May 2016. URL http://arxiv.org/abs/1511.02386. arXiv: 1511.02386.

[31] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1505.05770. arXiv: 1505.05770.

[32] Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized Inference Regularization. page 10.

[33] 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. *arXiv:2007.09114 [cs, q-bio, stat]*, July 2020. URL http://arxiv.org/abs/2007.09114. arXiv: 2007.09114.

[34] Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. Likelihood-free inference by ratio estimation. *arXiv:1611.10242 [stat]*, September 2020. URL http://arxiv.org/abs/1611.10242. arXiv: 1611.10242.

[35] Dustin Tran, Rajesh Ranganath, and David M. Blei. Hierarchical Implicit Models and Likelihood-Free Variational Inference. *arXiv:1702.08896 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1702.08896. arXiv: 1702.08896.

[36] D. C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T. E. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. W. Curtiss, S. Della Penna, D. Feinberg, M. F. Glasser, N. Harel, A. C. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. E. Petersen, F. Prior, B. L. Schlaggar, S. M. Smith, A. Z. Snyder, J. Xu, and E. Yacoub. The Human Connectome Project: a data acquisition perspective. *Neuroimage*, 62(4):2222–2231, Oct 2012.

[37] Antoine Wehenkel and Gilles Louppe. Graphical Normalizing Flows. *arXiv:2006.02548 [cs, stat]*, October 2020. URL http://arxiv.org/abs/2006.02548. arXiv: 2006.02548.

[38] Christian Weilbach, Boyan Beronov, William Harvey, and Frank Wood. Structured Conditional Continuous Normalizing Flows for Efficient Amortized Inference in Graphical Models. page 10.

[39] Mike Wu, Kristy Choi, Noah Goodman, and Stefano Ermon. Meta-Amortized Variational Inference and Learning. *arXiv:1902.01950 [cs, stat]*, September 2019. URL http://arxiv.org/abs/1902.01950. arXiv: 1902.01950.

[40] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. *arXiv:1703.06114 [cs, stat]*, April 2018. URL http://arxiv.org/abs/1703.06114. arXiv: 1703.06114.

[41] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, August 2019. ISSN 0162-8828, 2160-9292, 1939-3539. doi:10.1109/TPAMI.2018.2889774. URL https://ieeexplore.ieee.org/document/8588399/.

[42] Yizhen Zhang, Kuan Han, Robert Worth, and Zhongming Liu. Connecting concepts in the brain by mapping cortical representations of semantic relations. *Nature Communications*, 11(1):1877, April 2020. ISSN 2041-1723. doi:10.1038/s41467-020-15804-w.

Supplemental material

This supplemental material complements our main work both with theoretical points and experiments:

- A complements to the Gaussian random effects experiment (section 3.1). We present results mostly related to hyperparameter analysis.
- B complements to the Gaussian mixture with random effects experiment (section 3.2). We explore the complexity of the example at hand and present aditional SBI results [27; 34].
- C complements to the MS-HBM experiments (section 3.3). We present a toy dimensions experiment and implementation details.

A Complements to the Gaussian Random Effects experiment: hyperparameter analysis

This section is a complement to the experiment described in section 3.1, we thus consider the model described in eq. (8). We present results of practical value, mostly related to hyperparameters.

A.1 Descriptors, inputs to ADAVI

We can analyse the model decribed in eq. (8) using the descriptors defined in eq. (2). Those descriptors constitute the inputs our methodology needs to automatically derive the *dual* architecture from the generative HBM:

$$\begin{aligned}
\mathcal{V} &= \{\mu, M_G, X\} \\
\mathcal{P} &= \{\mathbf{P}_0, \mathbf{P}_1\} \\
\text{Card} &= \{\mathbf{P}_0 \mapsto N, \mathbf{P}_1 \mapsto G\} \\
\text{Hier} &= \{\mu \mapsto 2, M_G \mapsto 1, X \mapsto 0\} \\
\text{Shape} &= \{M \mapsto (D,), M_G \mapsto (D,), X \mapsto (D,)\} \\
\text{Link} &= \{M \mapsto \text{Identity}, M_G \mapsto \text{Identity}, X \mapsto \text{Identity}\}
\end{aligned} \tag{A.1}$$

A.2 Derivation of an analytic posterior

To have a ground truth to which we can compare our methods results (see fig. 2 notably), we derive the following analytic posterior distributions. Assuming we know $\sigma_\mu,\ \sigma_g,\ \sigma_x$:

$$\hat{\mu}^g = \frac{1}{N}\sum_{n=1}^{N} x_n^g \tag{A.2a}$$

$$\tilde{\mu}^g | \hat{\mu}^g \sim \mathcal{N}\left(\hat{\mu}^g, \frac{\sigma_x}{\sqrt{N}}\,\text{Id}_D\right) \tag{A.2b}$$

$$\hat{\mu} = \frac{1}{G}\sum_{g=1}^{G} \hat{\mu}^g \tag{A.2c}$$

$$\tilde{\mu} | \hat{\mu} \sim \mathcal{N}\left(\frac{\frac{G}{\sigma_g^2}\hat{\mu}}{\frac{1}{\sigma_\mu^2} + \frac{G}{\sigma_g^2}}, \sqrt{\frac{1}{\frac{1}{\sigma_\mu^2} + \frac{G}{\sigma_g^2}}}\,\text{Id}_D\right) \tag{A.2d}$$

Where in equation A.2b we neglect the influence of the prior (against the evidence) on the posterior in light of the large number of points drawn from the distribution.

A.3 Training losses full derivation and comparison

Full formal derivation Following the nomenclature introduced in Papamakarios et al. [26], there are 2 different ways in which we could train our variational distribution:

- using a *forward* KL divergence, benefiting from the fact that we can sample from our generative model to produce a dataset $\{(\theta^m, X^m)\}_{m=1\dots M}$. This is the loss used in most of the SBI literature [7], as those are based around the possibility to be *likelihood-free*, and have a target density p only implicitly defined by a simulator:

$$\begin{aligned}
\Psi^\star &= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathrm{KL}(p(\theta|X)||q_\Psi(\theta|X)] \\
&= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathbb{E}_{\theta\sim p(\theta|X)}[\log p(\theta|X) - \log q_\Psi(\theta|X)]] \\
&= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathbb{E}_{\theta\sim p(\theta|X)}[-\log q_\Psi(\theta|X)]] \\
&= \arg\min_\Psi \int p(X)\big[\int -p(\theta|X)\log q_\Psi(\theta|X)d\theta\big]dX \\
&= \arg\min_\Psi \int\int -p(X,\theta)\log q_\Psi(\theta|X)d\theta dX \\
&\approx \arg\min_\Psi \frac{1}{M} \times \sum_{(X^m,\theta^m)\sim p(X,\theta)} -\log q_\Psi(\theta^m|X^m)
\end{aligned} \tag{A.3}$$

- using a *reverse* KL divergence, already introduced in our paper (section 2.4):

$$\begin{aligned}
\Psi^\star &= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathrm{KL}(q_\Psi(\theta|X)||p(\theta|X)] \\
&= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathbb{E}_{\theta\sim q_\Psi(\theta|X)}[\log q_\Psi(\theta|X) - \log p(\theta|X)]] \\
&= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathbb{E}_{\theta\sim q_\Psi(\theta|X)}[\log q_\Psi(\theta|X) - \log p(X,\theta) + \log p(X)]] \\
&= \arg\min_\Psi \mathbb{E}_{X\sim p(X)}[\mathbb{E}_{\theta\sim q_\Psi(\theta|X)}[\log q_\Psi(\theta|X) - \log p(X,\theta)]] \\
&= \arg\min_\Psi \int p(X)\big[\int q_\Psi(\theta|X)[\log q_\Psi(\theta|X) - \log p(X,\theta)]d\theta\big]dX \\
&= \arg\min_\Psi \int\int p(X)q_\Psi(\theta|X)[\log q_\Psi(\theta|X) - \log p(X,\theta)]d\theta dX \\
&\approx \arg\min_\Psi \frac{1}{M} \times \sum_{\substack{X^m\sim p(X) \\ \theta^m\sim q_\Psi(\theta|X^m)}} \log q_\Psi(\theta^m|X^m) - \log p(X^m,\theta^m)
\end{aligned} \tag{A.4}$$

As it more uniquely fits our setup and provided better results experimentally, we chose to focus on the usage of the *reverse* KL divergence. During our experiments, we also tested the usage of the *unregularized ELBO* loss:

$$\Psi^\star = \arg\min_\Psi \frac{1}{M} \times \sum_{\substack{X^m\sim p(X) \\ \theta^m\sim q_\Psi(\theta|X^m)}} -\log p(X^m,\theta^m) \tag{A.5}$$

This formula differs from the one of the reverse KL loss by the absence of the term $q_\Psi(\theta^m|X^m)$, and is a converse formula to the one of the forward KL (in the sense that it permutes the roles of q and p).

Intuitively, it posits our architecture as a pure sampling distribution that aims at producing points θ^m in regions of high joint density p. In that sense, it acts as a first moment approximation for the target posterior distribution (akin to MAP parameter regression). Experimentally, the usage of the *unregularized ELBO* loss provided fast convergence to a mode of the posterior distribution, with very low variance for the variational approximation.

We argue the possibility to use the *unregularized ELBO* loss as a *warm-up* before switching to the reverse KL loss, with the latter considered here as a regularization of the former. We introduce this training strategy as an example of the modularity of our approach, where one could transfer the rapid learning from one task (amortized mode finding) to another task (amortized posterior estimation).

Graphical comparison In Figure A.1 we analyse the influence of these 3 different losses on the training of our posterior distribution, compared to the analytical ground truth. This example is typical of the relative behaviors induced on the variational distributions by each loss:

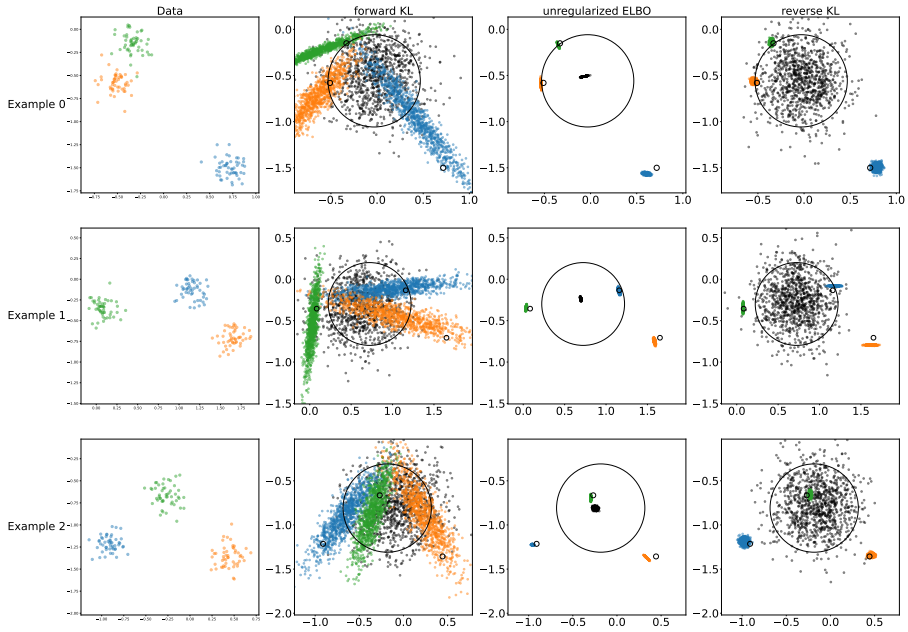

Figure A.1: Graphical results on the Gaussian random effects example for our architecture trained using 3 different losses. Rows represent 3 different data points. Left column represents data, with colors representing 3 different groups. Other columns represent posterior samples for μ (black) and μ^1, μ^2, μ^3. Visually, posterior samples μ^1, μ^2, μ^3 should be concentrated around the mean of the data points with the same color, and the black points μ should be repartitioned around the mean of the 3 group means (with a shift towards 0 due to the prior). Associated with the posterior samples are analytical solutions (thin black circles), centered on the analytical MAP point, and whose radius correspond to 2 times the standard deviation of the analytical posterior: 95 % of the draws from a posterior should fall within the corresponding circle.

- The forward KL provides very erratic training, and results after several dozen epochs (several minutes) with a careful early stopping in posteriors with too large variance.
- The unregularized ELBO loss converges in less then 3 epochs (a couple dozen seconds), and provides posteriors with very low variance, concentrated on the MAP estimates of their respective parameters.
- The reverse KL converges in less 10 epochs (less than 3 minutes) and provides relevant variance.

Losses convergence speed We analyse the relative convergence speed of our variational posterior to the analytical one when using the 3 aforementioned losses for training. To measure the convergence, we compute analytically the KL divergence between the variational posterior and the analytical one (every distribution being a Gaussian), summed for every distribution, and averaged over a validation dataset of size 2000.

We use a training dataset of size 2000, and for each loss repeated the training 20 times (batch size 10, 10 θ^m samples per X^m) for 10 epochs, resulting in 200 optimizer calls. This voluntary low number allows us to asses how close is the variational posterior to the analytical posterior after only a brief training. Results are visible in Table 1, showing a faster convergence for the *unregularized ELBO*. After 800 more optimizer calls, the tendency gets inverted and the *reverse KL* loss appears as the superior loss (though we still notice a larger variance).

Table 1: Convergence of the variational posterior to the analytical posterior over an early stopped training (200 batches) and after convergence (1000 batches) for the Gaussian random effects example

		Mean of analytical KL divergences from the theoretical posterior (low is good)			
		Early stopping		After convergence	
Loss	NaN runs	Mean	Std	Mean	Std
forward KL	0	3847.7	5210.4	2855.3	4248.1
unregularized ELBO	0	6.6	0.7	6.2	0.9
reverse KL	2	12.3	19.8	3.0	4.1

The large variance in the results may point towards the need for adapted training strategies involving Learning rate decay and/or scheduling [22], an extension that we leave for future work.

A.4 Monte Carlo approximation for the gradients and computational budget comparison

In section 2.4, for the reverse KL loss, we approximate expectations using Monte Carlo integration. We further train our architecture using minibatch gradient descent, as opposed to stochastic gradient descent as proposed by Kucukelbir et al. [22]. An interesting hyper-parametrization of our system resides in the effective batch size of our training, that depends upon:

- the size of the mini batches, determining the number of X^m points considered in parallel
- the number of θ^m draws per X^m point, that we use to approximate the gradient in the ELBO

More formally, we define a computational budget as the relative allocation of a constant effective batch size $\text{batch size} \times \theta \text{ draws per X}$ between batch size and $\theta \text{ draws per X}$.

To analyse the effect of the computational budget on training, we use a dataset of size 1000, and run experiment 20 times over the same number of optimizer calls with the same effective batch size per call. Results can be seen in fig. A.2. From this experiment we can draw the following conclusions:

- we didn't witness massive difference in the global convergence speed across computational budgets
- the bigger the budget we allocate to the sampling of multiple θ^m per point X^m (effectively going towards a stochastic training in terms of the points X^m), the more erratic is the loss evolution
- the bigger the budget we allocate to the X^m batch size, the more stable is the loss evolution, but our interpretation is that the resulting reduced number of θ^m draws per X^m augments the risk of an instability resulting in a NaN run

Experimentally, we obtained the best results by evenly allocating our budget to the X^m batch size and the number of θ^m draws per X^m point (typically, 32 and 32 respectively for an effective batch size of 1024). Overall, in the amortized setup, our experiment stand as a counterpoint to those of Kucukelbir et al. [22] who pointed towards the case of a single θ^m draw per point X^m as their preferred hyper-parametrization.

B Complements to the Gaussian mixture with random effects experiment: further posterior analysis

This section is a complement to the experiment described in section 3.2, we thus consider the model described in eq. (9). We explore the complexity of the theoretical posterior for this experiment. We also present additional SBI results [27; 34].

B.1 Descriptors, inputs to ADAVI

We can analyse the model decribed in eq. (9) using the descriptors defined in eq. (2). Those descriptors constitute the inputs our methodology needs to automatically derive the *dual* architecture from the

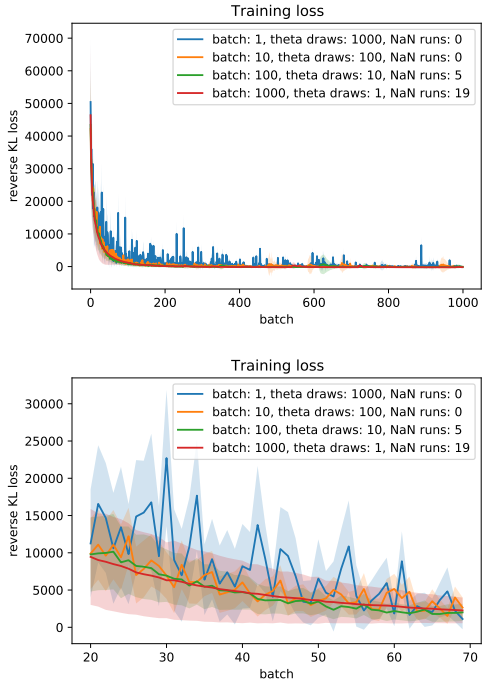

Figure A.2: Loss evolution across batches for different computational budgets. All experiments are designed so that to have the same number of optimizer calls (meaning that $\text{batch size} \times \text{epochs} = 1000$) and the same effective batch size (meaning that $\text{batch size} \times \theta$ draws per $\text{X} = 1000$). Every experiment is run 20 times, error bands showing the standard deviation of the loss at the given time point. Note that the blue line (batch size 1, 1000 θ draws per X) is more erratic than the other ones (even after a large number of batches). On the other hand, the red line (batch size 1000, 1 θ draws per X) is more stable, but 19 out of 20 runs ultimately resulted in an instability

generative HBM:

$$
\begin{aligned}
\mathcal{V} &= \{M, M_G, \Pi_G, X\} \\
\mathcal{P} &= \{\mathbf{P}_0, \mathbf{P}_1\} \\
\text{Card} &= \{\mathbf{P}_0 \mapsto N, \mathbf{P}_1 \mapsto G\} \\
\text{Hier} &= \{M \mapsto 2, M_G \mapsto 1, \Pi_G \mapsto 1, X \mapsto 0\} \\
\text{Shape} &= \{M \mapsto (L, D), M_G \mapsto (L, D), \Pi_G \mapsto (L,), X \mapsto (D,)\} \\
\text{Link} &= \{ \\
&\quad M \mapsto \text{Reshape}((LD,) \to (L, D)), \\
&\quad M_G \mapsto \text{Reshape}((LD,) \to (L, D)), \\
&\quad \Pi_G \mapsto \text{SoftmaxCentered}((L-1,) \to (L,)), \\
&\quad X \mapsto \text{Identity} \\
&\}
\end{aligned}
\tag{B.6}
$$

For the definition of the SoftmaxCentered link function, see Dillon et al. [8].

B.2 Amortized SBI techniques on the Gaussian mixture with random effects

We present here the results of 2 other amortized SBI techniques: NRE-B, based on likelihood ratio estimation [34] and NLE-A, estimating the density of the data X instead of the parameters θ [27]:

- for NRE-B, we used the embedder described in section 3.1 (contracting the G an N plates separately) and plugged it to a ResNet architecture (2 blocks of 50 units each). Training

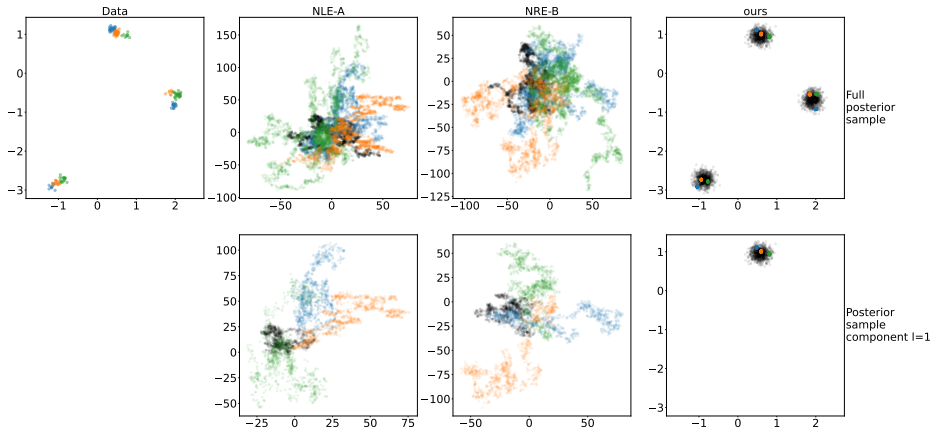

Figure B.3: Graphical representation of the inference results of various methods on the Gaussian mixture with random effects example. First column represents a data point. All other columns represent the posterior samples for $\mu_1, \ldots, \mu_3$ (black) and $\mu_1^1, \ldots, \mu_3^1, \mu_1^2, \ldots, \mu_3^2, \mu_1^3, \ldots, \mu_3^3$. Second column represents the results of an amortized NLE-A. Third column represents the results of an amortized NRE-B. Last column represent our amortized technique. For inference techniques only, the first row represent the superposition of the posteriors for all the mixture components, while the second row only represents the posterior samples for the first mixture component. Note that for NLE-A and NRE-B, the axis of the graph are not aligned to the ones of the data, and that the points are distributed on a very wide support.

took approximately 50h of CPU time. Sampling required a lengthy MCMC procedure (more that 80 hours of CPU time for a thousand samples).

- for NLE-A, we used 5 MAF blocks, each with 50 hidden units. To our knowledge NLE-A can only be applied to a flattened data vector, which implies a large parametrization $\mathcal{O}(G^2N^2D^2)$. Training took approximately 50h of CPU time (similar to NRE-B). In addition, sampling also requires a lengthy MCMC sampling (more that 80 hours of CPU time for a thousand samples, similar to NRE-B).

Results can be seen in Figure B.3. Both NLE-A and NRE-B sample posterior points in a very large support, underlining their difficulty to tackle the Gaussian mixture with random effects example.

B.3 Theoretical posterior recovery in the Gaussian mixture random effects model

We further analyse the complexity of model described in section 3.2.

Due to the label switching problem [18], the relative position of the L mixture components in the D space is arbitrary. Consider a non-degenerate example like the one in fig. B.4, where the data points are well separated in 3 blobs (likely corresponding to the $L = 3$ mixture components). Since there is no deterministic way to assign component $l = 1$ unequivocally to a blob of points, the posterior distribution for the position of the component $l = 1$ should be multi-modal, with -roughly- a mode placed at each one of the 3 blobs of points. This posterior would be the same for the components $l = 2$ and $l = 3$. In truth, the posterior is even more complex than this 3-mode simplification, especially when the mixture components are closer to each other in 2D (and the grouping of points into draws from a common component is less evident).

In fig. B.4, we note that our technique doesn't recover this multi-modality in its posterior, and instead assigns different posterior components to different blobs of points. Indeed, when plotting only the posterior samples for the first recovered component $l = 1$, all points are concentrated around the top-most blob, and not around each blob like the theoretical posterior would entail (see fig. B.4 second row).

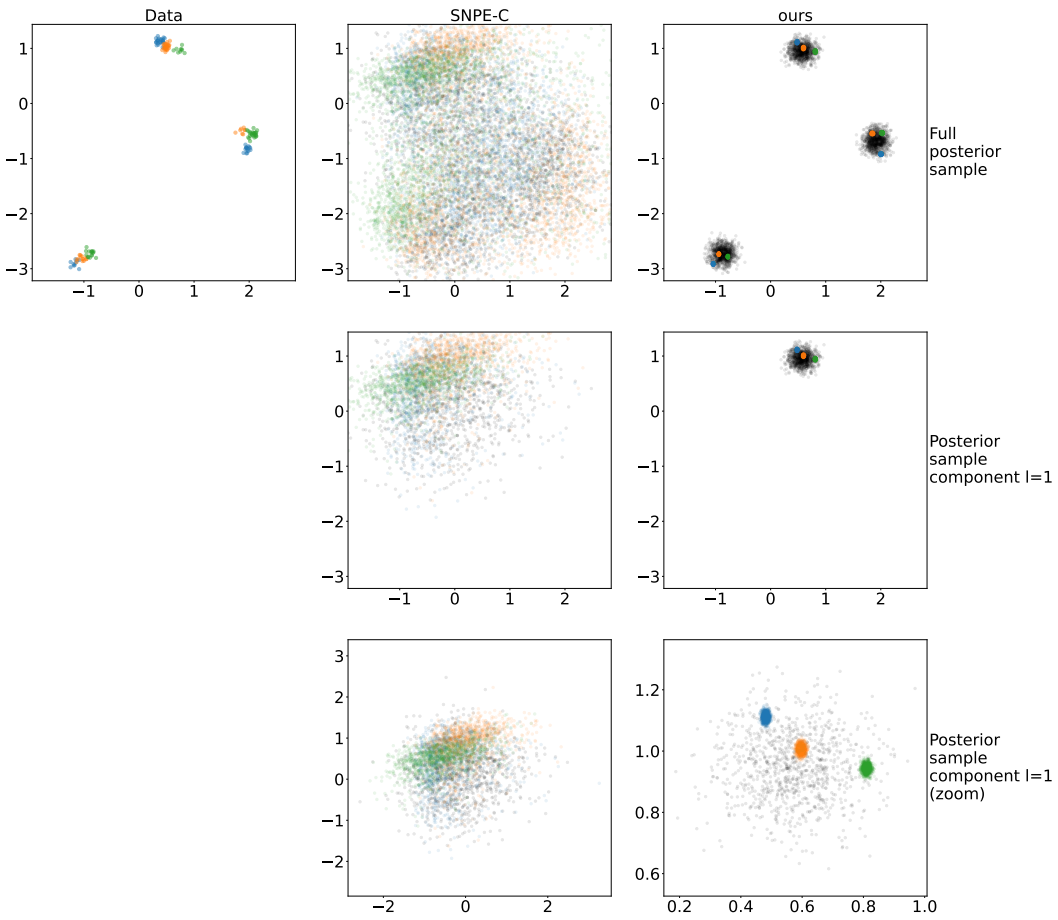

Figure B.4: Theoretical posterior recovery for the Gaussian mixture with random effects example. We represent the inference results of 2 methods: SNPE-C (second column) and ours (third column). First row represents the full posterior samples. Second and third row only represents the first mixture component samples (third row zooms in on the data). Notice how neither technique recovers the actual multi-modality of the theoretical posterior.

This behavior most likely represents a local minimum in the reverse KL loss that is common to many inference techniques [for instance consider multiple non-mixing chains for MCMC 18]. We note that training in forward KL wouldn't provide such a flexibility in that setup, as it would enforce the multi-modality of the posterior, even at the cost of an overall worst result (as it is the case for NPE-C and SNPE-C in fig. 3 and fig. B.3). Indeed, let's imagine that our training dataset features M' draws similar to the one in fig. B.4. Out of randomness, the labelling l of the 3 blobs of points would be permuted across those M' examples. A density estimator would then most likely attempt to model a multi-modal posterior.

Though it is not similar to the theoretical result, we argue that our result is of experimental value, and close to the intuition one forms of the problem: using our results one can readily estimate the original components for the mixture. Indeed, for argument's sake, say we would recover the theoretical, roughly trimodal posterior. To recover the original mixture components, one would need to split the 3 modes of the posterior and arbitrarily assign a label l to each one of the modes. In that sense, our posterior naturally features this splitting, and can be used directly to estimate the $L = 3$ mixture components.

C Complements to the neuroimaging experiment

This section is a complement to the experiment described in section 3.3, we thus consider the model described in eq. (11) and eq. (12). We present a toy dimension version of our experiment, useful to build an intuition of the problem. We also present implementation details for our experiment, and additional neuroimaging results.

C.1 Descriptors, inputs to ADAVI

We can analyse the model decribed in eq. (11) and eq. (12) using the descriptors defined in eq. (2). Those descriptors constitute the inputs our methodology needs to automatically derive the *dual* architecture from the generative HBM:

$$
\begin{aligned}
\mathcal{V} &= \{M^g, \epsilon, M^s, \sigma, M^{s,t}, \kappa, \Pi, X\} \\
\mathcal{P} &= \{\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2\} \\
\text{Card} &= \{\mathbf{P}_0 \mapsto N, \mathbf{P}_1 \mapsto T, \mathbf{P}_2 \mapsto S\} \\
\text{Hier} &= \{M^g \mapsto 3, \epsilon \mapsto 3, M^s \mapsto 2, \sigma \mapsto 3, M^{s,t} \mapsto 1, \kappa \mapsto 3, \Pi \mapsto 3, X \mapsto 0\} \\
\text{Shape} &= \{ \\
&\quad M^g \mapsto (L, D), \\
&\quad \epsilon \mapsto (L,), \\
&\quad M^s \mapsto (L, D), \\
&\quad \sigma \mapsto (L,), \\
&\quad M^{s,t} \mapsto (L, D), \\
&\quad \kappa \mapsto (1,), \\
&\quad \Pi \mapsto (L,), \\
&\quad X \mapsto (D,) \\
&\} \\
\text{Link} &= \{ \\
&\quad M^g \mapsto \mathcal{L} \circ \text{Reshape}((LD,) \to (L, D)), \\
&\quad \epsilon \mapsto \text{Exp}, \\
&\quad M^s \mapsto \mathcal{L} \circ \text{Reshape}((LD,) \to (L, D)), \\
&\quad \sigma \mapsto \text{Exp}, \\
&\quad M^{s,t} \mapsto \mathcal{L} \circ \text{Reshape}((LD,) \to (L, D)), \\
&\quad \kappa \mapsto \text{Exp}, \\
&\quad \Pi \mapsto \text{SoftmaxCentered}((L-1,) \to (L,)), \\
&\quad X \mapsto \mathcal{L} \\
&\}
\end{aligned}
\tag{C.7}
$$

C.2 Experiment on MS-HBM model on toy dimensions

To get an intuition of the behavior of our architecture on the MS-HBM model, we consider the following toy dimensions reproduction of the model:

$$
\begin{aligned}
N, T, S, D, L &= 50, 2, 2, 2, 2 \\
g^-, g^+ &= -4, 4 \\
\kappa^- &= -4 \\
\kappa^+ = \sigma^- &= -3 \\
\sigma^+ = \epsilon^- &= -2 \\
\epsilon^+ &= -1 \\
\pi &= 2 \\
\mathcal{L}^{-1}1(\mu_l^g) &\sim \text{Uniform}_{D-1}(-g^-, g^+) \\
\text{Log}(\epsilon_l) &\sim \text{Uniform}_L(\epsilon^-, \epsilon^+) \\
\mathcal{L}^{-1}1(\mu_l^s)|\mu_l^g,\ \epsilon_l &\sim \mathcal{N}(\mathcal{L}^{-1}(\mu_l^g), \epsilon_l) \\
\text{Log}(\sigma_l) &\sim \text{Uniform}_L(\sigma^-, \sigma^+) \\
\mathcal{L}^{-1}1(\mu_l^{s,t})|\mu_l^s,\ \sigma_l &\sim \mathcal{N}(\mathcal{L}^{-1}(\mu_l^s), \sigma_l) \\
\text{Log}(\kappa) &\sim \text{Uniform}_1(\kappa^-, \kappa^+) \\
\Pi &\sim \text{Dirichlet}([\pi] \times L) \\
\mathcal{L}^{-1}1(X_n^{s,t})|[\mu_1^{s,t},\ \dots\ , \mu_L^{s,t}], \kappa, \Pi &\sim \text{Mixture}(\Pi, [\mathcal{N}(\mathcal{L}^{-1}(\mu_1^{s,t}), \kappa),\ \dots\ , \mathcal{L}(\mathcal{L}^{-1}(\mu_L^{s,t}), \kappa)])
\end{aligned}
\tag{C.8}
$$

We use a Hierarchical encoder featuring 2 set transformers (encoding size of 32; split in 4 attention heads of size 8; encoder with 2 SAB blocks; decoder with 1 seed vector). For the conditional density estimators, we use a flow comprised of an affine block with diagonal scale, followed by a MAF with hidden dimensions [32]. We use the Adam optimizer (10^{-3} learning rate), minibatch gradient descent (batch size 32, 32 θ^m draws per X^m) over a training dataset of size $20,000$. Training took under 10 minutes using the 3-fold training strategy described in appendix C.3.

The results can be visualized on fig. C.5. This experiment shows the expressivity we gain from the usage of link functions.

C.3 Implementation details for the neuro-imaging MS-HBM example

Main implementation differences with the original MS-HBM model Our implementation of the MS-HBM (eq. (11) and eq. (12)) contains several notable differences with the original one from Kong et al. [21]:

- we model μ distributions as Gaussians linked to the positive quadrant of the unit sphere via the function $\mathcal{L}$. In the orignal model, RVs are modelled using *Von Mises Fisher* distributions. Our choice allows us to express the entirety of the connectivity vectors (that only lie on a portion of the unit sphere). However, we also acknowledge that the densities incurred by the 2 distributions on the positive quadrant of the unit sphere are different.
- we forgo any spatial regularization, and also the assumption that the parcellation of a given subject s should be constant across sessions t. This is to streamline our implementation. Adding components to the loss optimized at training could inject those constraints back into the model, but this was not the subject of our experiment, so we left those for future work.

Data pre-processing and dimensionality reduction Our model was able to run on the full dimensionality of the connectivity, $D^0 = 1483$. However, we obtained better results experimentally when further pre-processing the used data down to the dimension $D^1 = 141$. The displayed results in fig. 4 are the ones resulting from this dimensionality reduction:

1. we projected the (S, T, N, D^0) X connectome (lying on the D^0 unit sphere) to the unbounded $\mathbb{R}^{D^0-1}$ space using the function $\mathcal{L}$

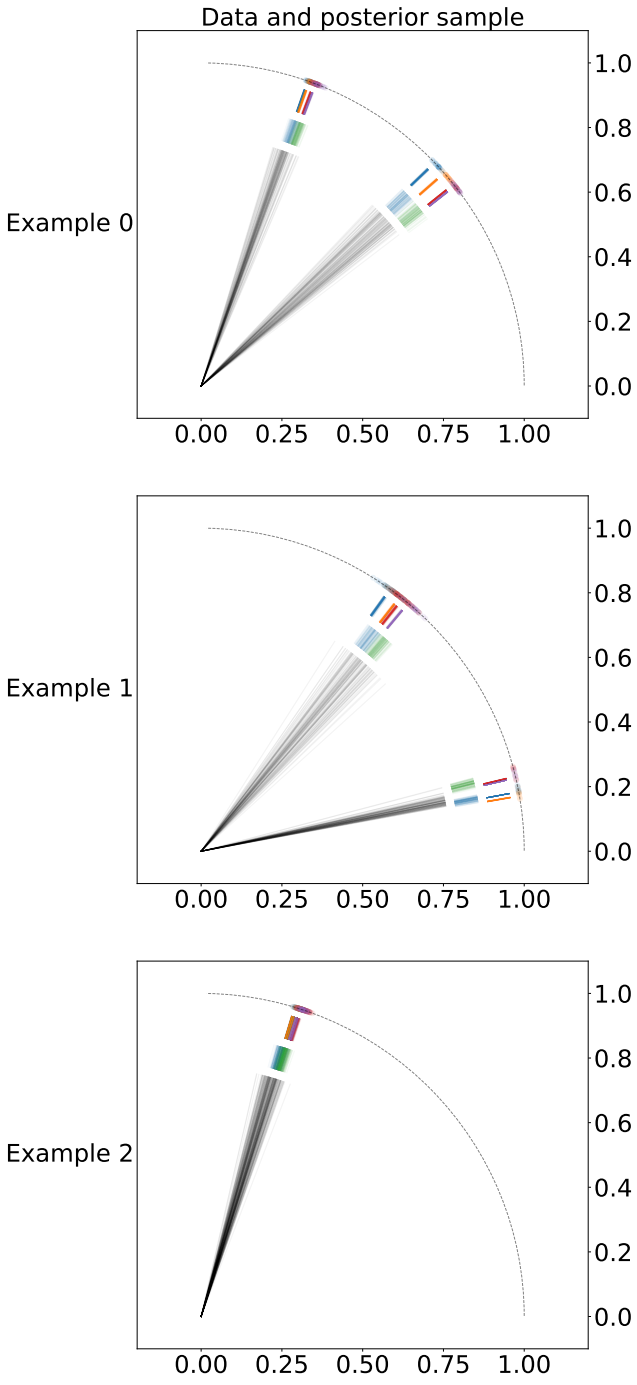

Figure C.5: Visual representation of our results on a synthetic MS-HBM example. Data is represented as colored points on the unit positive quadrant, each color corresponding to a subject $\times$ session. Samples from posterior distributions are represented as concentric colored markings. Just below the data points are $\mu^{s,t}$ samples. Then samples of μ^s. Then samples of μ^g (black lines). Notice how the μ posteriors are distributed around the angle bisector of the arc covered by the points at the subsequent plate.

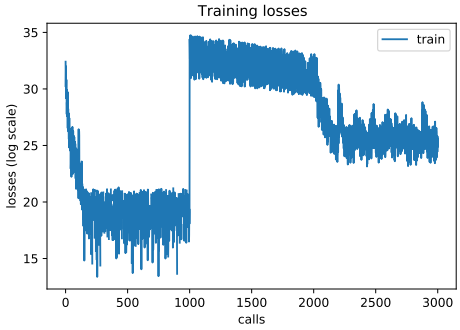

Figure C.6: 3-step loss evolution across epochs for the MS-HBM ADAVI training. Losses switch are visible at epochs 1000 and 2000. Training was run for a longer period after epoch 3000, with no significant results difference.

2. in this $\mathbb{R}^{D^0-1}$ space, we performed a Principal Component Analysis (PCA) to bring us down to $D^1 - 1 = 140$ dimensions responsible for 80% of the explained data variance
3. in the resulting $\mathbb{R}^{D^1-1}$ space, we calculated the mean of all the connectivity points, and their standard deviation, and used both to whiten the data
4. from the whitened data, we calculated the Ledoit-Wolf regularised covariance [23], that we used to construct the Σ_g matrix used in eq. (11)
5. finally, we projected the whitened data onto the unit sphere in $D^1 = 141$ dimensions via the function $\mathcal{L}$

To project our results back to the original D^0 space, we simply ran back all the aforementioned steps. Our prior for μ^g has been carefully designed so has to sample connectivity points in the vicinity of the data point of interest. Our implementation is therefore in spirit close to SBI [7; 14; 27; 34] that aims at obtaining an amortized posterior only in the relevant data regime.

Mutli-step training strategy In section 3.3 we described our conditional density estimators as the stacking of a MAF [25] on top of a diagonal-scale affine block. To accelerate the training of our architecture and minimize numerical instability (resulting in NaN evaluations of the loss) we used the following 3-step training strategy:

1. we only trained the *shift* part of our affine block into a Maximum A Posteriori regression setup. This can be viewed as the amortized fitting of the first moment of our posterior distribution
2. we trained both the *shift* and *scale* of our affine block using an *unregularized ELBO* loss. This is to rapidly bring the variance of our posterior to relevant values
3. we then trained our full posterior (*shift* and *scale* of our affine block, in addition to our MAF block) using the *reverse KL* loss.

This training strategy shows the modularity of our approach and the transfer learning capabilities already introduced in appendix A.3. Loss evolution can be seen in fig. C.6

Soft labelling In eq. (11) and eq. (12) we define μ variables as following Gaussian distributions in the latent space $\mathbb{R}^{D^1-1}$. This means that, considering a vertex $X_n^{s,t}$ and a session network $\mu_k^{s,t}$, the squared Euclidean distance between $\mathcal{L}^{-1}(X_n^{s,t})$ and $\mathcal{L}^{-1}(\mu_k^{s,t})$ in the latent space $\mathbb{R}^{D^1-1}$ is proportional to the log-likelihood of the point $\mathcal{L}^{-1}(X_n^{s,t})$ for the mixture component k:

$$\|\mathcal{L}^{-1}(X_n^{s,t}) - \mathcal{L}^{-1}(\mu_k^{s,t})\|^2 = \log p(X_n^{s,t}|l=k) + C(\kappa) \tag{C.9}$$

Note that κ is the same for both networks. Additionally, considering Bayes theorem:

$$\begin{aligned} \log p(l=k|X_n^{s,t}) &= \log p(X_n^{s,t}|l=k) + \log p(l=k) - \log p(X_n^{s,t}) \\ \log \frac{p(l=0|X_n^{s,t})}{p(l=1|X_n^{s,t})} &= \log p(X_n^{s,t}|l=0) - \log p(X_n^{s,t}|l=1) + \log \frac{p(l=0)}{p(l=1)} \end{aligned} \tag{C.10}$$

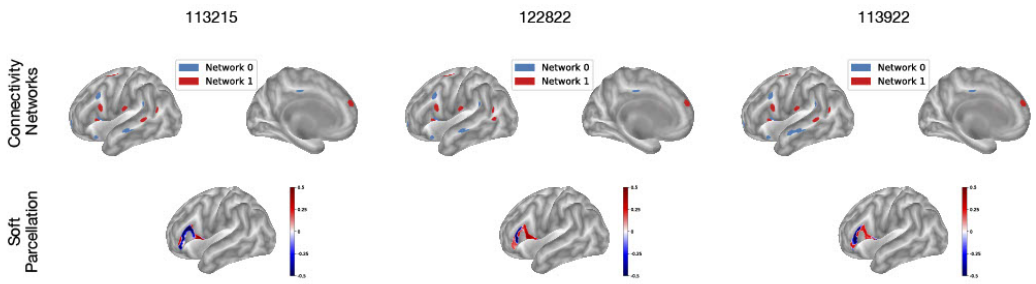

Figure C.7: Subject-level parcellations and networks. For 3 different HCP subjects, we display the individual parcellation (on the bottom) and the individual μ^s networks (on the top). Note how the individual parcellations, though showing the same general split between the *pars opercularis* and *pars triangularis*, slightly differ from each other and from the population parcellation (fig. 4). Similarly, networks μ^s differ from each other and from the population networks μ^g (fig. 4) but keep their general association to semantic/phonologic processing (0, in blue) and language production (1, in red) [15; 42]. To be able to model and display such variability is one of the interests of models like the MS-HBM [21].

Where $\log p(X_n^{s,t}|l = k)$ can be obtained through eq. (C.9) and $\log p(l = k)$ via draws from the posterior of Π (see eq. (11)). To integrate those equations, we used a Monte Carlo procedure.

C.4 Additional neuro-imaging results

As pointed out in section 3.3, the MS-HBM model aims at representing the functional connectivity of the brain at different levels, allowing for estimates of population characteristics and individual variability [21]. It is of experimental value to compare the parcellation for a given subject, that is to say the soft label we give to a vertex $X_n^{s,t}$, and how this subject parcellation can deviate from the population parcellation. Those differences underline how an individual brain can have a unique local organization. Similarly, we can obtain the subject networks μ^s and observe how those can deviate from the population networks μ^g. Those results underline how a given subject can have his own connectivity, or, very roughly, his own "wiring" between different areas of the brain. Results can be seen in fig. C.7.