Flexibly Regularized Mixture Models and Application to Image Segmentation

Jonathan Vacher[a], Claire Launay[b], Ruben Coen-Cagli[b,c]

[a]*Laboratoire des Systèmes Perceptif, Département d'Études Cognitives, École Normale Supérieure, PSL University, 24 rue Lhomond, Bâtiment Jaurès, 2éme étage, Paris, 75005, , France*
[b]*Dept. of Systems and Comp. Biology, Albert Einstein College of Medicine, 1300 Morris Park Ave, Bronx, 10461, NY, USA*
[c]*Dominick P. Purpura Dept. of Neuroscience, Albert Einstein College of Medicine, 1300 Morris Park Ave, Bronx, 10461, NY, USA*

Abstract

Probabilistic finite mixture models are widely used for unsupervised clustering. These models can often be improved by adapting them to the topology of the data. For instance, in order to classify spatially adjacent data points similarly, it is common to introduce a Laplacian constraint on the posterior probability that each data point belongs to a class. Alternatively, the mixing probabilities can be treated as free parameters, while assuming Gauss-Markov or more complex priors to regularize those mixing probabilities. However, these approaches are constrained by the shape of the prior and often lead to complicated or intractable inference. Here, we propose a new parametrization of the Dirichlet distribution to flexibly regularize the mixing probabilities of over-parametrized mixture distributions. Using the Expectation-Maximization algorithm, we show that our approach allows us to define any linear update rule for the mixing probabilities, including spatial smoothing regularization as a special case. We then show that this flexible design can be extended to share class information between multiple mixture models. We apply our algorithm to artificial and natural image segmentation tasks, and we provide quantitative and qualitative comparison of the performance of Gaussian and Student-t mixtures on the Berkeley Segmentation Dataset. We also demonstrate how to propagate class information across the layers of deep convolutional neural networks in a probabilistically optimal way, suggesting a new interpretation for feedback signals in biological visual systems. Our flexible approach can be easily generalized to adapt probabilistic mixture models to arbitrary data topologies.

Keywords: unsupervised learning, mixture models, graphical model, factor graph, image segmentation, convolutional neural networks

Email addresses: jonathan.vacher@ens.psl.eu (Jonathan Vacher), claire.launay@einsteinmed.org (Claire Launay), ruben.coen-cagli@einsteinmed.org (Ruben Coen-Cagli)
URL: https://jonathanvacher.github.io/ (Jonathan Vacher)

1. Introduction

Probabilistic finite mixture models are a class of statistical models that assume the density of observed data is a weighted sum of simpler component distributions. Finite mixture models aggregate data points by their statistical similarity and are widely applied to unsupervised clustering problems [38]. Because these models do not take into account topological dependencies between the observed data points, they often result in scattered clusters of data points. In practice, clustering results are often improved by accounting for the topology of the data [22, 60, 34, 20]. A common approach to achieve this is to introduce a dependence of the class probability of one data point on the class of the other data points, either explicitly or implicitly through structured priors that act effectively as a regularization term to the likelihood function. Here we present a new formulation of finite probabilistic mixtures that allows one to impose any topology on the class probabilities, and that offers a computationally efficient optimization of the model's parameters. To illustrate our approach, we consider unsupervised segmentation of natural images: a paradigmatic application of finite mixture models, in which accounting for the data topology is necessary to achieve good performance.

Segmentation is the task of partitioning an image into multiple areas or segments, or equivalently of assigning segment labels (*i.e.* mixture components, in the language of finite mixtures) to each image pixel. Recent advances in deep learning models allowed the development of successful strategies for supervised segmentation [39, 47, 35, 2, 8]. These approaches are trained with labeled images and are extremely effective, in part because they learn object appearance which implicitly forces those algorithms to respect the spatial topology of images, *i.e.* grouping together nearby pixels. A known limitation of these supervised strategies is that they do not learn to group pixels by their similarity and therefore generalize poorly to images of unseen object categories.

On the other hand, many algorithms for unsupervised segmentation have been developed, using different definitions of similarity between pixels, such as k-means clustering [14], active contours models [26] and graph cuts [6]. Different from those algorithms, finite probabilistic mixture models have two advantages that, as we show in this paper, can be leveraged to improve segmentation and also generalize to many other clustering problems. First, the mixture components represent an explicit model of the statistical distribution of the data points within each segment. Therefore, knowledge about the statistics of natural images [24, 44, 58, 9, 10, 55] can help choosing an appropriate parametric family for the component distributions. Second, the probabilistic formulation offers not just a segmentation of the image, but also a measure of the uncertainty of the assignment of pixels to labels, which can be used to combine optimally multiple cues for segmentation. These properties are important for successful segmentation, because studies of biological visual processing have shown that neurons in the visual cortex of the brain are sensitive to the statistical regularities of natural images [24, 58, 9, 10], and that human perception uses multiple segmentation cues tuned to those statistics [57, 17, 52, 37, 18] and combines them near-optimally [48].

Given the topology of images, existing probabilistic mixture models for segmentation have been extended to encourage the assignment of spatially neighboring pixels to the same mixture component. Many authors have proposed to add a penalty on the posterior class probabilities to enforce their local similarity [34, 20, 22, 60].

Other authors have proposed mixture models in which the prior mixing probabilities depend on the index of the sample (*e.g.* the spatial position of the pixel). The problem with this approach is that, because the mixing probabilities are parameters that have to be learned, it increases the number of model parameters way above the number of samples. A commonly adopted solution is to strongly regularize the mixing probabilities, by accounting for the topology of the dataset. For example, in order to favor grouping of neighboring samples Blei and Frazier [3] have developed the distance dependent Chinese restaurant process (ddCRP). The ddCRP has been further extended to perform image segmentation using a region-based hierarchical representation [21]. Similarly in a series of papers, Nikou *et al.* have proposed to consider the mixing probabilities as Gauss-Markov random fields either directly [50, 42, 51] or hierarchically [43]. More recently and using comparable ideas, Sun *et al.* have introduced Location Dependent Dirichlet Processes (LDDP) [54] which use exponential of Gaussian processes in combination with Dirichlet processes to describe class distributions.

Our work builds on the tools and concepts used by Sun *et al.* [54] and Nikou *et al.* [43], and extends their formulations to address two important limitations. First, those approaches do not allow any flexibility in defining how information about segment assignment is combined across pixels. Furthermore, during inference, this combination is typically nonlinear leading to computationally intensive, and sometimes unstable, optimization. Second, the use of *ad-hoc* solutions for spatial regularization does not provide a clear route for extending those models to different topologies. This is important not only for other clustering problems defined, for instance, on temporal sequences or tree graphs, but also for image segmentation itself: in natural images, there is another topology associated with the hierarchy of visual features, *e.g.* the hidden units at different layers of deep neural networks (DNN). It is well-known from image style-transfer applications [25] that shallow layers convey texture information while the deep layers convey geometric and structural information. Combining that information in a systematic way could improve image segmentation. Studies on human perception have shown that humans are sensitive to segmentation cues at several levels [57], that they can combine segmentation information from multiple levels near-optimally [48], and that high level features, like objects, strongly affect segmentation in human observers [45, 41]. Related work in computer vision has demonstrated that recurrent and feedback signals between feature levels [33, 27, 29] are crucial for perceptual grouping and segmentation. The framework we introduce here extends naturally to this hierarchical structure, and allows for optimal weighting of recurrence and feedback signals.

In the remainder of Section 1, we first briefly recap the formulation of probabilistic mixture models and the EM algorithm used for parameter optimization. Then, as they are closely related to our work, we describe the approaches of Sun *et al.* [54] and Nikou *et al.* [43] which account for image topology by regularizing the class probabilities. We next introduce our formulation and explain its advantages over those approaches. In Section 2 we provide full details on our model for the class probabilities, which requires a directed graphical model with loops and a specific prior parametrization, and we state formally the theoretical results that come with it. Then, we show how our formulation can be readily extended to combine multiple mixture models through their mixing probabilities, thus accounting for hierarchical structure. In Section 3, we illustrate the performance of our model in synthetic and

natural image segmentation tasks.

Notations. We use the following notations. Integers H, N, K and D denote respectively, the number of layers, the number of samples, the number of classes and the dimension. A random variable is denoted by a capital letter X. The probability density function of X is denoted $\mathbb{P}_X$ while x_n denotes a sample. The set Δ^K represents the K-dimensional simplex. A bold letter (lowercase or capital) is a collection of K variables $\mathbf{b} = (b_1, \dots, b_K)$. A set of N samples $(x_n)_{1 \leqslant n \leqslant N}$ is shortened by $(x_n)_n$ or $x_.$. This notation also holds for random variables.

1.1. Probabilistic mixture models and Expectation-Maximization

A sample $x_n \in \mathbb{R}^D$ is a D-dimensional feature vector which is a realization of a random vector X_n. When the distribution of X_n is a weighted sum of $K > 0$ distributions, we say that X_n follows a mixture distribution with K components. Each random variable X_n is associated with a discrete latent random variable C_n denoting its class among the K components.

Specifically, we consider mixtures of parametric distributions, *i.e.* for all $k \in \{1, \dots, K\}$, the distribution of any observed variable with class $C_n = k$ belongs to the same parametric family (Gaussian, Exponential, ...). In the following, we write the distribution parameter (or set of parameters) associated with the class k as a_k and the probability of a sample x_n given its class $C_n = k$ as $\mathbb{P}_{X^{(k)}}(x_n; a_k)$.

Standard probabilistic mixture models [38] assume that the latent random variables $(C_n)_n$ are i.i.d. and follow a multinomial distribution, *i.e.* for all $n \in \{1, \dots, N\}$, for all $k \in \{1, \dots, K\}$,

$$\mathbb{P}_{C_n}(k) = p_k, \tag{1}$$

where $\mathbf{p} = (p_1, \dots, p_K) \in \Delta^K$ is the vector of class probabilities, also called the mixing probabilities. Thus, given the mixing probabilities $\mathbf{p}$ and the collection of distribution parameters $\mathbf{a} = (a_1, \dots, a_K)$, the density function writes

$$\mathbb{P}_{X|\mathbf{P},\mathbf{A}}(x|\mathbf{p}, \mathbf{a}) = \sum_{k=1}^{K} p_k \mathbb{P}_{X^{(k)}}(x; a_k). \tag{2}$$

The graphical model of those mixture models is shown in Figure 1a.

Given samples $(x_n)_n$, the Maximum Likelihood (ML) or Maximum A Posteriori (MAP) estimates of the model parameters $\boldsymbol{\theta} = (\mathbf{p}, \mathbf{a})$ can be found using the Expectation-Maximization (EM) algorithm. The EM algorithm is an iterative optimization method which proceeds in two steps: (i) the expectation step (E-step) which aims at estimating the objective function knowing previous parameter estimates; (ii) the maximization step (M-step) which aims at maximizing the objective function estimated during the E-step in order to update the previous parameter estimates.

One approach to arrive at these two-steps is to consider the likelihood of the samples completed by their class $((x_n, C_n))_n$ *i.e.*

$$\begin{aligned} \ell\left(\mathbf{p}, \mathbf{a}; (x_n, C_n)_n\right) &= \ln \left(\prod_{n=1}^{N} \mathbb{P}_{X,C|\mathbf{P},\mathbf{A}}(x_n, C_n|\mathbf{p}; \mathbf{a}) \right) \\ &= \ln \left(\prod_{n=1}^{N} \prod_{k=1}^{K} \left(p_k \mathbb{P}_{X^{(k)}}(x_n; a_k)\right)^{\mathbb{1}_k(C_n)} \right) \end{aligned} \tag{3}$$

where

$$\mathbb{1}_j(i) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Such a completion turns the sum in Equation (2) into a product which in turn will behave nicely when considering the log-likelihood. Indeed Equation (3) becomes

$$\ell\left(\mathbf{p}, \mathbf{a}; (x_n, C_n)_n\right) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}_k(C_n) \ln(p_k) + \mathbb{1}_k(C_n) \ln(\mathbb{P}_{X^{(k)}}(x_n; a_k)). \tag{5}$$

The cost of this completion is to introduce unknown class variables $(C_n)_n$ which makes the log-likelihood ℓ a random variable preventing its direct maximization. This is solved by the two steps of the EM algorithm. Given the previous parameter estimates $\boldsymbol{\theta}^{(t)} = \left(\mathbf{p}^{(t)}, \mathbf{a}^{(t)}\right)$, the E-step estimates the completed-data log-likelihood Q by taking expectation of ℓ knowing $\boldsymbol{\theta}^{(t)}$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{C_n|(X_n)_n, \boldsymbol{\theta}} \left(\ell\left(\mathbf{p}, \mathbf{a}; (x_n, C_n)_n\right) | (x_n)_n, \boldsymbol{\theta}^{(t)} \right). \tag{6}$$

In practice, the E-step amounts to computing for all $n \in \{1, \ldots, N\}$, for all $k \in \{1, \ldots, K\}$, the posterior class probabilities

$$\tau_{n,k}^{(t)} = \mathbb{P}_{C_n|X_n, \boldsymbol{\theta}}(k|x_n, \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{C_n|X_n, \boldsymbol{\theta}}(\mathbb{1}_k(C_n)|x_n, \boldsymbol{\theta}^{(t)}). \tag{7}$$

Then, the M-step updates the previous parameter estimates by maximizing Q *i.e.*

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}). \tag{8}$$

With an appropriate choice of the component distributions, for instance Gaussian or Exponential, closed-form maximum likelihood estimates of the component parameters $\mathbf{a}$ are available. In addition, the update rule for the mixing probabilities does not depend on the choice of the component distribution and it writes

$$p_k^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} \tau_{n,k}^{(t)}. \tag{9}$$

This procedure was introduced by Dempster *et al.* [13] which proved that the likelihood is non-decreasing at each iteration of the EM algorithm. There are no general guarantees that the sequence $\{\boldsymbol{\theta}^{(t)}\}$ converges to a maximum likelihood estimator. Under some conditions verified by many models, the EM algorithm converges to a stationary value of the complete-data log-likelihood function while the convergence of the sequence $\{\boldsymbol{\theta}^{(t)}\}$ to a point $\boldsymbol{\theta}^*$ requires stronger conditions [59, 7].

1.2. Previous work and contributions

As explained above, an important limitation of mixture models is that they assume independence between samples, and therefore ignore the underlying topology of the dataset. For instance, considering image segmentation, when the samples are pixels of an image, the location of a pixel and the classes assigned to its neighbors may provide information about the class assignment of that pixel. We consider the

approach in which the mixing probabilities depend on the index n of the sample (*i.e.* $\mathbf{p}$ becomes $\mathbf{p}_n$), and therefore the mixture model writes

$$\mathbb{P}_{X_n|\mathbf{P}_n,\mathbf{A}}(x|\mathbf{p}_n,\mathbf{a}) = \sum_{k=1}^{K} p_{n,k}\mathbb{P}_{X^{(k)}}(x;a_k), \tag{10}$$

where for all $n \in \{1,\dots,N\}$, $\mathbf{p}_n \in \Delta^K$.

The model now is over-parametrized, but the topology of the data can be exploited to regularize the mixing probabilities. Specifically, our work is closely related to the approaches of Sun *et al.* [54] and Nikou *et al.* [43]. Therefore, in the following we summarize their work before introducing ours.

Preliminary definitions. First, we say that a random variable R follows a Gamma distribution knowing the parameter S when its density writes

$$\mathbb{P}_{R|S}(r|s) = \frac{r^{s-1}\exp(-r)}{\Gamma(s)}. \tag{11}$$

We denote $R \sim \mathcal{G}(S)$. Then, we say that a random vector $\mathbf{R} \in \Delta^K$ follows a Dirichlet distribution knowing the parameters $\mathbf{S}$ when its density writes

$$\mathbb{P}_{\mathbf{R}|\mathbf{S}}(\mathbf{r}|\mathbf{s}) = \frac{\Gamma\left(\sum_{k=1}^{K} s_k\right)}{\prod_{k=1}^{K}\Gamma(s_k)}\prod_{k=1}^{K} r_k^{s_k-1}. \tag{12}$$

We denote $\mathbf{R} \sim \mathcal{D}(\mathbf{S})$. Then, a function $F : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is a Gaussian process with mean $\mu : \mathbb{R}^2 \longrightarrow \mathbb{R}$ and covariance $\Sigma : \mathbb{R}^2 \times \mathbb{R}^2 \longrightarrow \mathbb{R}$ when for any $N > 0$ and any locations $(l_1,\dots,l_N) \in \mathbb{R}^{2\times N}$, $(F(l_n))_n \sim \mathcal{N}\left(\bar{\mu},\bar{\Sigma}\right)$ where $\bar{\mu} = (\mu(l_n))_n$ and $\bar{\Sigma} = (\Sigma(l_m,l_n))_{m,n}$. We denote $F \sim \mathcal{GP}(\mu,\Sigma)$. The Gaussian process F is a Gauss-Markov process when for all $(m,n) \in \{1,\dots,N\}^2$ such that $l_m \notin \mathcal{C}_n$, $\Sigma^{-1}(l_m,l_n) = 0$ where $\mathcal{C}_n$ is a neighborhood of l_n. We denote $F \sim \mathcal{GMP}(\mu,\Sigma)$. The Gaussian process F is stationary when μ is constant and $\Sigma(u,v) = \Sigma(u-v)$. We denote $F \sim \mathcal{SGP}(\mu,\Sigma)$.

Location-dependent Dirichlet Process. In the work of Sun *et al.* [54] the mixing probabilities are modeled as:

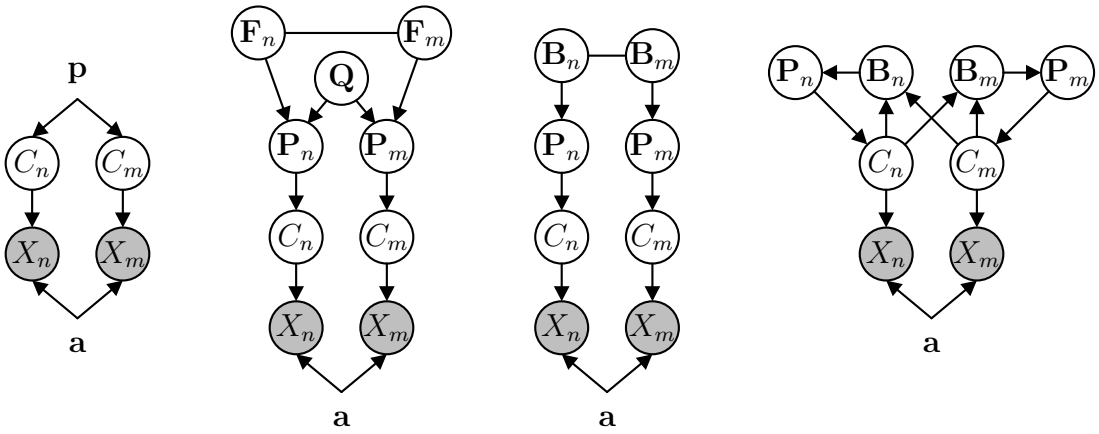
$$P_{n,k} = \frac{Q_k \exp(F_k(l_n))}{\sum_{i=1}^{K} Q_i \exp(F_i(l_n))} \tag{13}$$

where

$$\forall k \in \{1,\dots,K\},\ Q_k \sim \mathcal{G}(b) \quad \text{and} \quad F_k \sim \mathcal{SGP}(0,\Sigma)$$

with $P_{n,k} = P_k(l_n)$ and $b > 0$. The associated graphical model is shown in Figure 1b.

In the work of Sun *et al.*, model training is achieved using variational inference. However, for the sake of comparison we have derived the EM update rule for the mixing probabilities (Table 1, left). These equations highlight how information about neighboring pixels is combined to obtain mixing probabilities. This formulation, involving additional variables, is more complex and difficult to interpret than ours.



(a) Standard mixture model.

(b) Location Dependent Dirichlet Process

(c) Spatially varying mixture models.

(d) Proposed mixture model.

Figure 1: *Probabilistic graphical models corresponding to previously proposed mixture models (a-c) and our model (d) for two observed variables X_n and X_m. Circles represent random variables (gray: observed; white: latent), uncircled letters represent parameters, directed and undirected edges represent probabilistic dependencies. The variables are described in the main text.*

Spatially-varying mixtures. In the work of Nikou *et al.* [43] the mixing probabilities are modeled as

$$\mathbf{P}_n = \mathbf{P}(l_n) \sim \mathcal{D}(\mathbf{B}(l_n)) \quad \text{where} \quad \forall k \in \{1, \dots, K\},\ B_k \sim \mathcal{GMP}(0, \Sigma_k). \tag{14}$$

The associated graphical model is shown in Figure 1c. The EM update rule for the mixing probabilities consists in solving a third order polynomial equation (Table 1, middle), which can cause numerical instabilities.

Our model: Flexibly regularized mixture models (FlexMM). We assume that the random vector $\mathbf{P}_n$ follows a Dirichlet distribution whose parameter $\mathbf{B}_n \in \mathbb{R}^K$ depends linearly on the classes $(C_n)_n$. In other words, the Dirichlet parameter pulls information from the classes of other samples to regularize $\mathbf{P}_n$. Our model for the mixing probabilities writes, for all $n \in \{1, \dots, N\}$,

$$\mathbf{P}_n = \mathbf{P}(l_n) \sim \mathcal{D}(\mathbf{B}(l_n)) \tag{15}$$

where

$$\forall k \in \{1, \dots, K\},\ B_k(l_n) = u_{n,k}((\mathbb{1}_k(C_n))_n) - \mathbb{1}_k(C_n) + 1 \tag{16}$$

with $u_{n,k} : \mathbb{R}^N \longrightarrow \mathbb{R}$ is a linear function such that $u_{n,k}\left([0, +\infty[^N\right) \subset \mathbb{R}_+$. The graphical model associated with our model is shown in Figure 1d. Importantly, in contrast to [54, 43], in our model the update rule depends linearly on the posterior class probabilities and is entirely determined by the functions $u_{n,k}$ (Table 1, right; see next section for the derivation).

Therefore, our formulation has the advantage of guaranteeing linear updates while maintaining full flexibility in how class information is integrated across samples. Yet, this comes at the cost of introducing loops in the graphical model. In the following section, first, we justify that our loopy model is well-defined, providing a detailed description, and we state formally the theoretical results which comes

with it. Then, we show how the flexible update rule allows one to combine multiple mixture models through their mixing probabilities.

	Sun *et al.* [54]	Nikou *et al.* [43]	Ours
Mixing probabilities update	$q_k^{(t+1)} = b - 1 + \sum_{n=1}^{N} \tau_{n,k}^{(t)} - p_{n,k}^{(t)}$ $f_{\cdot,k}^{(t+1)} = \bar{\Sigma}(\tau_{\cdot,k}^{(t)} - p_{\cdot,k}^{(t)})$ $p_{n,k}^{(t+1)} = \frac{q_k^{(t+1)} \exp\left(f_{n,k}^{(t+1)}\right)}{\sum_{i=1}^{K} q_i^{(t+1)} \exp\left(f_{n,i}^{(t+1)}\right)}$	$p_{n,k}^{(t+1)}$ is the root of a 3rd order polynomial which depends on the posterior class probabilities $\tau_{n,k}^{(t)}$.	$p_{n,k}^{(t+1)} = \frac{u_{n,k}(\tau_{\cdot,k}^{(t)})}{\sum_{k=1}^{K} u_{n,k}(\tau_{\cdot,k}^{(t)})}$

Table 1: Comparison of the mixing probabilities update rule of our model to the ones of previous models. We use the notation $p_{n,k} = p_k(l_n)$. These update rules should be compared to the standard mixture update rule given in the previous subsection: $p_k^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} \tau_{n,k}^{(t)}$.

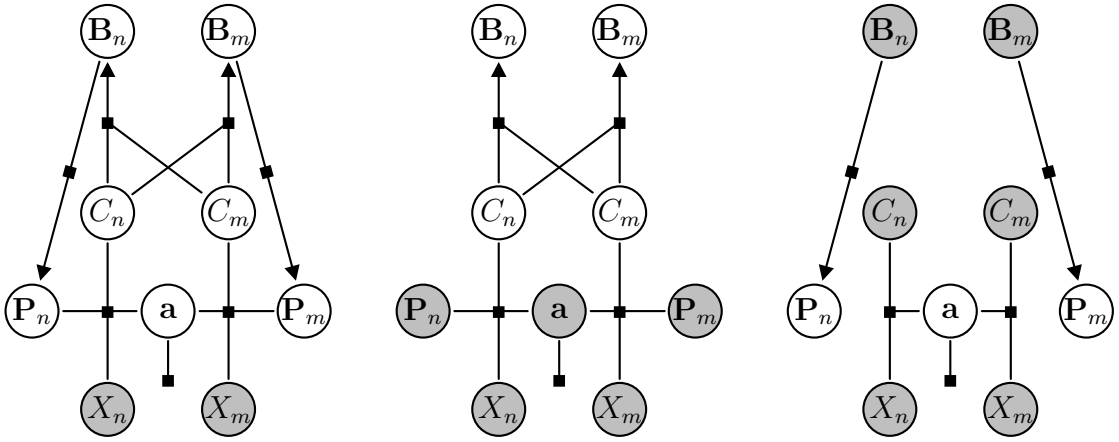
2. Linear Update of the Mixing Probabilities

2.1. Single mixture model

As explained in the previous section, the method we propose to regularize the over-parametrized mixing probabilities $\mathbf{p}_n$ introduces loops in our graphical model (Figure 1d). Such loops could be problematic as they may be inconsistent or prevent one from defining a joint distribution. To circumvent these issues, we identify our directed graph with a factor graph.

A variant of the EM algorithm, called the factor graph EM algorithm [15], was developed to obtain ML or MAP estimates for models associated with a graph containing cycles or loops. Factor graphs [19, 40] are graphical models introduced to explicitly represent arbitrary factorization of the joint distribution. A factor graph has two type of nodes. Variables, either known or hidden, are often represented by letters in circles and factors, which define relations between variables, are represented by black squares and are associated with factor functions given in the factorization. Eckford and Pasupathy [15, 16] state that this EM algorithm on factor graphs is able to break cycles and infer the parameters of the model. The key of this strategy is to extract from the initial factor graph two subgraphs. The first subgraph contains the hidden random variables of the model that are estimated during the E-step, and it is therefore called the E-step factor graph. The second subgraph contains the parameters of the model that are estimated during the M-step of the algorithm, and is called the M-step factor graph. Note that in that case, each step of the EM algorithm can also be seen as local message computations [12, 11], which are at the core of message passing algorithms. Eckford [16] shows that if both subgraphs are cycle-free, then the EM algorithm can be implemented exactly. Eckford [16] and Dauwels *et al.* [12] show that using a likelihood function computed from the model, one can define the function Q adapted to the factor graph that will be computed and maximized iteratively during the EM algorithm.

Thanks to the Hammersley-Clifford theorem [28], we know that the joint distribution of the hidden variables C, $\mathbf{B}$, the observations X and the parameters $\mathbf{p}, \mathbf{a}$ is



(a) Initial factor graph (b) E-factor graph (c) M-factor graph

Figure 2: *Factor graphs associated with our model, the factorization defined in* (17) *and with each step of the EM algorithm. The graphs connect each factor node (black squares) to the corresponding variables (gray circles: observed variable; white circles: latent variables) as given in Equations (17-21). Arrows indicate conditional probabilities.*

proportional to the product of potential functions on the cliques of the associated graph. We choose the following factorization associated to our model

$$\begin{aligned}\mathbb{P}_{X,C,\mathbf{B},\mathbf{P},\mathbf{A}}&((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n,(\mathbf{p}_n)_n,\mathbf{a})\\ &= \frac{1}{Z} f_{\mathbf{A}}(\mathbf{a}) \prod_{n=1}^{N} f_{X,C,\mathbf{A},\mathbf{P}}(x_n,C_n,\mathbf{a},\mathbf{p}_n) f_{\mathbf{P},\mathbf{B}}(\mathbf{p}_n,\mathbf{B}_n) f_{\mathbf{B},(C_n)_n}(\mathbf{B}_n,(C_n)_n),\end{aligned} \tag{17}$$

where the potential functions are chosen to be equal to the following conditional probabilities,

$$f_{X,C,\mathbf{A},\mathbf{P}}(x_n,C_n,\mathbf{a},\mathbf{p}_n) = \mathbb{P}_{X,C|\mathbf{A},\mathbf{P}}(x_n,C_n|\mathbf{a},\mathbf{p}_n) \tag{18}$$

$$f_{\mathbf{P},\mathbf{B}}(\mathbf{p}_n,\mathbf{B}_n) = \mathbb{P}_{\mathbf{P}|\mathbf{B}}(\mathbf{p}_n|\mathbf{B}_n) \tag{19}$$

$$f_{\mathbf{B},(C_n)_n}(\mathbf{B}_n,(C_n)_n) = \mathbb{P}_{\mathbf{B}|(C_n)_n}(\mathbf{B}_n|(C_n)_n) \tag{20}$$

$$f_{\mathbf{A}}(\mathbf{a}) = \mathbb{P}_{\mathbf{A}}(\mathbf{a}), \tag{21}$$

and where Z is normalization constant. Figure 2a presents the factor graph associated with this factorization and our initial model, using the same notations as the seminal papers on factor graphs [30, 19]. The E-step of the EM algorithm computes an expectation given the parameters estimated during the previous iteration, which correspond to an implicit inference of the hidden variables $(C_n)_n$ and $(\mathbf{B}_n)_n$. The M-step corresponds to the maximization of this expectation to obtain new estimates for $\mathbf{p}$ and $\mathbf{a}$. Given the factorization and the computations done during the E-step and the M-step, both subgraphs associated with each step of the EM algorithm are presented in Figure 2. Note that there is no loop in either subgraph. Thus, the proofs in [15, 12] still hold and the EM algorithm is guaranteed to iteratively increase the incomplete log-posterior.

The estimation of the component parameter $\mathbf{a}$ is independent from the estimation of the mixing probabilities $\mathbf{p}_n$. Therefore, any prior $\mathbb{P}_{\mathbf{A}}$ can be used for the

component parameters $\mathbf{a}$. In the EM approach, we consider the joint probability of x_n and C_n. Hence, the log-posterior distribution of $(\mathbf{p}, \mathbf{a})$ writes

$$\begin{aligned}\ell(\mathbf{p}, \mathbf{a}; (x_n, C_n, \mathbf{B}_n)_n) &= \ln(\mathbb{P}_{\mathbf{P},\mathbf{A}|X,C,\mathbf{B}}(\mathbf{p}, \mathbf{a}|(x_n)_n, (C_n)_n, (\mathbf{B}_n)_n)) \\ &= \ln(\mathbb{P}_{X,C,\mathbf{B},\mathbf{P},\mathbf{A}}((x_n)_n, (C_n)_n, (\mathbf{B}_n)_n, (\mathbf{p}_n)_n, \mathbf{a})) - \ln(\mathbb{P}_{X,C,\mathbf{B}}((x_n)_n, (C_n)_n, (\mathbf{B}_n)_n) \\ &= \sum_{n=1}^{N} \ln(\mathbb{P}_{X,C|\mathbf{P},\mathbf{A}}(x_n, C_n|\mathbf{p}_n, \mathbf{a})) + \ln(\mathbb{P}_{\mathbf{P}|\mathbf{B}}(\mathbf{p}_n|\mathbf{B}_n)) + \ln(\mathbb{P}_{\mathbf{B}_n|(C_n)_n}(\mathbf{B}_n|(C_n)_n)) \\ &\quad + \ln(\mathbb{P}_{\mathbf{A}}(\mathbf{a})) - \ln(\mathbb{P}_{X,C,\mathbf{B}}((x_n)_n, (C_n)_n, (\mathbf{B}_n)_n) - \ln(Z).\end{aligned} \tag{22}$$

We recall that we only observe realizations of X. Therefore in the EM approach, the unobserved pixel classes $(C_n)_n$ and prior parameters $(B_n)_n$ will be estimated by taking the expectation during the E-step. In this mixture model, the expression of $\mathbb{P}_{X,C|\mathbf{P},\mathbf{A}}$ used in Equation (3) still holds. Then,

$$\mathbb{P}_{X,C|\mathbf{P},\mathbf{A}}(x_n, C_n|\mathbf{p}_n; \mathbf{a}) = \prod_{k=1}^{K} \left(p_{n,k} \mathbb{P}_{X^{(k)}}(x_n; a_k)\right)^{\mathbb{1}_k(C_n)} \tag{23}$$

In addition, we assume that $\mathbf{B}_n$ is entirely characterized by the classes $(C_n)_n$ *i.e.* it has the following Dirac density

$$\mathbb{P}_{\mathbf{B}_n|(C_n)_n}(\mathbf{B}|(C_n)_n) = \delta_{\mathbf{v}_n(\mathbb{1}_1(C_\cdot),\ldots,\mathbb{1}_K(C_\cdot))}(\mathbf{B}), \tag{24}$$

where $\delta_x(y)$ is the Dirac delta distribution and where $\mathbf{v}_n : \mathbb{R}^{N\times K} \mapsto \mathbb{R}^K$ is a linear function. This amounts to consider that $\mathbf{B}_n$ is equal in distribution to $\mathbf{v}_n\left(\mathbb{1}_1(C_\cdot), \ldots, \mathbb{1}_K(C_\cdot)\right)$. In particular, when $\mathbf{v}_n\left(\mathbb{1}_1(C_\cdot), \ldots, \mathbb{1}_K(C_\cdot)\right) = (u_{n,1}(\mathbb{1}_1(C_\cdot)) - \mathbb{1}_1(C_n) + 1, \ldots, u_{n,K}(\mathbb{1}_K(C_\cdot)) - \mathbb{1}_K(C_n) + 1)$, this comes down to assume that the distribution of the probabilistic maps $\mathbf{P}_n$ is given by Equation (15) with parameters given by Equation (16). Finally the log-posterior can be simplified as

$$\begin{aligned}\ell(\mathbf{p}, \mathbf{a}; (x_n, C_n, \mathbf{B}_n)_n) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}_k(C_n)\Big[\ln\left(p_{n,k}\right) + \ln\left(\mathbb{P}_{X^{(k)}}(x_n; a_k)\right)\Big] + \ln(\mathbb{P}_{\mathbf{A}}(\mathbf{a})) \\ &\quad + \ln\left(\mathbb{P}_{\mathbf{P}|\mathbf{B}}\left(\mathbf{p}_n \middle| \mathbf{v}_n\left(\mathbb{1}_1(C_\cdot), \ldots, \mathbb{1}_K(C_\cdot)\right)\right)\right) + W((x_n, C_n, \mathbf{B}_n)_n),\end{aligned} \tag{25}$$

where W is the function that gathers all the terms of ℓ that do not depend on $\mathbf{p}$ or $\mathbf{a}$. From there assuming that $\mathbb{P}_{\mathbf{P}|\mathbf{B}}$ is a Dirichlet distribution as given by Equation (12), it is possible to derive a custom update rule for the mixing probabilities which is applicable to any mixture model as stated in the following proposition.

Proposition 1. *For all $(n, k) \in \Omega_{N,K} = \{1, \ldots, N\} \times \{1, \ldots, K\}$, let $u_{n,k} : \mathbb{R}^N \longrightarrow \mathbb{R}$ be any linear function such that $u_{n,k}\left([0, +\infty[^N\right) \subset \mathbb{R}_+$. If $\mathbf{v}_n\left(\mathbb{1}_1(C_\cdot), \ldots, \mathbb{1}_K(C_\cdot)\right) = (u_{n,1}(\mathbb{1}_1(C_\cdot)) - \mathbb{1}_1(C_n) + 1, \ldots, u_{n,K}(\mathbb{1}_K(C_\cdot)) - \mathbb{1}_K(C_n) + 1)$, then, the mixing probability updates are*

$$\forall (n, k) \in \Omega_{N,K}, \quad p_{n,k}^{(t+1)} = \frac{u_{n,k}(\tau_{\cdot,k}^{(t)})}{\sum_{k=1}^{K} u_{n,k}(\tau_{\cdot,k}^{(t)})}. \tag{26}$$

where $\tau_{n,k}^{(t)} = \mathbb{P}_{C_n|X_n,\boldsymbol{\Theta}}(k|x_n, \boldsymbol{\theta}^{(t)})$ is the k^{th} component posterior probability of sample x_n computed at the previous E-step and $\boldsymbol{\theta}^{(t)}$ is the previous parameter estimate.

Proof. The proof has two steps: (i) take the conditional expectation of the log-posterior (25) knowing the data and the parameters estimated at the last M-step and use the equality $\mathbb{E}((B_{n,k} - 1 + \mathbb{1}_k(C_n))|(x_n)_n, \boldsymbol{\theta}^{(t)}) = u_{n,k}(\tau_{\cdot,k}^{(t)})$; (ii) write the Karush–Kuhn–Tucker condition [5] for $p_{n,k}$. See supplementary section Appendix B for details. □

Defining the distribution of the Dirichlet parameters $\mathbf{B}$ as in (24) enables to adapt the way spatial information is propagated, given a set of linear functions $(u_{n,k})_{n,k}$. It also leads to linear update equations (26) which was our initial goal. These functions can be defined according to the application of the model (such as image segmentation, text categorization, scene classification) and may also depend on the position of the sample they are applied to. In particular, when $u_{n,k}(\tau_{\cdot,k}) = \sum_m \tau_{m,k}$, the update corresponds to the standard mixture model. When $u_{n,k}(\tau_{\cdot,k}) = \tau_{n,k}$, the mixing probabilities will be equal to the component posterior probability of x_n. Finally, when $u_{n,k}(\tau_{\cdot,k}) = G * \tau_{\cdot,k_{|n}}$ where G is any averaging kernel and $*$ is a convolution operator, both adapted to the topology of indexes n, the update corresponds to a local average of the posterior as has been used recently for spatial smoothing [56, 22]. Despite the reformulation of our model using factor graphs, we are still able to compute the quantities needed to perform an EM algorithm and we can prove the following result.

Proposition 2. *The EM algorithm applied to this model is an ascent algorithm, meaning that at each iteration $t+1$, the incomplete log-posterior increases as the parameters are updated:*

$$\ln \mathbb{P}_{\mathbf{P},\mathbf{A}|X}(\mathbf{p}^{(t+1)}, \mathbf{a}^{(t+1)}|(x_n)_n) \geqslant \ln \mathbb{P}_{\mathbf{P},\mathbf{A}|X}(\mathbf{p}^{(t)}, \mathbf{a}^{(t)}|(x_n)_n).$$

The proof of this proposition adapts the proof of the monotone behavior of log-likelihood of the usual EM algorithm [13] to our model, it can be found in Supplementary Section Appendix C. Similarly, the convergence properties of the EM algorithm given in [59] remain valid in our framework, thus the log-posterior function converges to a stationary point.

2.2. Combining mixture models

Real datasets are often composed of multiple feature vectors that have different dimensionality and different number of samples. In addition, multiple feature vectors can be organized according to a natural topology, *e.g.* the activations at different layers of DNNs correspond to different feature vectors and are organized hierarchically. In such settings, a simple approach could be to train several mixture models independently. However, doing so would ignore the topological organization of the feature vectors, and furthermore the interchangeability of mixture components will prevent to gather the results without relabelling. Relabelling is a costly (non-polynomial) counting problem.

We propose here to alleviate these issues by sharing class information across different mixture models, using a strategy similar to the one introduced in the previous Section. The idea is that instead of training multiple mixture models independently, these can be trained in parallel and regularize each other through the mixing probabilities (see Figure 3). As we show below, the update of the mixing probabilities is linear and can be fully specified through the $\mathbf{v}_n^{(h)}$ function, allowing one to decide how to combine the class information from different mixture models.

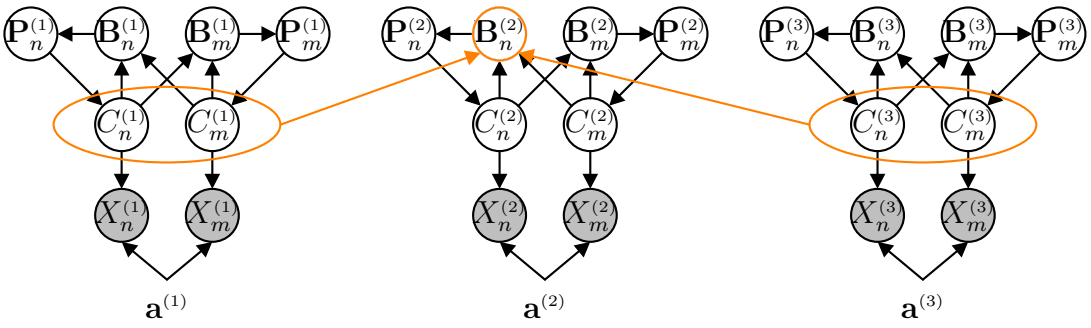


Figure 3: *Mixture model for two observed variables X_n and X_m and 3 layers. Our model enables the combination of the 3 mixtures models, each one associated to a layer. In this graph, we focus on the conditional dependencies of $B_n^{(2)}$ which are determined by $K \times H$ linear functions $(u_{n,k}^{(h)})_{k,h}$.*

We assume that each data sample is associated with a collection of features vectors $(x_n^{(h)})_{h\in\{1,\dots,H\}}$ indexed by the index h. In the case of images, h denotes the DNN layers which contain the activation vector to an image at each pixel location l_n. Again, the Dirichlet parameters $\mathbf{B}_n^{(h)}$ follow a Dirac distribution

$$\mathbb{P}_{\mathbf{B}_n^{(h)}|(C_n^{(h)})_{n,h}}(\mathbf{B}|(C_n^{(h)})_{n,h}) = \delta_{\mathbf{v}_n^{(h)}\left(\mathbb{1}_1(C_\cdot^{(1)}),\dots,\mathbb{1}_1(C_\cdot^{(H)}),\dots,\mathbb{1}_K(C_\cdot^{(1)}),\dots,\mathbb{1}_K(C_\cdot^{(H)})\right)}(\mathbf{B}) \tag{27}$$

where $\mathbf{v}_n^{(h)} : \mathbb{R}^{N\times H\times K} \mapsto \mathbb{R}^K$ is a linear function. As such, this function will set the Dirichlet parameters so that it will regularize the mixing probability $\mathbf{p}_n^{(h)}$ with the class information of the neighboring samples and layers.

As in the previous section, we estimate the parameters of our model using MAP inference

$$(\hat{\mathbf{a}}, \hat{\mathbf{p}}) = \underset{(\mathbf{a},\mathbf{p})}{\operatorname{argmax}} \ \ell\left(\mathbf{p}, \mathbf{a}; (x_n^{(h)}, C_n^{(h)}, \mathbf{B}_n^{(h)})_{n,h}\right) \tag{28}$$

where

$$\ell(\mathbf{p}, \mathbf{a}; (x_n^{(h)}, C_n^{(h)}, \mathbf{B}_n^{(h)})_{n,h}) = \ln\left(\mathbb{P}_{\mathbf{A},\mathbf{P}|X,C,\mathbf{B}}(\mathbf{a}, \mathbf{p}|(x_n^{(h)})_{n,h}, (C_n^{(h)})_{n,h}, (\mathbf{B}_n^{(h)})_{n,h})\right). \tag{29}$$

Assuming additionally that the feature and class variables $(x_n^{(h)}, C_n^{(h)})_h$ are independent knowing $(\mathbf{p}_n^{(h)}, \mathbf{a}^{(h)})_h$ and that the mixing probabilities $(\mathbf{p}_n^{(h)})_h$ are independent knowing $(\mathbf{B}_n^{(h)})_h$, the posterior in Equation (29) can be factorized in a very similar way as in the previous section. Indeed, the log-posterior writes

$$\begin{aligned}\ell(\mathbf{p}, \mathbf{a}; (x_n^{(h)}, C_n^{(h)}, \mathbf{B}_n^{(h)})_{n,h}) = &\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{k=1}^{K} \mathbb{1}_k(C_n^{(h)})\left[\ln\left(p_{n,k}^{(h)}\right) + \ln\left(\mathbb{P}_{X^{(h,k)}}(x_n^{(h)}; a_k^{(h)})\right)\right] \\ &+ \ln\left(\mathbb{P}_{\mathbf{P}|\mathbf{B}}\left(\mathbf{p}_n^{(h)}\middle|\mathbf{v}_n^{(h)}\left(\mathbb{1}_1(C_\cdot^{(1)}),\dots,\mathbb{1}_1(C_\cdot^{(H)}),\dots,\mathbb{1}_K(C_\cdot^{(1)}),\dots,\mathbb{1}_K(C_\cdot^{(H)})\right)\right)\right) \\ &+ \sum_{h=1}^{H}\mathbb{P}_{\mathbf{A}}(\mathbf{a}^{(h)}) + W((x_n^h, C_n^h, \mathbf{B}_n^h)_{n,h}),\end{aligned} \tag{30}$$

where W is the function gathering the quantities in ℓ that do not depend on $\mathbf{p}$ or $\mathbf{a}$. As in Proposition 1, we can derive linear update rules for the probability maps in the EM algorithm as detailed in the following proposition.

Algorithm 1 MAP inference on a probabilistic model combining mixture models.

Input: Data feature vectors $(x_n^{(h)})_{n,h}$, numbers of iterations n_{iter}, of components K and of layers H, linear functions $(u_{n,k}^{(h)})_{n,h}$.

Output: Mixing probability maps $\hat{\mathbf{p}}^{(h)}$ and mixture parameters $\hat{\mathbf{a}}^{(h)}$.

1. Initialize mixture parameters of layer 1 with K-means algorithm.
2. Initialize mixing probabilities of other layers with the posterior probabilities of layer 1.
3. Run M-step for all layer $h \geqslant 1$.
4. For $t \leqslant n_{iter}$,
   - E-step:
     For $h \leqslant H$, compute $\tau_{n,k}^{(t,h)}$.
   - M-step:
     For $h \leqslant H$, compute mixing probability maps $p_{n,k}^{(t+1,h)}$ using Equation (31) and model parameters $\mathbf{a}^{(t+1,h)}$.

Proposition 3. *For all $(n,k,h) \in \Omega_{N,K,H} = \{1,\dots,N\} \times \{1,\dots,K\} \times \{1,\dots,H\}$, let $u_{n,k}^{(h)} : \mathbb{R}^N \longrightarrow \mathbb{R}$ be any linear function such that $u_{n,k}^{(h)}\left([0,+\infty[^{NH}\right) \subset \mathbb{R}_+$. If*

$$\begin{aligned}\mathbf{v}_n^{(h)}(\mathbb{1}_1(C_\cdot^{(1)}),\dots,\mathbb{1}_1(C_\cdot^{(H)}),\dots,\mathbb{1}_K(C_\cdot^{(1)}),\dots,\mathbb{1}_K(C_\cdot^{(H)})) \\ = (u_{n,1}^{(h)}(\mathbb{1}_1(C_\cdot^{(1)}),\dots,\mathbb{1}_1(C_\cdot^{(H)})) - \mathbb{1}_1(C_\cdot^{(h)}) + 1,\dots, \\ u_{n,K}^{(h)}(\mathbb{1}_K(C_\cdot^{(1)}),\dots,\mathbb{1}_K(C_\cdot^{(H)})) - \mathbb{1}_K(C_\cdot^{(h)}) + 1)\end{aligned}$$

then, the mixing probability updates of layer h are

$$\forall (n,k,h) \in \Omega_{N,K,H}, \quad p_{n,k}^{(t+1,h)} = \frac{u_{n,k}^{(h)}(\tau_{\cdot,k}^{(t,1)},\dots,\tau_{\cdot,k}^{(t,H)})}{\sum_{k=1}^K u_{n,k}^{(h)}(\tau_{\cdot,k}^{(t,1)},\dots,\tau_{\cdot,k}^{(t,H)})} \tag{31}$$

where $\tau_{n,k}^{(t,h)} = \mathbb{P}_{C_n^{(h)}|X_n^{(h)},\boldsymbol{\Theta}}(k|x_n^{(h)},\boldsymbol{\theta}^{(t,h)})$ is the k^{th} component posterior probability of sample $x_n^{(h)}$ at the previous E-step and $\boldsymbol{\theta}^{(t,h)}$ is the previous parameter estimate.

The proof of this proposition is similar to the proof of Proposition 1 and can be found in supplementary section Appendix B. Moreover, it is also possible to derive two sub-factor graphs from this model, each one associated with a step of the EM algorithm with no loop, so each iteration of the EM algorithm does increase the incomplete log-posterior. The pseudo code implementing our model for combining multiple mixture models is given in Algorithm 1.

3. Numerical Experiments

3.1. Application to Synthetic Image Segmentation

3.1.1. Single Mixture

Implementation details. We first validated our algorithms on synthetic data generated from a mixture model with $K = 3$ components and Gaussian samples per component. We arranged observations on a 2D grid of size $N = 256 \times 256$ with feature dimension $D = 3$, similar to an RGB image with 256×256 pixels. To generate

synthetic data compatible with the model of Figure 1d, we started from deterministic component–assignment maps (*i.e.* $\mathbf{P}_n$ is a one-hot vector), obtained by manually partitioning the image in 3 spatially compact regions. Then, we sampled the component assignments $\mathbf{C}$ and observations X from (23).

One important test of our model is how it handles uncertainty about component assignment. Therefore, we performed two manipulations of the model parameters, to control uncertainty (Figure 4). First, we controlled prior location uncertainty by systematically smoothing the maps $\mathbf{P}$ using a Gaussian function with increasing width (see Figure 4a and Figure 5a). Second, we increased the overlap between the Gaussian distributions of the different classes, by increasing the observation variance (Figure 4b). Figure 4c illustrates one sample image for each of the 9 uncertainty combinations we explored, with location uncertainty increasing from left to right, and component overlap increasing from top to bottom.

We applied our algorithm using 2-D Gaussian smoothing functions for $\mathbf{v}_n$ such that the update rule is

$$p_{n,k}^{(t+1)} = G * \tau_k^{(t)}(l_n) \tag{32}$$

where G is a Gaussian kernel with width $\sigma = 5.25$, $\tau_k^{(t)} : l_n \mapsto \tau_{n,k}^{(t)}$ is the posterior maps and $*$ denotes the discrete convolution. For comparison, we also used a standard Gaussian Mixture Model with three components.

Results. We analyzed the component probability maps learned by our model (Figure 4d, labeled "Ours"; each map corresponds to one component, and lighter gray scale values correspond to higher probability of that component), and compared them to the ground truth maps ("GT") as well as to a standard Gaussian Mixture model ("GMM"). When uncertainty is minimal, our algorithm recovers the GT maps accurately, and it improves slightly over the GMM, by removing isolated outliers thanks to the spatial smoothing (Figure 4d, top–left block). Increasing uncertainty either by location prior or component overlap, leads to more dramatic failures of the GMM, whereas our model captures both the shape of the GT maps, as well as the increased uncertainty (abundance of intermediate gray levels, corresponding to probabilities close to 0.5, in the top row and left column of Figure 4d). Importantly, when prior location uncertainty is low, our model is more robust to component overlap uncertainty than the GMM (left column and bottom blocks), whereas it displays similar failures as the GMM (except for the spatial smoothing) when both component overlap and prior location uncertainties become too large (Figure 4d, bottom–right block).

In summary, together our simulations demonstrate that our algorithm learns probabilistic component-assignment maps, correctly capturing ground–truth labels and their uncertainty, and performing spatial smoothing. We note that the spatial smoothing achieved by our algorithm is qualitatively similar to that obtained by other approaches [43, 54]. However, our formulation also allows for straightforward and efficient integration of information consistent with different data topologies, well beyond spatial smoothing, which we illustrate next.

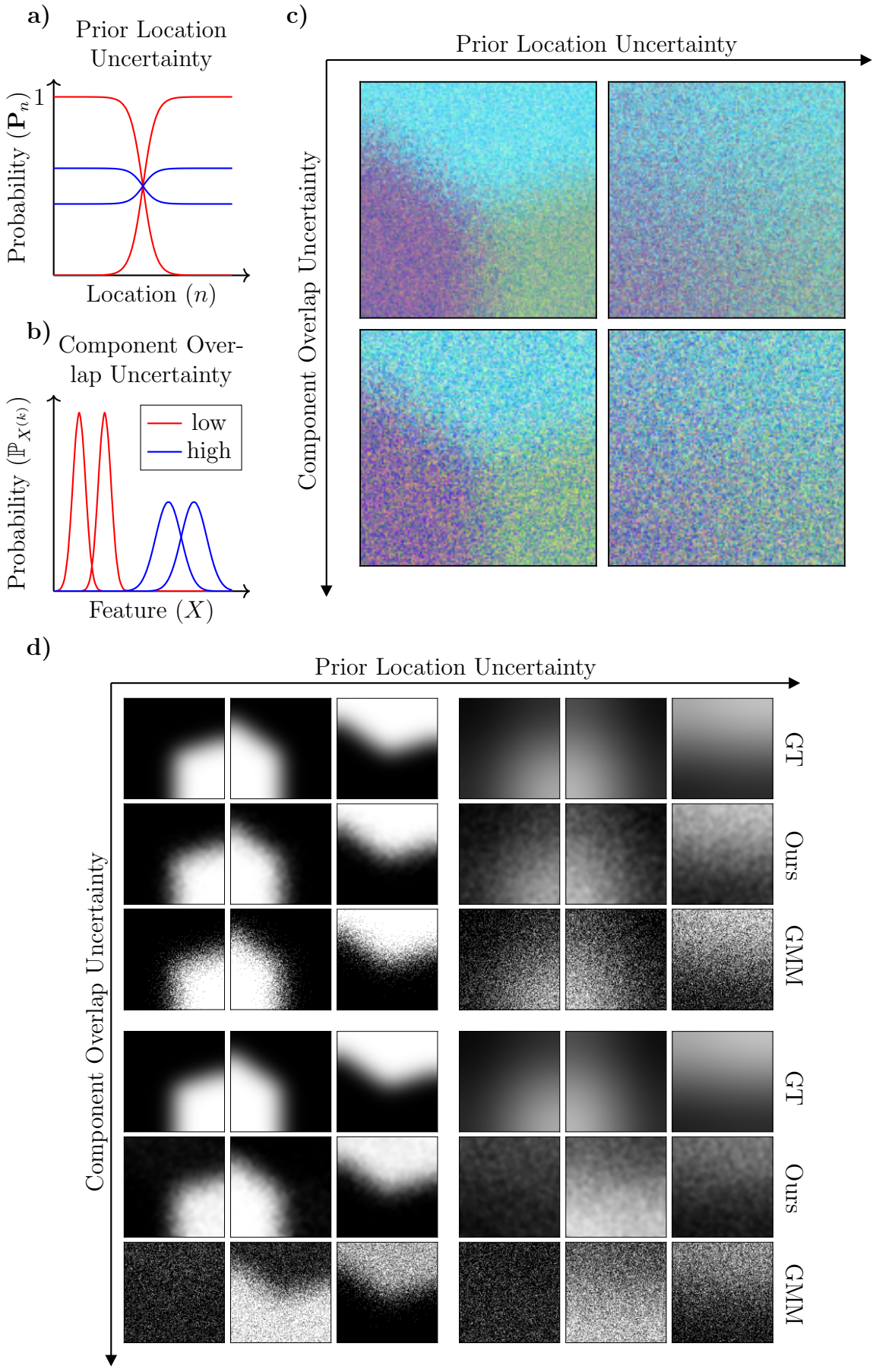


Figure 4: See caption on the next page.

Figure 4: Validation on synthetic color images with uncertainty manipulation. **a)** We control location uncertainty by setting the probability of each component to values close to 1 (red, low uncertainty) or to 0.5 (blue, high uncertainty), at any given location in the image (illustrated for a 1-dimensional image in the cartoon). **b)** We control component uncertainty by decreasing (red, low uncertainty) or increasing (blue, high uncertainty) the overlap of the Gaussian distributions of the two features within each component (illustrated for a 1-dimensional feature). **c)** Example synthetic images with three feature channels (corresponding to RGB values) and three components, generated from the model with increasing location uncertainty from left to right, and and increasing component overlap from top to bottom. **d)** Each block of 3 by 3 maps contains the ground truth component probability maps (top row, labeled GT), and the maps recovered by our algorithm (middle row, labeled Ours) and by a Gaussian Mixture Model (bottom row, labeled GMM). Each map corresponds to a different component, and lighter pixels correspond to higher probability that the pixel is assigned to that component. Different 3 by 3 blocks correspond to different levels of location uncertainty and component overlap uncertainty, as described in **a)** and **b)**. Figure is best seen in color.

3.1.2. Hierarchical Mixtures

Implementation details. To illustrate the flexibility of our approach, we applied it to data generated from a 3–layer model as in Figure 3. This can be interpreted as a simple hierarchical model of images, with features corresponding to different layers of a DNN. For these simulations, we used three feature channels at each layer, and displayed the data as RGB images (Figure 5a right). The model at each layer is as in the previous section, with identical prior maps at all layers, except that the location uncertainty is largest in the first layer and smallest in the third layer (illustrated by the Ground Truth Maps in Figure 5a center). To perform the prior maps inference, we applied the model of Figure 3 (*i.e.* where component–assignments at one layer influence the prior of the other neighboring layers). In other words, we applied our algorithm using a combination of Gaussian smoothing functions for $\mathbf{v}_n$ leading to the following update rule

$$p_{n,k}^{(h,t+1)} = \frac{s_n^{(h,t)^2} s_n^{(h+1,t)^2} m_{n,k}^{(h-1,t)} + s_n^{(h-1,t)^2} s_n^{(h+1,t)^2} m_{n,k}^{(h,t)} + s_n^{(h-1,t)^2} s_n^{(h,t)^2} m_{n,k}^{(h+1,t)}}{s_n^{(h,t)^2} s_n^{(h+1,t)^2} + s_n^{(h-1,t)^2} s_n^{(h+1,t)^2} + s_n^{(h-1,t)^2} s_n^{(h,t)^2}}, \tag{33}$$

where

$$m_{n,k}^{(h,t)} = G^{(h)} * \tau_k^{(h,t)}(l_n), \tag{34}$$

$$s_n^{(h,t)^2} = \frac{\sum_{k=1}^{K} G^{(h)} * \tau_k^{(h,t)^2}(l_n) - m_{n,k}^{(h,t)^2}}{K(1 - G * G(0))}, \tag{35}$$

are respectively the local mean and variance of the posterior maps at layer h; $\tau_k^{(h,t)} : l_n \mapsto \tau_{n,k}^{(h,t)}$ is the posterior map at layer h; and $G^{(h)}$ is a 2-D Gaussian kernel with width $\sigma^{(h)} = 5.25$.

Results. Figure 5b illustrates that our algorithm learned maps consistent with the ground truth at all layers, and also qualitatively reflected the decreasing uncertainty across layers. Importantly, because the integration across layers of assignment information is weighted by the relative uncertainties, the maps in the first layer are less uncertain than the corresponding ground truth (average difference between the entropy of the Fit and Ground Truth maps = -0.14), and, correspondingly, the maps in the second and third layer are more uncertain than the ground truth (entropy difference = 0.13 for layer 2, and 0.23 for layer 3).

As a control, we also applied our single–mixture model independently to each layer (Figure 5c). As expected, the algorithm recovered the maps correctly in the

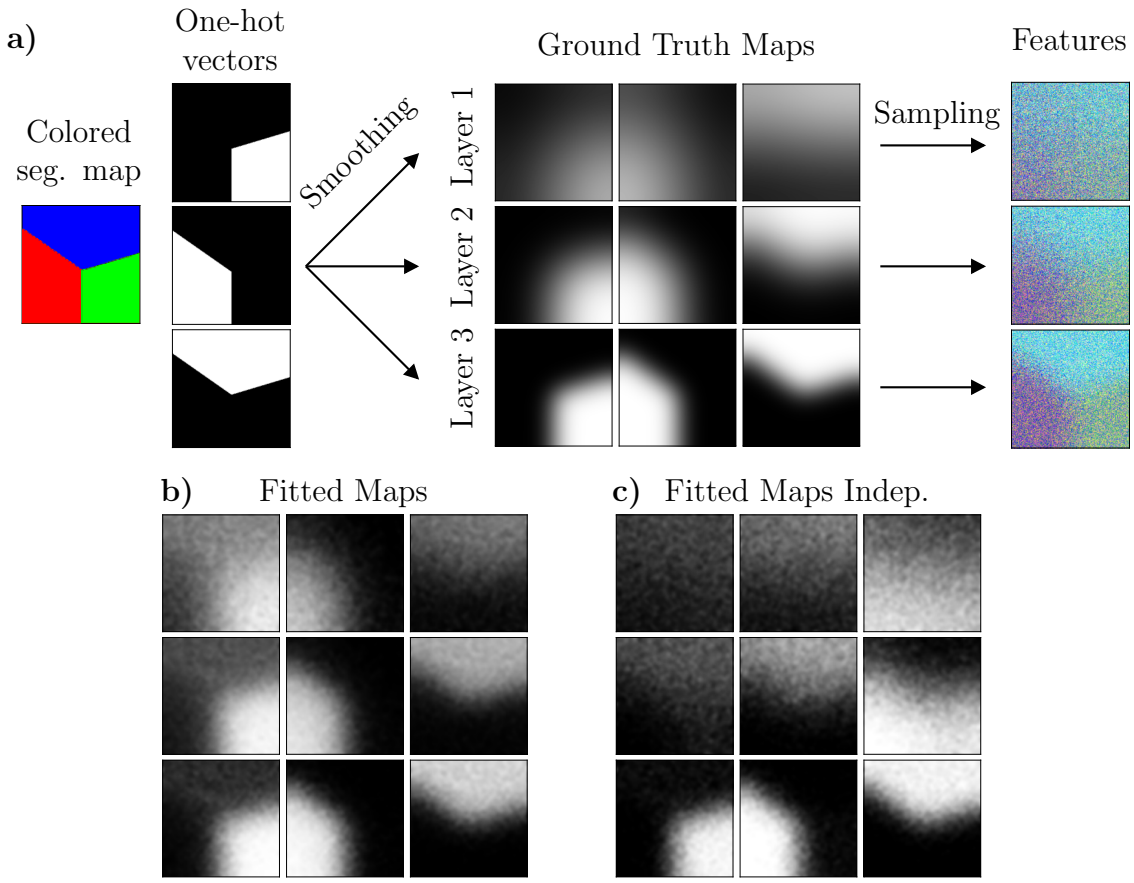


Figure 5: Validation of hierarchical mixtures. **a)** Pipeline to generate the synthetic hierarchical data. Example synthetic images have three feature channels (displayed as RGB images on the right) and three components (columns of Ground Truth Maps), at three different layers (rows of Ground Truth Maps). **b)** Maps recovered by our hierarchical multi-layer model (Figure 3). Same conventions as in **a)**. **c)** Maps recovered by applying our single mixture model independently at each layer. Same conventions as in **a)**. Figure is best seen in color.

less uncertain layer, but was not able to recover them in more the uncertain layers (due to the high component uncertainty regime).

In summary, these results on synthetic data illustrate that our approach can integrate information about mixture components from multiple sources in a principled way, and easily adapt to different topologies of the data (in our examples, spatial arrangement and hierarchical organization).

3.2. Application to Natural Image Segmentation

To test our algorithms on natural images, we considered unsupervised segmentation of natural images. This is a classical application of mixture model, and particularly relevant for our framework, given that natural images have both spatial and hierarchical structure, as demonstrated by the success of DNN models on many visual tasks. We therefore applied our algorithms to natural images from the BSD500 dataset [1], and compared the results between different variants of our model and to the segmentation maps of human observers.

3.2.1. Single Mixture

Implementation details. As in the synthetic image experiments, we used RGB colors as features. In addition to GMM, we ran our algorithm using Student-t Mixture Models (SMM), that is using Student-t distributions to describe the data distribution of each mixture component. The SMM is more robust to outliers, and the comparison with the GMM allowed us to assess the importance of such robustness for segmentation performance [9, 10, 58]. For both GMM and SMM, we also compared the results with and without using our smoothing priors, to study their relevance for segmentation. As the number K of components is unknown, we also ran our algorithm with $K = 3$, 6 and 9, and studied the results separately. As for synthetic data, the functions $\mathbf{v}_n$ are 2-D Gaussians, with width $\sigma = 2.75$. To obtain segmentation maps directly comparable with those measured in humans, we used a maximum a posteriori (MAP) approach: we assign each pixel to the component with the highest probability, and therefore discard the probabilistic information.

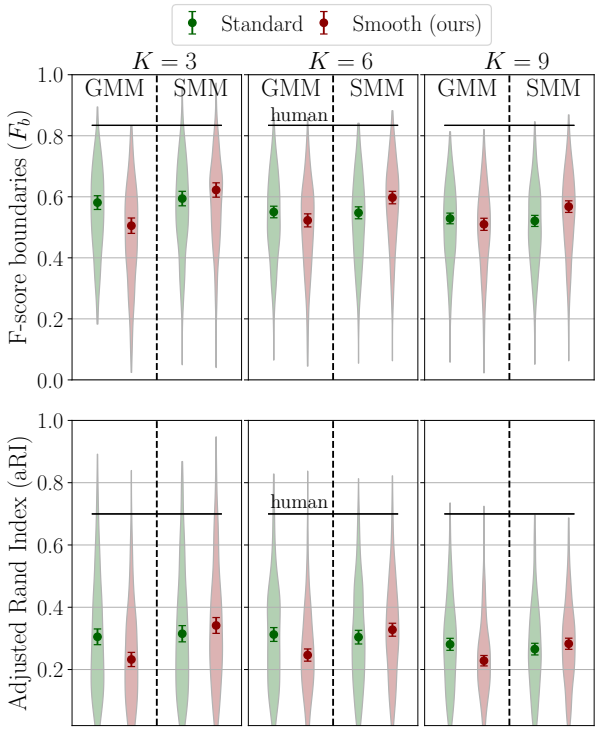


Figure 6: Performance of single mixture models trained with RGB color features on the BSD 500 for the two scores defined in text. Error bars indicate 3 times the standard error of the mean. Shaded areas represent scores density. Figure is best seen in color.

Quantitative Results. To quantify the performance of our algorithms, we used two widely-adopted scores, that measure the similarity between the segmentation maps of humans and of the algorithms: the adjusted Rand Index (aRI) [23] and the F-score for boundaries (F_b) [46]. The aRI

measures the overlap between regions in the two segmentation maps, whereas F_b measures the consistency of boundaries between regions. The results are reported in Figure 6. Overall performances of mixture models are much lower than those of humans (which is assessed by quantifying consistency across different observers), and generally the best performances are achieved using the SMM and smoothing. Interestingly, for all tested K, our smoothing improves both scores for SMM while it impairs both scores for GMM. The reason is that the smoothing dramatically affects the parameters learned in Gaussian components while the robustness of the Student's components prevents this effect. Furthermore, increasing the number of components decreases the average scores in general, although the best K varies substantially across images. The average decrease in performance at larger K might be due to an increase in the detection of false contours.

Qualitative Results. In Figure 7, we show four segmentation examples among which two have high scores (top) and two have low scores (bottom). The main qualitative difference is that images with high score have sharp contours delimiting weakly textured areas, and different areas have largely different color, whereas low score images have blurry contours delimiting rough textured areas. Similar to what we have shown in the synthetic data, our smoothing enforces contiguity of component areas. The examples of Figure 7 also illustrate some aspects of our quantitative results. For both images on top, when we train mixture models with $K = 9$ components, we observe false contours and therefore the emergence of superfluous areas, which impairs both scores. For the top-right image, we observe that the bear is well-delimited by the GMM, whereas the smoothing gives a coarser segmentation that changes the learned mean of the bear component (from gray to lighter gray). This is not the case for SMM with smoothing, thanks to the robustness of the Student-t distribution.

3.2.2. Multiple Mixtures

Implementation details. For the hierarchical observations $(x_n^{(h)})_{n,h}$, we used image features extracted by the pre-trained deep network VGG 19 [53]. We compared Gaussian (GMM) and Student-t (SMM) mixture components. The Student-t distribution captures the sparse, heavy-tailed behavior of both low-level features (*e.g.* wavelet coefficients [58, 55]) and higher-level features extracted by DNN [49]. Therefore, comparing GMM to SMM allows us to study the importance of modeling accurately the component distributions, for segmentation performance.

To assess the relative importance of integrating information across spatial and hierarchical dimensions, we implemented two models. Model **a** used a nearest-neighbor hierarchical structure, similar to Figure 3. Therefore, in this model, each layer integrated information spatially as well as from one layer below and one above. Because model **a** resulted in a separate (though not independent) segmentation map per feature layer, which need to be combined afterward, we also considered a second model (model **b**), in which the observations at all layers shared a single segmentation map, *i.e.* a single set of class labels $(C_n)_n$. Therefore, in model **b**, the prior parameters and corresponding mixing probabilities were influenced by the observations at all layers, with uncertainty-weighting. Again, by choosing the function $\mathbf{v}_n^{(h)}$ appropriately, model **a** has the update rule given by Equation (33)

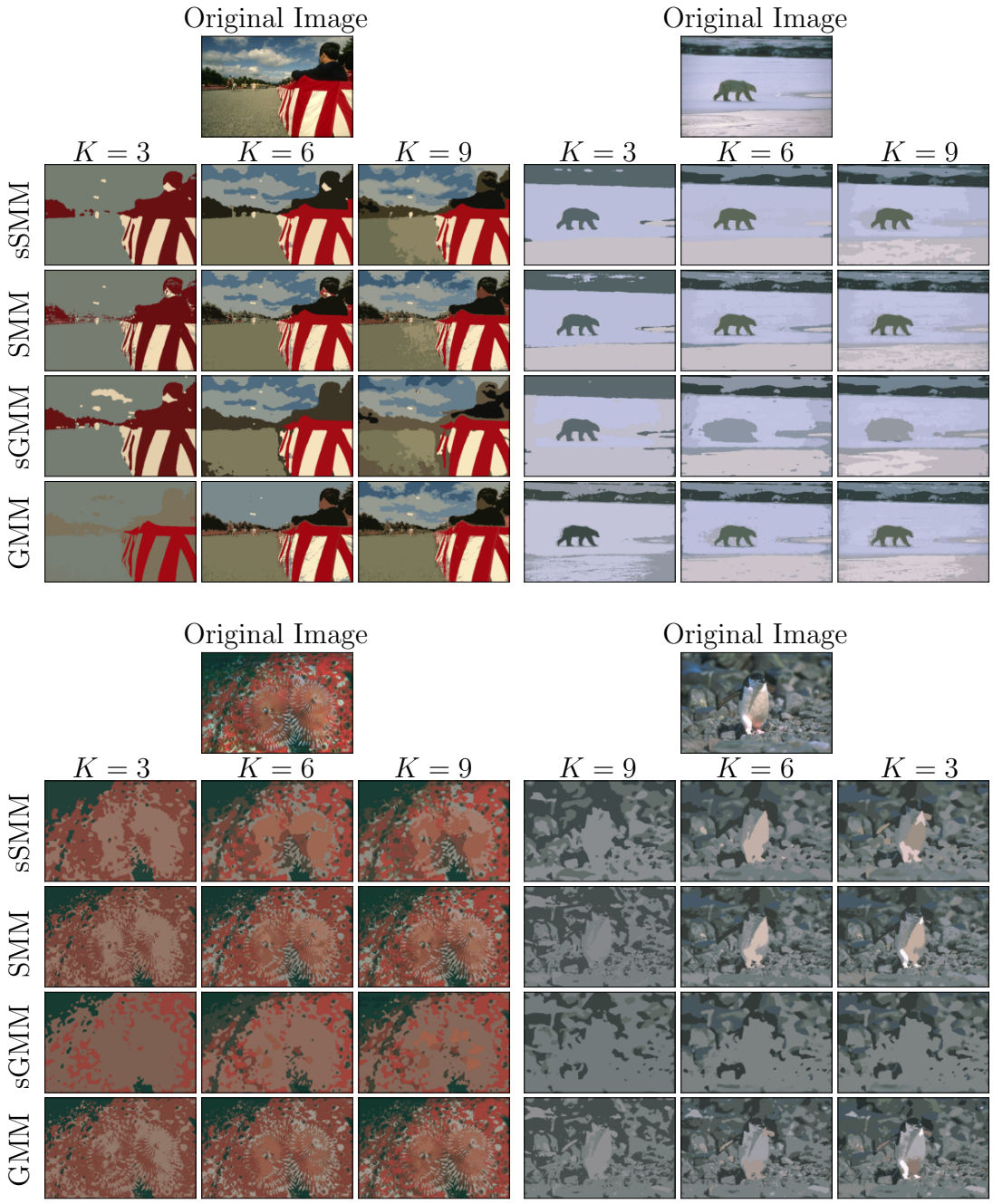


Figure 7: Segmentation of natural images using mixture models with (sGMM/sSMM) and without (GMM/SMM) smoothing. Top: high-scoring images. Bottom: low-scoring images. We display the area corresponding to each component with the average color of the original image inside that area. Figure is best seen in color.

whereas model **b** has the following update rule

$$p_{n,k}^{(1,t+1)} = \frac{\sum_{h=1}^{H} \prod_{i \neq h} {s_n^{(i,t)}}^2 m_{n,k}^{(h,t)}}{\sum_{h=1}^{H} \prod_{i \neq h} {s_n^{(i,t)}}^2}, \tag{36}$$

where $m_{n,k}^{(h,t)}$ and ${s_n^{(h,t)}}^2$ are respectively the local mean and variance of the posterior maps at layer h, defined in Equation (34) and (35) respectively; $\tau_k^{(h,t)} : l_n \mapsto \tau_{n,k}^{(h,t)}$ is the posterior map at layer h; and $G^{(h)}$ is a 2-D Gaussian kernel with width $\sigma^{(h)}$. The two models **a** and **b** are fitted using Algorithm 1, with respectively the modified update rule (33) and (36). We used the following values of $\sigma^{(h)}$: 4.25, 4.25, 3.25, 3.25, 2.25, 2.25, 2.25, 2.25, 0.75, ..., 0.75 for both models. Importantly, the modified update rules implement uncertainty-based integration: the local mean of each layer is weighted by the local variances of the other layers, such that a layer with higher uncertainty (large variance) will contribute relatively less to the update of the other layers.

We adapted our implementation of the models to the structure of VGG features. First, the number of pixels (*i.e.* the number of samples) is not same in the different layers, therefore we up-sampled the posterior probability maps $\tau_k^{(h,t)}$ using nearest neighbor interpolation before convolution with $G^{(h)}$ when it was necessary. Second, the decreasing number of samples and the increasing dimension of features along the depth of the network often caused numerical issues, therefore we reduced the dimension at each layer using Principal Component Analysis (PCA), including only the dimensions needed to capture 95% of the variance. Third, the first layer of the deep network is a linear transform of the input image, in contrast with all subsequent layers, therefore we added the features of the first layer to all subsequent layers (using average pooling when necessary). As mentioned above, model **a** provides one map per layer. To allow for comparison with model **b**, we used two approaches: (i) we computed the product of all the probability maps at all layers, and then computed the MAP segmentation map; (ii) we computed the MAP segmentation map from the probabilities of the first layer only.

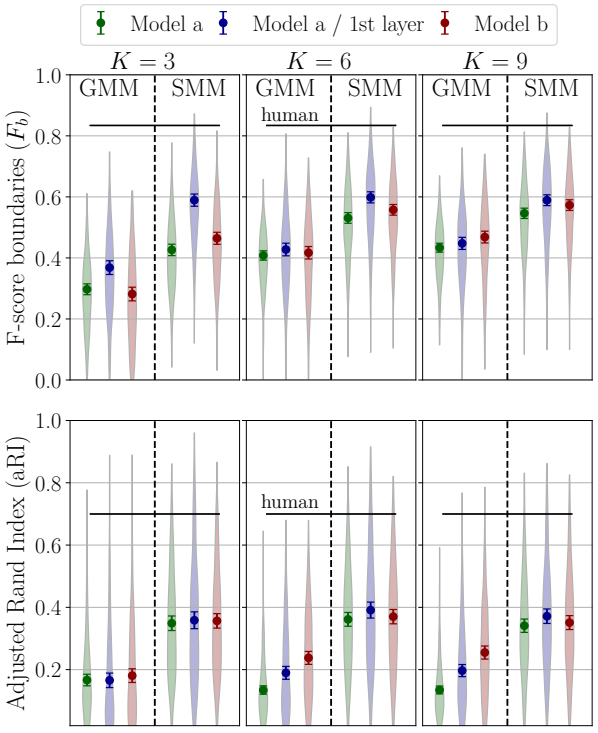


Figure 8: Performance of model **a** and **b** trained with VGG 19 features on the BSD 500 for the two scores defined in text. Error bars and shaded areas: same convention as in Figure 7. Figure is best seen in color.

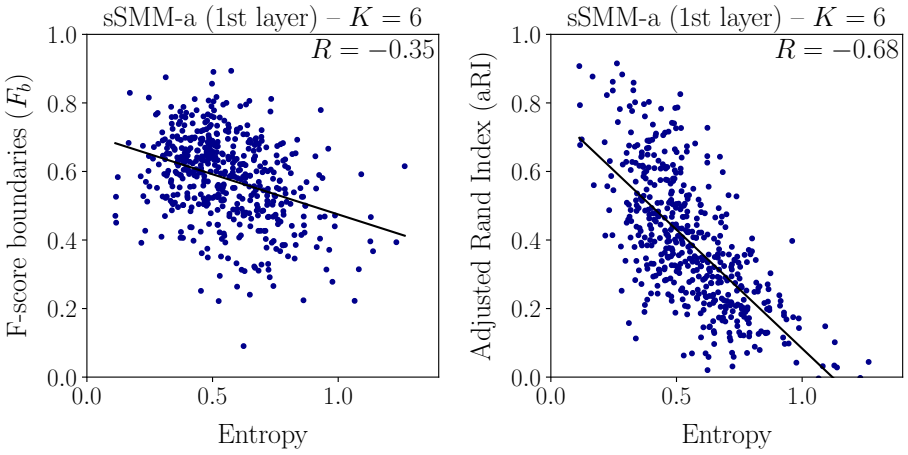


Figure 9: Negative correlation between both scores and entropy of model **a**/1st layer when using Student-t mixtures.

Quantitative Results. The F-score and aRI of model **a**, model **a**/1st layer only, and model **b** are reported in Figure 8. The use of VGG features impairs contour detection in comparison to the use of RGB features, for the GMMs and the SMM without smoothing. Indeed the F-score is much lower for all model variants except for model **a**/1st layer and model **b** both with SMM (Figure 8, top *vs* Figure 6, top). Increasing the number of component K tends to recover the performance on the F-scores, for the models that are impaired by VGG features. The impairment of contour detection is due to the reduced resolution of the maps at deeper layer, which influences the combined segmentation map, but much less the map obtained with the 1st layer only. In contrast, the use of VGG features improves the aRI of SMM while impairing the aRI of GMM, in comparison with RGB features (Figure 8, bottom *vs* Figure 6, bottom). Yet, the use of combined maps does not improve the aRI score of the 1st layer segmentation maps. Increasing the number of components K has almost no effect on average performances, although, as for RGB, the best K varies substantially across images. Overall performances of model **b** are higher than those of model **a** but are lower than those of the model **a**/1st layer. Lastly, as for RGB, the performance of both models is much lower than humans. We show in Figure 9 that both scores are negatively correlated with the level of uncertainty captured by the Student-t mixture model. Such correlation is interesting because it shows that resolving high uncertainty areas using additional low uncertainty features can improve segmentation quality. In addition, this could explain the marginal superiority of model **b** over the combination of probabilities of model **a**. Indeed model **b** weights low uncertainty layer maps more which should lead to higher scores. This observation does not hold for Gaussian mixture (see supplementary Figure A.12).

Qualitative Results. We show in Figure 10 four segmentation examples, two with high scores (top) and two with low scores (bottom) for model **a** (1^{st} layer and combined) and model **b**. Similar to the segmentation based on RGB features, images with high scores tend to have sharper contours between segments and more uniform textures within each segment. Comparing the top-left image with the RGB case, it is evident that both hierarchical models lead to better reduction of noise (*e.g.* within the areas of snow and the top part of the image for $K = 9$) at the expense of lower spatial resolution, both of which are due to incorporating information from deeper

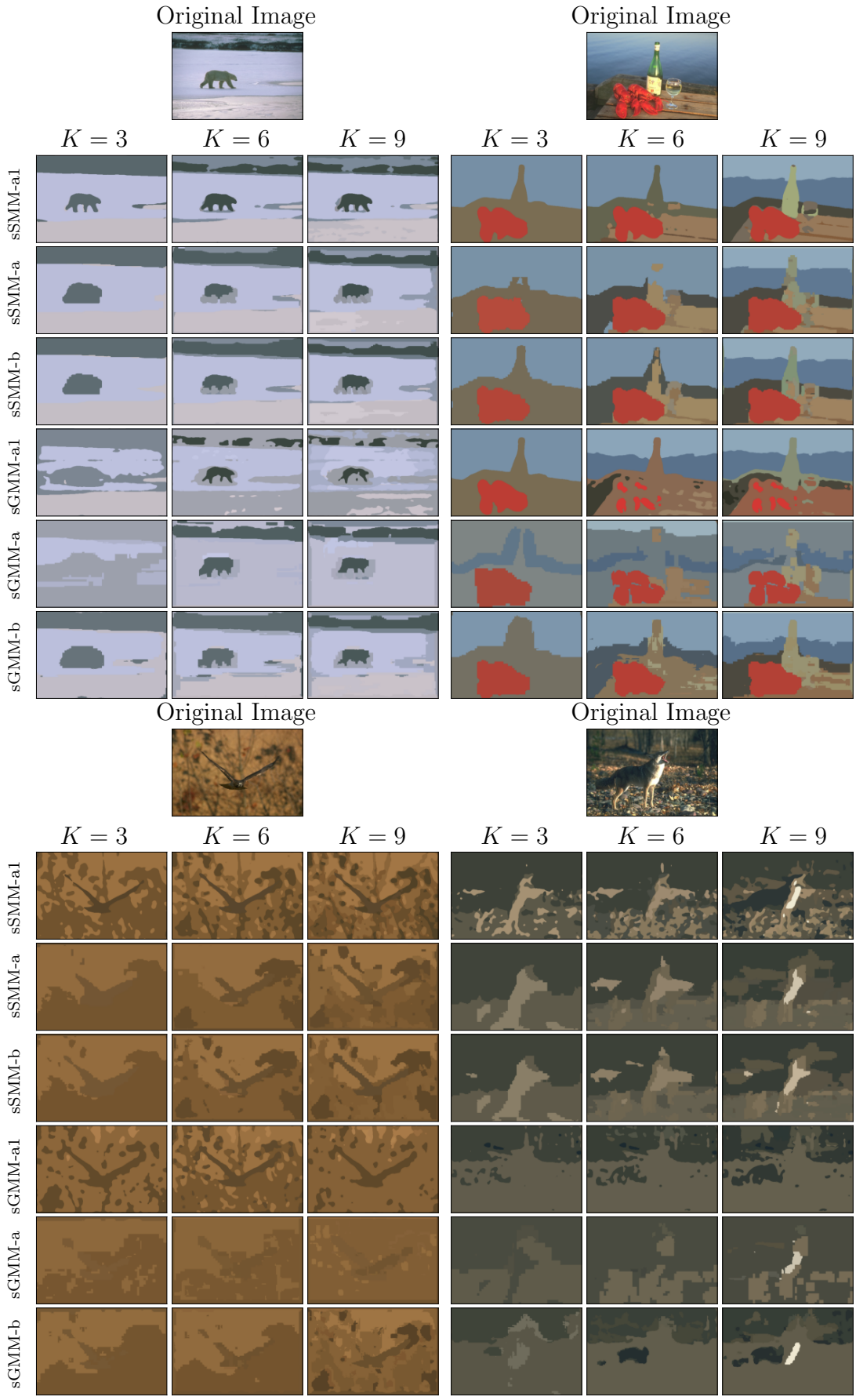


Figure 10: Segmentation of natural images using models **a** (1st layer and combined) and **b** on VGG 19 features with smoothing for Gaussian (sGMM) and Student-t mixture (sSMM). Top: high-scoring images. Bottom: low-scoring images. Figure is best seen in color.

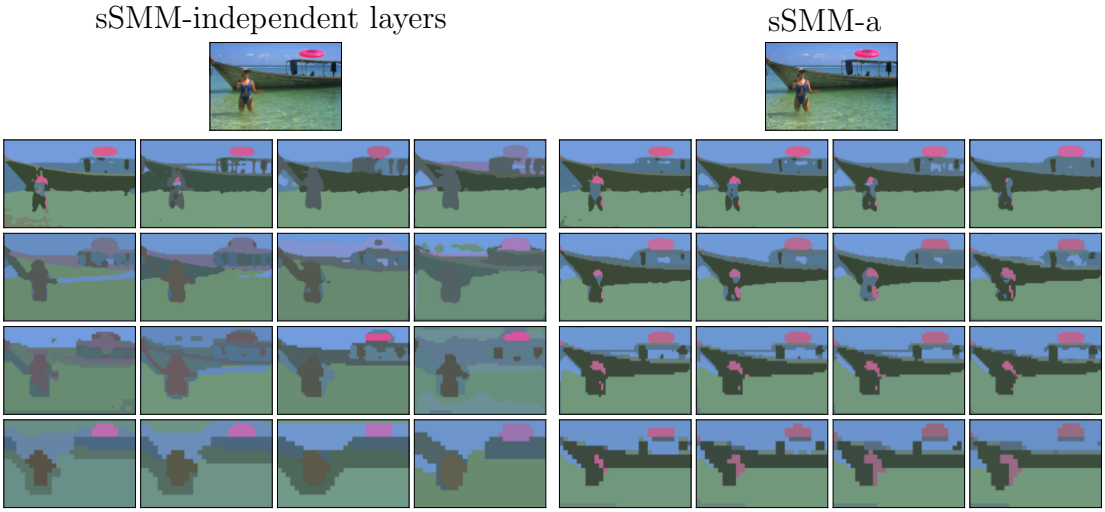


Figure 11: Segmentation at all layers for $K = 6$. Left: sSMM trained independently on each VGG layer. Right: sSMM-a. Figure is best seen in color.

layers (compare both models to model **a** 1st layer). However, the segmentation maps provided by models **a** and **b** are hard to distinguish, indicating that both combination methods (uncertainty weighting *vs* product) lead to similar results for this particular image. Nevertheless the negative correlation between scores and entropy suggests that model **b** (*i.e.* uncertainty weighting) is best across multiple images. Finally, we illustrate in Figure 11 how our local combination of layers (model **a**) enforces consistency of segmentation maps across layers in comparison to when mixture models are trained on each layer independently.

3.3. Discussion

We have presented a new approach to unsupervised clustering via probabilistic mixture models, that exploits the topology of the dataset to improve clustering performance. We have shown that, similar to other related algorithms, our method can be understood by considering the probability that each datapoint belongs to a latent class (the mixing probability) as a free parameter, and by introducing topological information as a strong regularization. Our main innovation consists in how we formulate the joint prior over mixing probabilities of all datapoints (*i.e.* the regularizer): With our formulation, when optimizing the parameters via the EM algorithm, the update rule of the mixing probability of one datapoint is linear in the mixing probabilities of the other datapoints (Table 1). Furthermore, our method allows us to choose any desired linear combination of the mixing probabilities, and therefore can be adapted to arbitrary data topology.

We have demonstrated how the approach can be used to obtain spatial smoothing of clusters when the data have two spatial dimensions (*e.g.* images; Figure 1d), and uncertainty-weighted integration across multiple feature channels organized hierarchically (*e.g.* across the layers of a DNN; Figure 3). We demonstrated on synthetic data that our method performs spatial smoothing and captures ground-truth clustering uncertainty accurately (Figures 4 and 5). In comparison to related methods [43, 54], ours has the advantages of computational simplicity and flexibility, because it allows the user to choose any desired linear update rule, to combine information according to the data topology.

Our application to unsupervised segmentation of natural images, based on the pixels RGB values, demonstrates that smoothing is effective at removing spurious spatial discontinuities in segmentation maps, but also that, when doing so, it is important to adopt an observation model that is robust to outliers (*e.g.* mixtures of Student-t rather than Gaussian distributions; Figures 6 and 7). Our application to natural image segmentation based on VGG-19 demonstrates the flexibility of our approach to combine segmentation information across hierarchical layers. Because of the lower spatial resolution of deep layers, combining segmentation information across layers increases the resolution of the segmentation maps in deeper layers, while reducing it in superficial layers. In addition, our method produces segmentation maps that are consistent across all layers (Figures 10) and exploits higher-level information about objects (encoded implicitly by the deep layers of VGG-19) to correctly group areas that could otherwise be segmented based on low-level feature appearance, and to further improve noise reduction compared to spatial smoothing alone (top-left of Figure 10 *vs* top-right of Figure 7 for $K = 9$).

Our analysis of quantitative performance on the BSD500 (Figures 6 and 8) confirms the importance of using mixture components robust to outliers: the Student-t mixtures generally outperformed the Gaussian mixtures, particularly when using VGG19 features. The comparisons also highlight that information can be combined across the VGG19 hierarchy in different ways, leading to different outcomes. For the F_b score (related to contour detection), a nearest-neighbor strategy performs better than a single prior map shared between all layers, because the former preserves higher spatial resolution; whereas for aRI (related to region overlap) there is not much difference. Overall, the performances of our models are far from human to human consistency and from state of the art unsupervised algorithms [36]. This could be improved by fine tuning certain implementation details, such as the dimensionality reduction of the feature space, which can be improved with subspace clustering methods [4]. In addition, as usual when using mixture models, the number of components K can substantially increase the score obtained for a single image, therefore overall performance can be improved by methods to choose the best K per image. Another avenue for improvement is suggested by the fact that image segmentation in human perception is influenced strongly by semantic information about the objects that are recognized in the image [41]. Therefore, future extensions of our model for segmentation should include explicit object information, either directly, *e.g.* using object classes predicted by VGG to bias the prior on mixing probabilities, or indirectly using human segmentations to refine the observation model of each cluster via semi-supervised learning.

The computational advantages of our approach come at the cost of introducing loops in the probabilistic graphical model, specifically between the class information and the prior on the class (Figure 1d). In this paper, we have shown that this issue is mitigated by considering factor graphs theory and by observing that the loops are absent from the two subgraphs corresponding to the two steps of the EM algorithm. Recurrent network models, that have been proposed recently to account for the fact that human image segmentation likely involves heavy recurrence and hierarchical feedback [32], also rely on loopy architectures and are typically trained unrolling the model in time. Those models have been tested on specific tasks and benchmarks designed to highlight the importance of recurrence, but not on natural image segmentation databases. Although our approach is based on probabilistic graphi-

cal models and therefore is not directly comparable to recurrence and feedback in deterministic networks, our method to combine class information across datapoints effectively achieves similar goals. Additionally, our approach being fully probabilistic, it guarantees that the recurrent and feedback information are correctly weighted by their uncertainty, and the resulting segmentation maps are probabilistic. In future work, this could be exploited to study quantitatively recurrent processing in human perception and in neural processing in the visual cortex [56, 31].

Acknowledgements

We thanks Pascal Mamassian for fruitful discussion. RCC is supported by NIH (NIH/CRCNS EY031166). JV is supported by ANR (ANR-19-NEUC-0003-01).

References

[1] Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5), 898–916.

[2] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12), 2481–2495.

[3] Blei, D. M. & Frazier, P. I. (2011). Distance dependent chinese restaurant processes. *Journal of Machine Learning Research*, 12(8).

[4] Bouveyron, C. & Brunet-Saumard, C. (2014). Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71, 52–78.

[5] Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[6] Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11), 1222–1239.

[7] Boyles, R. A. (1983). On the convergence of the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 45(1), 47–50.

[8] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 834–848.

[9] Coen-cagli, R., Dayan, P., & Schwartz, O. (2009). Statistical models of linear and nonlinear contextual interactions in early visual processing. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems*, volume 22: Curran Associates, Inc.

[10] Coen-cagli, R., Dayan, P., & Schwartz, O. (2012). Cortical surround interactions and perceptual salience via natural scene statistics. *PLOS Computational Biology*, 8(3), 1–18.

[11] Dauwels, J., Eckford, A., Korl, S., & Loeliger, H.-A. (2009). Expectation maximization as message passing-part i: Principles and gaussian messages. *arXiv preprint arXiv:0910.2832*.

[12] Dauwels, J., Korl, S., & Loeliger, H. . (2005). Expectation maximization as message passing. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* (pp. 583–586).

[13] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1–22.

[14] Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54, 764–771.

[15] Eckford, A. & Pasupathy, S. (2000). Iterative multiuser detection with graphical modeling. In *2000 IEEE International Conference on Personal Wireless Communications. Conference Proceedings (Cat. No. 00TH8488)* (pp. 454–458).: IEEE.

[16] Eckford, A. W. (2004). Channel estimation in block fading channels using the factor graph em algorithm. In *22nd Biennial Symposium on Communications* (pp. 1–3).

[17] Elder, J. H. & Goldberg, R. M. (2002). Ecological statistics of gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2(4), 5–5.

[18] Fowlkes, C. C., Martin, D. R., & Malik, J. (2007). Local figure–ground cues are valid for natural images. *Journal of Vision*, 7(8), 2–2.

[19] Frey, B. J. (2003). Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03 (pp. 257–264). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[20] Gan, H., Sang, N., & Huang, R. (2015). Manifold regularized semi-supervised gaussian mixture model. *JOSA A*, 32(4), 566–575.

[21] Ghosh, S., Ungureanu, A. B., Sudderth, E. B., & Blei, D. M. (2011). Spatial distance dependent chinese restaurant processes for image segmentation. In *Advances in Neural Information Processing Systems* (pp. 1476–1484).

[22] He, X., Cai, D., Shao, Y., Bao, H., & Han, J. (2010). Laplacian regularized gaussian mixture model for data clustering. *IEEE Transactions on Knowledge and Data Engineering*, 23(9), 1406–1418.

[23] Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1), 193–218.

[24] Hyvärinen, A., Hurri, J., & Hoyer, P. O. (2009). *Natural image statistics: a probabilistic approach to early computational vision.* Springer.

[25] Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., & Song, M. (2019). Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11), 3365–3385.

[26] Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4), 321–331.

[27] Kim, J., Linsley, D., Thakkar, K., & Serre, T. (2019). Disentangling neural mechanisms for perceptual grouping. *arXiv preprint arXiv:1906.01558*.

[28] Koller, D. & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[29] Kreiman, G. & Serre, T. (2020). Beyond the feedforward sweep: feedback computations in the visual cortex. *Annals of the New York Academy of Sciences*, 1464(1), 222–241.

[30] Kschischang, F., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 498–519.

[31] Lee, T. S. & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7), 1434–1448.

[32] Linsley, D., Kim, J., Ashok, A., & Serre, T. (2019). Recurrent neural circuits for contour detection. In *International Conference on Learning Representations*.

[33] Linsley, D., Kim, J., Veerabadran, V., Windolf, C., & Serre, T. (2018). Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems* (pp. 152–164).

[34] Liu, J., Cai, D., & He, X. (2010). Gaussian mixture model with local consistency. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[35] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

[36] Maninis, K.-K., Pont-Tuset, J., Arbeláez, P., & Van Gool, L. (2018). Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 819–833.

[37] Martin, D. R., Fowlkes, C. C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5), 530–549.

[38] McLachlan, G. J., Lee, S. X., & Rathnayake, S. I. (2019). Finite mixture models. *Annual review of statistics and its application*, 6, 355–378.

[39] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[40] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

[41] Neri, P. (2017). Object segmentation controls image reconstruction from natural scenes. *PLoS biology*, 15(8), e1002611.

[42] Nikou, C., Galatsanos, N. P., & Likas, A. C. (2007). A class-adaptive spatially variant mixture model for image segmentation. *IEEE Transactions on Image Processing*, 16(4), 1121–1130.

[43] Nikou, C., Likas, A. C., & Galatsanos, N. P. (2010). A bayesian framework for image segmentation with spatially varying mixtures. *IEEE Transactions on Image Processing*, 19(9), 2278–2289.

[44] Olshausen, B. A. & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 607.

[45] Peterson, M. A. & Gibson, B. S. (1994). Object recognition contributions to figure-ground organization: Operations on outlines and subjective contours. *Perception & Psychophysics*, 56(5), 551–564.

[46] Pont-Tuset, J. & Marques, F. (2013). Measures and meta-measures for the supervised evaluation of image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2131–2138).

[47] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241).: Springer.

[48] Saarela, T. P. & Landy, M. S. (2012). Combination of texture and color cues in visual segmentation. *Vision research*, 58, 59–67.

[49] Sanchez-Giraldo, L. G., Laskar, M. N. U., & Schwartz, O. (2019). Normalization and pooling in hierarchical models of natural images. *Current opinion in neurobiology*, 55, 65–72.

[50] Sfikas, G., Nikou, C., & Galatsanos, N. (2007). Robust image segmentation with mixtures of student's t-distributions. In *2007 IEEE International Conference on Image Processing*, volume 1 (pp. I–273).: IEEE.

[51] Sfikas, G., Nikou, C., & Galatsanos, N. (2008). Edge preserving spatially varying mixtures for image segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–7).: IEEE.

[52] Sigman, M., Cecchi, G. A., Gilbert, C. D., & Magnasco, M. O. (2001). On a common circle: natural scenes and gestalt rules. *Proceedings of the National Academy of Sciences*, 98(4), 1935–1940.

[53] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[54] Sun, S., Paisley, J., & Liu, Q. (2017). Location dependent dirichlet processes. In *International Conference on Intelligent Science and Big Data Engineering* (pp. 64–76).: Springer.

[55] Vacher, J., Davila, A., Kohn, A., & Coen-Cagli, R. (2020). Texture interpolation for probing visual perception. *Advances in Neural Information Processing Systems*, 33.

[56] Vacher, J., Mamassian, P., & Coen-Cagli, R. (2018). Probabilistic model of visual segmentation.

[57] Wagemans, J., Elder, J. H., Kubovy, M., Palmer, S. E., Peterson, M. A., Singh, M., & von der Heydt, R. (2012). A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6), 1172.

[58] Wainwright, M. J. & Simoncelli, E. P. (2000). Scale mixtures of gaussians and the statistics of natural images. In *Advances in neural information processing systems* (pp. 855–861).

[59] Wu, C. F. J. (1983). On the Convergence Properties of the EM Algorithm. *The Annals of Statistics*, 11(1), 95 – 103.

[60] Ye, X., Zhao, J., & Chen, Y. (2018). A nonparametric model for multi-manifold clustering with mixture of gaussians and graph consistency. *Entropy*, 20(11), 830.

Appendix A. Supplementary Figures

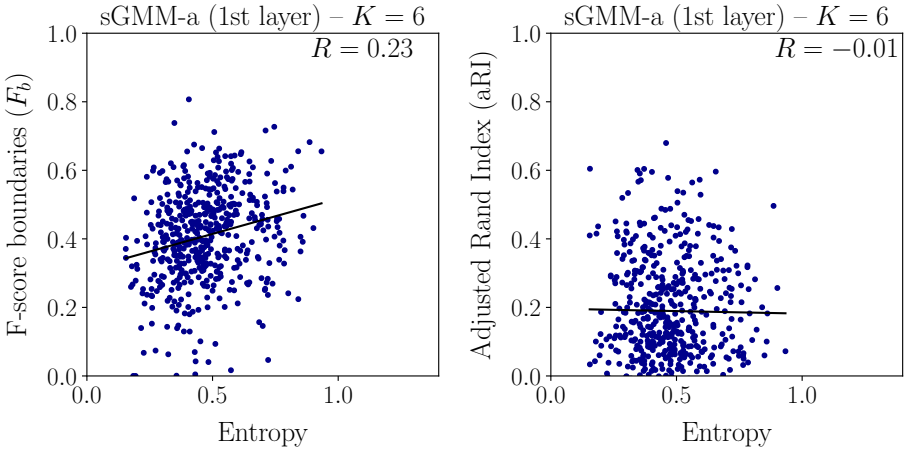


Figure A.12: Positive or null correlation between both scores and entropy model **a**/1st layer when using Gaussian mixtures.

Appendix B. Proof of propositions 1 and 3

Proof of Proposition 1. Using the Dirichlet prior (12) of the main paper, the completed log-posterior writes

$$\ell\left(\boldsymbol{\theta}; (x_n, C_n, \mathbf{B}_n)_n\right) = \sum_{n=1}^{N}\sum_{k=1}^{K}(B_{n,k} - 1 + \mathbb{1}_k(C_n))\ln\left(p_{n,k}\right) + W((x_n, C_n, \mathbf{B}_n)_n; \boldsymbol{\alpha}),$$

where W is the function that gathers all the terms of ℓ that do not depend on $p_{n,k}$. Knowing the previous parameter estimate $\boldsymbol{\theta}^{(t)}$, the E-step consists in taking the conditional expectation of the log-posterior ℓ which is

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n)=\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}\left(B_{n,k}-1+\mathbb{1}_k(C_n)|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)\ln\left(p_{n,k}\right)+w((x_n)_n;\boldsymbol{\alpha},\boldsymbol{\theta}^{(t)}),$$

where $w((x_n)_n;\boldsymbol{\alpha},\boldsymbol{\theta}^{(t)})=\mathbb{E}\left(W((x_n,C_n,\mathbf{B}_n)_n;\boldsymbol{\alpha})|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)$ and

$$\begin{aligned}\mathbb{E}\left(B_{n,k}-1+\mathbb{1}_k(C_n)|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)&=\mathbb{E}\left(u_{n,k}(\mathbb{1}_k(C_\cdot))|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)\\&=u_{n,k}\left(\mathbb{E}\left(\mathbb{1}_k(C_\cdot)|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)\right)=u_{n,k}\left(\tau_{\cdot,k}^{(t)}\right),\end{aligned}$$

where $\tau_{n,k}^{(t)}=\mathbb{P}_{C_n|X_n,\boldsymbol{\Theta}}(k|x_n,\boldsymbol{\theta}^{(t)})$. Then, the M-step consists in maximizing the expected log-posterior Q with respect to $\boldsymbol{\theta}=(\mathbf{p}_n,\boldsymbol{\alpha})$. We only consider optimization with respect to $\mathbf{p}_n$ which is independent from the optimization with respect to $\boldsymbol{\alpha}$. To obtain the update rule for $\mathbf{p}_n$ with first add the Lagrange multiplier associated to the constraint $\sum_k p_{n,k}=1$ and compute the partial derivative with respect to $p_{n,k}$. Therefore,

$$\forall(n,k)\in\{1,\dots,N\}\times\{1,\dots,K\},\quad\frac{u_{n,k}\left(\tau_{\cdot,k}^{(t)}\right)}{p_{n,k}}+\lambda_n=0,$$

which leads to the update rule (26) by setting λ_n such that $\sum_k p_{n,k}=1$. □

Proof of Proposition 3. The proof is similar to the proof of Proposition 1 and starts by writing the log-posterior of each layers and then taking the conditional expectation knowing all the other features and previous parameter estimations at all layers

$$\begin{aligned}\mathbb{E}\left(B_{n,k}^{(h)}-1+\mathbb{1}_k(C_n^{(h)})\middle|(x_n)_n,\boldsymbol{\theta}^{(t)}\right)&=\mathbb{E}\left(u_{n,k}^{(h)}\left(\mathbb{1}_k(C_\cdot^{(1)}),\dots,\mathbb{1}_k(C_\cdot^{(H)})\right)\middle|\left((x_n^{(h)})_n,\boldsymbol{\theta}^{(t,h)}\right)_h\right)\\&=u_{n,k}^{(h)}\left(\mathbb{E}\left(\left(\mathbb{1}_k(C_\cdot^{(1)}),\dots,\mathbb{1}_k(C_\cdot^{(H)})\right)\middle|\left((x_n^{(h)})_n,\boldsymbol{\theta}^{(t,h)}\right)_h\right)\right)\\&=u_{n,k}^{(h)}(\tau_{\cdot,k}^{(t,1)},\dots,\tau_{\cdot,k}^{(t,H)}).\end{aligned}$$

We conclude by using Lagrange multipliers as in the proof of Proposition 1. □

Appendix C. Proof of propositions 2

Proof that the EM algorithm increases the log-posterior of mixture models. First, let us recall why each iteration of the EM algorithm applied to a general probabilistic mixture models as presented in the introduction increases the (incomplete) log-likelihood. At iteration $t+1$, the E-step of the algorithm consists in computing

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n)=\mathbb{E}_{C|X,\boldsymbol{\theta}^{(t)}}\left(\ln\ \mathbb{P}_{(X_n)_n,(C_n)_n|\boldsymbol{\theta}}((x_n)_n,(C_n)_n|\boldsymbol{\theta})\right),$$

while the M-step maximizes $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n)$ with respect to $\boldsymbol{\theta}$. For clarity purposes, we drop probability subscripts in the following. First, using the chain rule, we have $\mathbb{P}((x_n)_n,(C_n)_n|\boldsymbol{\theta})=\mathbb{P}((C_n)_n|(x_n)_n,\boldsymbol{\theta})\mathbb{P}((x_n)_n|\boldsymbol{\theta})$, so for all $\boldsymbol{\theta}$,

$$\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}))=\ln(\mathbb{P}((x_n)_n,(C_n)_n|\boldsymbol{\theta}))-\ln(\mathbb{P}((C_n)_n|(x_n)_n,\boldsymbol{\theta})).$$

Then,

$$\mathbb{E}_{C|X,\boldsymbol{\theta}^{(t)}}(\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}))) = \mathbb{E}_{C|X,\boldsymbol{\theta}^{(t)}}(\ln(\mathbb{P}((x_n)_n,(C_n)_n|\boldsymbol{\theta}))) - \mathbb{E}_{C|X,\boldsymbol{\theta}^{(t)}}(\ln(\mathbb{P}((C_n)_n|(x_n)_n,\boldsymbol{\theta}))$$
$$\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta})) = Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n) + H\left((C_n)_n|(x_n)_n,\boldsymbol{\theta}^{(t)},(C_n)_n|(x_n)_n,\boldsymbol{\theta}\right),$$

where $H(p,q)$ is the cross-entropy of the probability distributions p and q. The previous equation holds for all parameters $\boldsymbol{\theta}$ so, by subtracting the same equation for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$, we obtain

$$\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta})) - \ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}^{(t)})) = Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n) - Q(\boldsymbol{\theta}^{(t)};\boldsymbol{\theta}^{(t)},(x_n)_n) +$$
$$H\left((C_n)_n|(x_n)_n,\boldsymbol{\theta}^{(t)},(C_n)_n|(x_n)_n,\boldsymbol{\theta}\right) - H\left((C_n)_n|(x_n)_n,\boldsymbol{\theta}^{(t)},(C_n)_n|(x_n)_n,\boldsymbol{\theta}^{(t)}\right).$$

The Gibbs' inequality [40] states that for all probability distributions p and q, $H(p,q) \geqslant H(p,p)$, so

$$\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta})) - \ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}^{(t)})) \geqslant Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n) - Q(\boldsymbol{\theta}^{(t)};\boldsymbol{\theta}^{(t)},(x_n)_n).$$

At iteration $t+1$; the M-step consists in defining $\boldsymbol{\theta}^{(t+1)}$ such that it maximizes $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n)$. Therefore,

$$\ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}^{(t)})) - \ln(\mathbb{P}((x_n)_n|\boldsymbol{\theta}^{t})) \geqslant 0.$$

□

Proof that EM algorithm increases the log-posterior of our models. Similarly, in our framework, each iteration the EM-algorithm applied to the complete log-posterior increases the incomplete log-posterior. $\boldsymbol{\theta}$ denotes the model parameters $\boldsymbol{a}, (\boldsymbol{p}_n)_n$. At iteration $t+1$, the E-step computes

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n) = \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}\left(\ln\mathbb{P}_{\boldsymbol{\theta}|X,C,\mathbf{B}}(\boldsymbol{\theta}|(x_n)_n,(C_n)_n,(\mathbf{B}_n)_n)\right)$$
$$= \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}\left(\ln\left(\mathbb{P}_{X,C,\mathbf{B},\boldsymbol{\theta}}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n,\boldsymbol{\theta})\right) - \ln\left(\mathbb{P}_{X,C,\mathbf{B}}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n)\right)\right),$$

and the M-step maximizes $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n)$ with respect to $\boldsymbol{\theta}$. Once again, we drop the probability subscripts for clarity purposes and using the chain rule, we have

$$\mathbb{P}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n,\boldsymbol{\theta}) = \mathbb{P}((C_n)_n,(\mathbf{B}_n)_n|(x_n)_n,\boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta}|(x_n)_n)\mathbb{P}((x_n)_n),$$

so for all $\boldsymbol{\theta}$,

$$\ln\mathbb{P}(\boldsymbol{\theta}|(x_n)_n) = \ln\left(\mathbb{P}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n,\boldsymbol{\theta})\right) - \ln\left(\mathbb{P}((C_n)_n,(\mathbf{B}_n)_n|(x_n)_n,\boldsymbol{\theta})\right) - \ln\left(\mathbb{P}((x_n)_n)\right).$$

Then,

$$\mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}(\ln\mathbb{P}(\boldsymbol{\theta}|(x_n)_n)) = \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}(\ln\mathbb{P}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n,\boldsymbol{\theta}))$$
$$- \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}(\ln\mathbb{P}((C_n)_n,(\mathbf{B}_n)_n|(x_n)_n,\boldsymbol{\theta})) - \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}(\ln\mathbb{P}((x_n)_n))$$

$$\ln\mathbb{P}(\boldsymbol{\theta}|(x_n)_n) = Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(t)},(x_n)_n) + \mathbb{E}_{C,\mathbf{B}|X,\boldsymbol{\theta}^{(t)}}\left(\ln\left(\mathbb{P}_{X,C,\mathbf{B}}((x_n)_n,(C_n)_n,(\mathbf{B}_n)_n)\right)\right)$$
$$+ H((C_n)_n,(\mathbf{B}_n)_n|(x_n)_n,\boldsymbol{\theta}^{(t)},(C_n)_n,(\mathbf{B}_n)_n|(x_n)_n,\boldsymbol{\theta}) - \ln\mathbb{P}((x_n)_n),$$

As before, by subtracting the same equation for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ and thanks to the Gibbs' inequality, we obtain

$$\ln \mathbb{P}(\boldsymbol{\theta}|(x_n)_n) - \ln \mathbb{P}(\boldsymbol{\theta}^{(t)}|(x_n)_n) \geqslant Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}, (x_n)_n) - Q(\boldsymbol{\theta}^{(t)}; \boldsymbol{\theta}^{(t)}, (x_n)_n).$$

The M-step at iteration $t+1$ consists in defining $\boldsymbol{\theta}^{(t+1)}$ such that it maximizes $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}, (x_n)_n)$. Therefore,

$$\ln \mathbb{P}(\boldsymbol{\theta}^{(t+1)}|(x_n)_n) - \ln \mathbb{P}(\boldsymbol{\theta}^{(t)}|(x_n)_n) \geqslant 0.$$

Note that the proof is similar for our model combining mixture models. □