Learning to be attractive: probabilistic computation with dynamic attractor networks

Alexander Gepperth[1] and Mathieu Lefort[2]

Abstract— **In the context of sensory or higher-level cognitive processing, we present a recurrent neural network model, similar to the popular dynamic neural field (DNF) model, for performing approximate probabilistic computations. The model is biologically plausible, avoids impractical schemes such as log-encoding and noise assumptions, and is well-suited for working in stacked hierarchies. By Lyapunov analysis, we make it very plausible that the model computes the maximum a posteriori (MAP) estimate given a certain input that may be corrupted by noise. Key points of the model are its capability to learn the required posterior distributions and represent them in its lateral weights, the interpretation of stable neural activities as MAP estimates, and of latency as the probability associated with those estimates. We demonstrate for in simple experiments that learning of posterior distributions is feasible and results in correct MAP estimates. Furthermore, a pre-activation of field sites can modify attractor states when the data model is ambiguous, effectively providing an approximate implementation of Bayesian inference.**

I. Introduction

A. Context and scope

This contribution is conducted in the context of cognitive neural modeling, essentially trying to elucidate how neural populations can perform probabilistic computations which are speculated to play an important role in the considerable performance of biological brains. However, especially the currently influential Bayesian paradigm, stating that the basic quantities to be manipulated and passed around should be probability distributions, poses considerable difficulties. Considering the limited operations available to neurons (essentially just weighted summation), the following questions may be asked: How should such distributions be represented on a neural level? How can the multiplications required for manipulating them be performed, not to speak of the necessary renormalizations after each such step? And, lastly, how can neural populations take optimal decisions based on distributions? Please see Sec. V for a discussion of these questions in hindsight.

This article presents a network model that proposes answers to some of these questions, is biologically plausible (see Sec. V for a discussion of this claim) and yet computationally (rather) inexpensive. The basic properties of the model, some of which are already demonstrated in [1], some of which shall be demonstrated or at least made plausible here, are as follows:

- **Bayesian processing is approximate**: we approximate full posterior probabilities in each layer by their argmax (the MAP estimate) and its associate posterior probability, see Fig. 1 for a visualization of this process.
- **Neural activities represent MAP estimates**: given an afferent input, we assume that neural dynamics, guided by lateral connections, converge to a MAP estimate given the input. Neural activities are therefore not at all related to probability distributions.
- **Weights represent distributions**: for ensuring convergence towards MAP estimates, the lateral connections must represent in some way a posterior distribution of the data
- **posterior distributions are learned**: as a model with pre-specified posterior distributions would be useless in practice, we propose a simple learning scheme for adapting the lateral connections to data statistics

The basic functionality achieved by the model is depicted in Fig. 2: following a period of training with clean stimuli, the model receives a noisy stimulus and attempts to converge to the noise-free ("true") underlying stimulus based on the results of the training phase. In this contribution, we shall make it very plausible (by considering the Lyapunov functional of the model) that the model in fact converges to the maximum a posteriori (MAP) estimate for the given

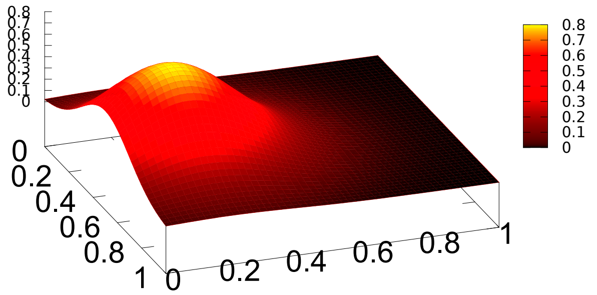

Fig. 1. Illustration of the basic approximation process proposed in this article: the approximation of a full probability distribution by its argmax (the MAP estimate), and its associated probability. We assume a 2D neural layer where each neuron represents a certain stimulus, and a probability distribution defined over stimuli (and thus over neuronal sites). The depicted Gaussian distribution can then be approximated by the three numbers: 0.2 and 0.5 for the "position" of the MAP state, as well as 0.75 for its probability under the given distribution. In our neural implementation, these quantities are expressed by the 2D position of the attractor state as well as its latency (time to convergence).

[1] UIIS lab, FLOWERS team, INRIA, ENSTA ParisTech, 828 Blvd des Marechaux, 91762 Palaiseau (France), alexander@gepperth.net
[2]LIRIS laboratory, university Lyon 1, Lyon (France), mathieu.lefort@liris.cnrs.fr

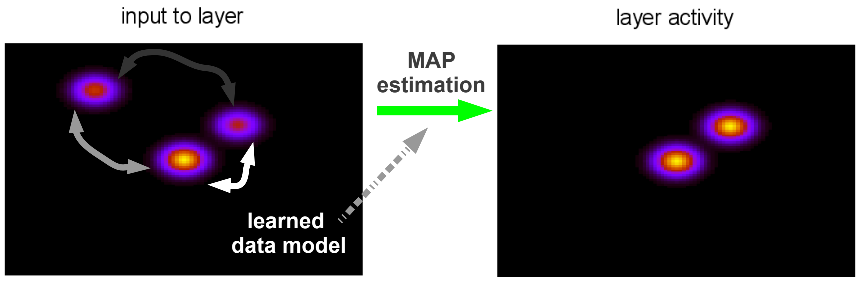

Fig. 2. Functionality of the proposed neural model. Given an afferent synaptic input (left), supposed to be population-coded so that activity at a certain place codes for a certain quantity/concept, a neural layer converges to an attractor state (right) with activity corresponding to the MAP estimate of the underlying true stimulus. Input is supposed to be a noisy version of this stimulus, which is recovered from input by making use of learned statistical properties of the data, indicated on the left by synaptic connections whose strength (indicated by brightness) represents the probability of co-activation. The time until the attractor state is reached is a measure for the posterior probability of the MAP estimate. In this concrete example, the left stimulus is suppressed because it has zero probability of co-occurrence with the right stimulus, and a weaker probability of co-occurrence with the central stimulus.

stimulus. We have already shown in [1] that the time to convergence to this estimate, denoted *latency*, is a measure of its actual probability under the posterior distribution.

It is these two quantities, the MAP estimate and its probability, that are therefore passed to subsequent neural layers, thus replacing the actual posterior distribution by an unimodal approximation (see Fig. 1 for a visualization). This approximation allows a mapping of probabilistic processing to a neural model, as the competitive dynamics prevents the representation of arbitrary distributions in the form of neural activities. Lastly, this model can be explicitly fed with prior distributions by pre-activating certain field sites, thus approximating Bayesian inference in the course of the MAP calculation process.

B. Rigorous problem formulation

The basic representational unit we consider here is a neural layer X where neural activities (firing rates) $\vec{z}^X(t)$ are arranged on a rectangular grid.

For a layer receiving an input $\vec{I}$, we suppose the input to be a noisy version of the underlying "true" stimulus $\vec{T}$. Input and underlying stimulus are related by the probabilistic input model $p(\vec{I}|\vec{T})$. The basic task we wish to accomplish is the estimation of the most likely "true" stimulus $\vec{\mathcal{T}}^*$ given the current input $\vec{I}$, as well as its associated probability $p(\vec{T} = \vec{\mathcal{T}}^*|\vec{I})$. In other words, what we are looking for is the so-called maximum a posteriori (MAP) estimate given the input stimulus $\vec{I}$. This task must involve some way of estimating the "inverse" distribution $p(\vec{T} = \vec{\mathcal{T}}|\vec{I})$ which is related to the input model by the law of Bayes:

$$p(\vec{T} = \vec{\mathcal{T}}|\vec{I}) = \frac{p(\vec{I}|\vec{T} = \vec{\mathcal{T}})p(\vec{T} = \vec{\mathcal{T}})}{p(\vec{I})} \tag{1}$$

.

Summing up, we wish to approximate the following quantities:

$$\begin{aligned} \vec{\mathcal{T}}^* &= \text{argmax}_{\vec{\mathcal{T}}} p(\vec{T} = \vec{\mathcal{T}}|\vec{I}) \\ p^* &= p(\vec{T} = \vec{\mathcal{T}}^*|\vec{I}) \end{aligned} \tag{2}$$

Furthermore, we wish to influence the outcome of this computation by specifying priors for the "true" stimulus $\vec{T}$. This is totally consistent with Bayes' formula, which in our case states that the basic distribution $p(\vec{T} = \vec{\mathcal{T}}|\vec{I})$ can be re-stated as

$$p(\vec{T} = \vec{\mathcal{T}}|\vec{I}) \sim p(\vec{I}|\vec{T} = \vec{\mathcal{T}})p(\vec{T} = \vec{\mathcal{T}}), \tag{3}$$

where the a priori probability for the true stimulus appears explicitly on the right-hand side of the equation.

C. Modeling approach: competitive neural dynamics and space-latency coding

We will demonstrate by Lyapunov analysis that a competitive neural dynamic approach very similar to the dynamic neural field (DNF) model [2] converges to an attractor state that is maximally compatible with afferent and lateral input, the latter defined by lateral connections. Learning and representation of the distribution $p(\vec{T} = \vec{\mathcal{T}}|\vec{I})$ required for the computation of the MAP estimate in eqn.(2) is realized by the lateral connections weights by linear regression. If the lateral input thus obtained can be associated with a likelihood distribution, this would make it very plausible that the attractor state of the model indeed corresponds to, or at least approximates, the MAP estimate[1].

In previous works on the subject [1], we already showed that this model can represent both the MAP estimate and its probability under the posterior distribution, $\vec{\mathcal{T}}^*$ and p^*, by its attractor state and the associated latency (time-to-convergence). This naturally gives rise to a novel way of encoding information, which we term space-latency coding [1], into neural populations, which is exclusively due to the dynamic properties of the DNF model. We already showed in [3], [4] that this space-latency code is well suited for transmitting information in a neural hierarchy, and demonstrated its appropriateness for realistic information processing.

[1]We would be delighted about rigorous mathematical proofs on this subject

D. Related work

There exists a large body of literature [5], [6], [7], [8], [9], [10] on probabilistic aspects of neural coding. Most authors explicitly assume that neural population activity is directly related to probability distributions [5], [6], [8], [9], [10]; posing the question of how to appropriately multiply and renormalize neural activities in a biologically plausible way. A very influential idea for addressing this issue posits that single-neuron activity is in fact related to log-probability [9], [10], [7], which would allow to perform multiplication by summation, which neurons do easily. Other authors have questioned the practicability of this scheme [5] as it would require all sorts of cut-offs due to the unbounded nature of logarithms as probabilities go to zero, and which would have to be performed at each hierarchy level which is deemed unfeasible and lossy. An alternative approach [5] is to consider single neuron's firing rates as the realizations of Poisson-like random variables whose mean is determined by the match of neural preferences to afferent input. Under certain conditions, sums of two such variables can be proven to come from a distribution whose mean corresponds to the product of individual means, thus realizing a multiplication by summation.

E. Novelty

When comparing our approach to related work, several differences are evident: First of all, and different from [7], [10], [9], our approach does not treat neural activities as log-probabilities. More generally, and in line with [5], we do not treat the set of input activations $\vec{S}$ as a probability distribution but as the expression of match to individual neuron's preferences, corrupted by noise. In contrast to [5], however, we do not require this noise to have any particular form in order for our approach to work. Conversely, our approach does not profit from noise but tries to remove it to recover the underlying stimulus. This will work best if the noise is not on the single-neuron level but consists, e.g., in the apparition or fluctuation of local activity bubbles. A further, related difference to [5], who treat each neuron in absolute isolation from is neighbours, is that we consider data models that are global in the sense that they consider the values of other, not necessarily adjacent neurons in the approximation of the MAP estimate. In contrast to all of the approaches listed here, our model alone exploits the fact of using a dynamic model for encoding information, in this case by latency. Furthermore, as we could show in previous work [1], [3], our model accounts for the encoding, the transmission and the decoding of this so-called *space-latency code* in a deep hierarchy as a direct consequence of its dynamics. Lastly, the idea of using a DNF-like model with a plastic, learned kernel is, to our knowledge, unique w.r.t. to related work in the subject. Summarizing, this article suggests a new way of approximately representing and processing probabilistic information in neural hierarchies which is quite different from what has been proposed in previous works, although it will be validated only for very simple stimuli. To be fair, on the other hand, most previous work on the subject uses test stimuli of similar simplicity with the possible exception of [10].

F. Goals of the article

As this article extends our previous work mainly by the aspects of Lyapunov analysis as well as the learning of lateral connection weights, we wish to show that

- convergence towards the MAP estimate is plausible
- learned lateral connections (empirically) produce the correct MAP estimates in simple cases
- the space-latency code produced by learned connections is consistent with data statistics
- pre-activation of field sites acts as a prior distribution in ambiguous cases

II. Methods

Throughout the paper, we will make use of an extension of the dynamic neural field (DNF) model [2], [11], which represents two-dimensional sheets of rate-coded model "neurons" who have an internal state termed the "membrane potential" that evolves according to a differential equation taking into account the membrane potential, afferent synaptic input and lateral input mediated via a symmetric weight matrix termed "convolution kernel" or "interaction kernel". In its original formulation, the DNF model assumes infinitesimal model neurons, which allows analytical treatments of simple situations which would otherwise be impossible. In this article, all insights are gained through simulation experiments for which naturally a neural sheet has to be discretized.

A. Notation

We will consider, for simplicity of notation, the case of discretized fields in two dimensions having a size of (N, M) elements. The set of membrane potentials in the whole field at a given instant shall be denoted $\vec{u}(t)$, a particular activity value $u_i(t)$ being identified by its one-dimensional index i. Please note that this is done for simplicity of notation: the field is considered to be two-dimensional. The dimension of the field is in fact irrelevant except in the definition of the lateral interaction, as all the other operations updating the field are applied in an element-wise fashion. The element-wise application of a non-negative, monotonous transfer function f to all field potentials, giving the firing rate vector $\vec{z}(t)$ shall be denoted as $\vec{z}(t) = f[\vec{u}(t)]$. In this article, we will always a transfer function of the form

$$f(x) = \frac{1}{1 + \exp\left(-\frac{x-\theta}{\nu}\right)}, \tag{4}$$

with two free parameters, the *threshold* θ and the *slope* ν. The application of a symmetric 2D convolution kernel K of size $(N', M') < (N, M)$, represented by a vector of coefficients $(k_j), j = 0 \ldots N' \times M'$, to a vector of firing rates $\vec{z}(t)$ (producing the transformed version $\vec{z}'$), is denoted by

$$z'_i = (K * \vec{z})_i. \tag{5}$$

The afferent input vector to a neural field is denoted by $\vec{I}$, whose entries we suppose to be always positive and normalized in the interval $[0, 1]$. Using this notation, the classical DNF model evolves according to:

$$\tau \dot{u}_i = -u_i + I_i + (K * f[\vec{u}])_i + h, \qquad (6)$$

h denoting the so-called *resting potential*.

B. Lyapunov analysis of the DNF model

Lyapunov analysis denotes the analysis of the asymptotic behavior of a dynamical system by means of a Lyapunov functional $\mathcal{L}$, more commonly termed "energy functional" by analogy to physics. This is a functional that maps the current system state to a real number (the current "energy"): $\mathcal{L}[\chi(t)] \to \mathbb{R}$, and which has the property that it is being minimized by the system's dynamics. In addition, the Lyapunov functional is bounded from below, which means that at least one stable state exists, the ground state where $\mathcal{L}$ reaches the lower bound and thus cannot decrease any further, making the state stable. All these properties make Lyapunov analysis interesting for studying non-linear dynamic systems such as the extended DNF model which is proposed here, given of course that a Lyapunov functional exists, which is not self-evident and has to be rigorously proven for each dynamical system under consideration. Fur a more in-depth discussion of Lyapunov analysis, see [12].

In [13], a Lyapunov functional for the DNF model was shown to exist, and an explicit expression for it was presented. In our notation, it has the following form:

$$\mathcal{L}[\vec{u}(t)] = -\sum_i z_i(I_i - h) - \frac{1}{2}\sum_i z_i(K * \vec{z}_i)_i + \sum_i f^{-1}(z_i) =$$
$$= -\left\langle \vec{z}, \vec{I} - h \right\rangle - \frac{1}{2}\left\langle \vec{z}, K * \vec{z} \right\rangle + \sum_i F(z_i), \qquad (7)$$

where $F(z)$ denotes a monotonous function of z which is derived from the transfer function f. As we can see, the first two terms can be restated as scalar products, whereas the third acts as a bound on total field activity. In qualitative terms, the evolution of a neural field thus aims simultaneously to achieve the highest possible similarity between firing rate and input, between firing rate and lateral input, all of this while maintaining a bounded overall activity.

C. Extension of DNF model

We now propose the following formal generalizations:

- Position-dependent global kernel: in order to learn complex data models as shown in Fig. 2, separate lateral interaction kernels must be learned for all neuronal sites. Furthermore, these local kernels must take into account all other field activities and not just adjacent ones, as statistical relations will not necessarily be local in the 2D plane. We therefore replace the convolution operation in eqn.(6) by a simple linear transformation L, which amounts to a normal matrix multiplication.
- Separation into excitatory and inhibitory lateral interactions: We wish to split the lateral interaction into a part that always gives positive results and which is potentially plastic, and a part that always acts in an inhibitory fashion and is fixed, ensuring a global attractor dynamics of the model. Therefore the lateral interaction term in eqn. (6) is split into a negative part with a constant coefficient γ, and a positive part depending on the position-dependent global kernel L.
- Learning of excitatory lateral interactions: A suitable learning algorithm adapts the excitatory part of lateral interactions (parametrized by L) to the statistics of input data

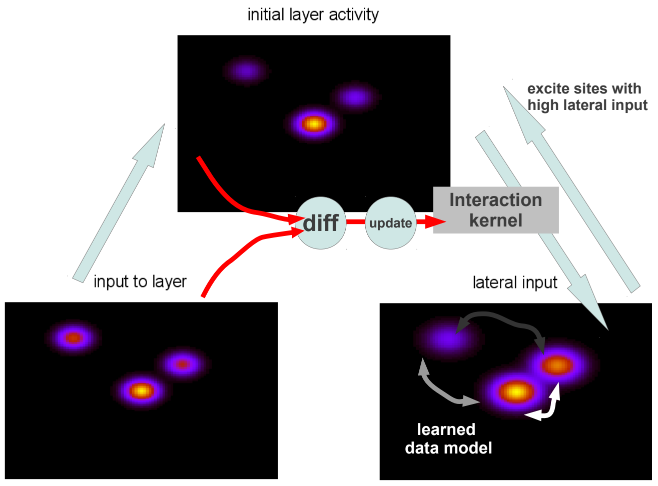

Fig. 3. Schematic sketch of the learning of kernel matrices, i.e., lateral connection weights. Basically, weights are adapted so that the difference between field activity (center,top) and input (left,bottom) is minimized. Modified kernel matrices act upon field activities in a non-linear way via lateral inputs (right,bottom).

All of this motivates us to reformulate the DNF equation (6) as follows, additionally introducing the coefficients α, β and γ before key terms:

$$\tau \dot{u}_i = -u_i + \alpha I_i + \beta(Lf[\vec{u}])_i - \gamma \sum_i z_i + h, \qquad (8)$$

where we introduce a symmetric linear transformation $L = (l_{ij})$ to a vector of firing rates $\vec{z}$, producing the transformed version $\vec{z}'$ of same dimensionality (the matrix L is therefore square and has dimensions $(N * M, N * M)$):

$$z'_i = (L\vec{z})_i = \sum_j l_{ij} z_j,$$

which does not in any way change the Lyapunov function of the model given in eqn.(7) as the lateral interaction terms are effectively still described by a symmetric convolution kernel (the matrix L which is supposed symmetric, and the constant γ which is position-independent and thus symmetric by definition. As long as the learning rule for L therefore guarantees symmetry of L, according to [13], the Lyapunov functional stays well defined.

D. Learning of kernel matrices

We suppose that the task of the kernel is make fields converge, starting with an empty field, to the underlying true stimulus that has the highest probability given the input. This will depend upon statistical properties of the data which are encoded in the kernel matrices L by a learning process. Assuming an unsupervised learning scenario where the true stimuli $\vec{T}$ are unavailable, we suppose that the noise on inputs is such as to cancel by averaging over longer time scales. Thus, we are in an "autoencoder" type of learning scenario, where kernel matrices are adapted to make the field state reproduce the (average) input, see Fig. 3. In the experiments presented here, we do not even go this far as all training stimuli are "clean", and just test stimuli are noisy. In this case, learning supports pure disambiguation, i.e., deciding which training stimulus is closest to the noisy test stimulus. We therefore propose a simple linear regression model that adapts kernel values guided by the difference between neural field state (i.e., firing rates) and inputs, which we suppose normalized in the same interval $[0, 1]$:

$$l_{ij}(t+1) = l_{ij}(t) + 2\epsilon z_j \left[(L\vec{z})_i - I_i \right], \tag{9}$$

. This is stochastic gradient descent approximating a mean-squared-error loss function, which has the advantage of online learning capability: eqn. (9) is applied after each iteration of the field using a suitable value of ϵ in order to learn at the correct time scale.

III. MAP calculations: how and why it works

When performing Bayesian inference, the terms appearing in the likelihood and in the a priori distribution are usually of different quality. An example would be $p(\text{rain=yes}|\text{season=summer}) \sim p(\text{season=summer}|\text{rain=yes})$ $p(\text{rain=yes})$, where the quantities "rain" and "season" live in different spaces that are not directly related. In contrast to this, in our case the true stimulus $\vec{T}$ and the input $\vec{I}$ live essentially in the same space, which does not in the least invalidate our approach as we can still define meaningful likelihoods and priors. What a neural field does is really a sort of auto-encoder process, trying to derive a less noisy version of its own input according to a learned data model. This could be of immense practical value if learning could efficiently be conducted in an unsupervised manner, which is exactly what we propose in Sec. II-D.

A. How

Supposing a kernel matrix L has been learned on a set of input samples following a certain distribution, we draw samples from the same distribution and put them as input to a neural field with kernel L (and parameters to be specified in more detail in Sec. IV). This input will evoke a time-varying firing rate $\vec{z}(t)$ that will converge after a certain time. Time-to-convergence can be measured in various ways, most simply by demanding that the sum over absolute values of the left-hand side of the extended DNF equation (8) be smaller than a threshold:

$$\tau \sum_i |\dot{u}_i| < \Omega \tag{10}$$

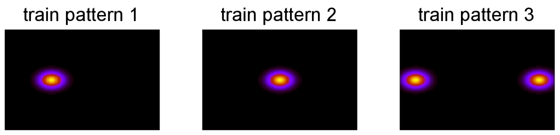

Fig. 4. 100x100 pixel input patterns used for training the lateral connection weights of the model (axes are scaled differently).

As will be shown in the rest of this section, the converged attractor state represents the maximum a posteriori (MAP) estimate of the true stimulus $\vec{T}$.

B. Why

As we supposed that the kernel matrix L has been learned in such a way as to transform the current field activity $\vec{z}(t)$ into a superposition of possible true stimuli, $\vec{z}'(t) = L(t)\vec{z}(t)$. This is, as explained in Sec. II-D, because we assume that noise influences are weak and cancel each other over the course of the learning process. From the Lyapunov analysis in Sec. II-B we know that the field will evolve such that its firing rate resembles as closely as possible the input *and* $\vec{z}'$ while maintaining a bounded activity. This process acts as a disambiguation between the possible true stimuli, represented as a superposition, only one of which will be selected due to global competitive dynamics.

Due to the nature of linear regression, which is sensitive to the relative frequency of training samples, the sum in $\vec{z}'$ will already be weighted by the prior probability $p(\vec{T} = \vec{\mathcal{T}})$ of each true stimulus $\vec{\mathcal{T}}$. By applying preshape in the form of certain true stimuli, or by adding an attenuated version of the stimulus we wish to enhance to the input, we can manipulate these prior probabilities directly, although the exact amount of probability that is added by a certain amount of preshape is difficult to quantify for the moment.

IV. Experiments

All experiments are conducted using the exact same parameters and differ just by the nature of the input stimuli. The parameters, as defined in Sec. II-C, are the following: $\alpha = \beta = 2$, $\gamma = 0.15$, $h = -1$, $\theta = 0$, $\nu = 2.5$,$\tau = 25$. To ensure numerical stability, field potentials are clipped to the range $[-2, 5]$ after each iteration. The neural field itself has a size of 100x100 elements, and so do the inputs. Input patterns are always presented for $N = 400$ iterations, and

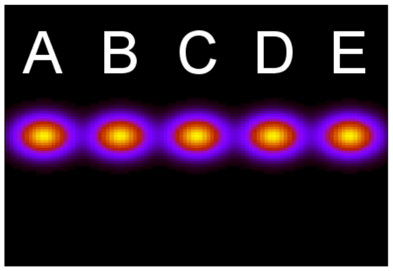

Fig. 5. Discrete positions of stimuli in the test patterns used for subsequent experiments. At each discrete position, a Gaussian stimulus can appear, either in isolation or in combination with Gaussians at other positions.

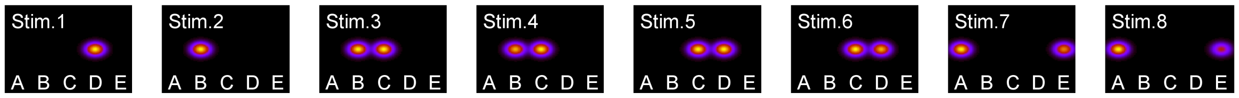

Fig. 6. The eight 100x100 pixel test stimuli used in the experiments, consisting of superpositions of Gaussians (axes are scaled differently). Gaussian amplitudes are 1.0 where not otherwise stated. In stimulus 4, the Gaussian at site B has amplitude 0.9. In stimulus 6, the Gaussian at site D has amplitude 0.8. In stimulus 8, the Gaussian at site D has amplitude 0.5.

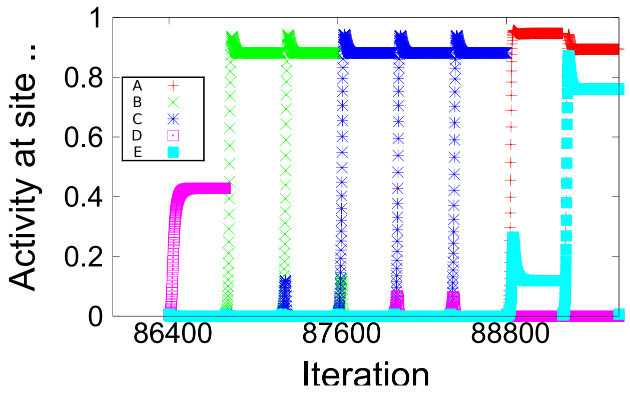

Fig. 7. Overall field responses to the 8 test stimuli, presented sequentially for 400 iterations each starting directly after the training phase at $t = 86400$ iterations. The curves correspond to field activity at each of the sites A-E defined in Fig. 5. Please see text for a discussion of the significance of these results. Best viewed in color.

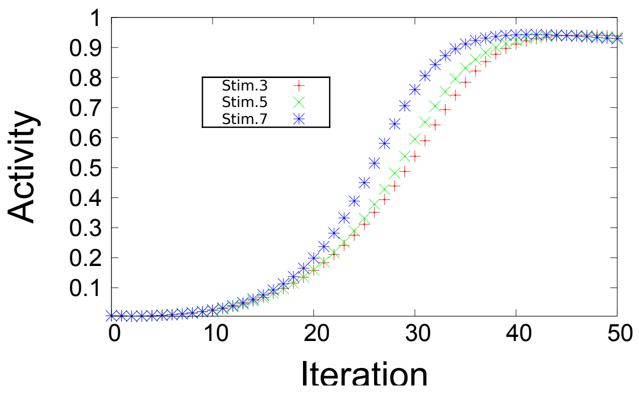

Fig. 8. High temporal resolution of the development of field activities at "winning" sites for stimuli 3 (strong competition), 5(weak competition) and 7 (strong cooperation). It is evident that the time to convergence (latency) depends on the amount of competition in the field. This is exactly what we need to find if we wish to interpret latency as the posterior probability of the converged state.

fields are initialized to h before each pattern presentation. The lateral connection learning rate is set to $\epsilon = 0.001$. Common to all experiments is a training phase, where the three training patterns (see Fig. 4) are repeatedly presented, one after the other, for a total duration of 86400 iterations. As each pattern presentation lasts 400 iterations, this amounts to 216 pattern presentations, or 72 presentations for each pattern. Each test pattern (see Figs. 5, 6) is presented once for $N = 400$ iterations. Inputs are superpositions of unnormalized Gaussians having the same variance of 3 "pixels". For simplicity, each Gaussian can appear at one of 5 predetermined positions, see Fig. 5 for details. The test stimuli used to perform experiments are depicted in Fig. 6. Some Gaussians are multiplied by a constant factor, reducing their amplitude in order to test a particular point. Results are presented in two forms: first of all, we wish to show the overall behavior of the field at all of the sites A-E. For this purpose, we simply plot the maximal value in a small region around each of these sites over time, which is shown in Fig. 7. Results for each pattern can be stated as follows:

- Stimulus 1: since the stimulus at site A was never observed during training (compare with the training stimuli depicted in Fig. 4, it can not create any lateral input to support itself and is thus strongly suppressed by global inhibition.
- Stimulus 2: the stimulus at site B was observed during training, leading to a lateral response that sustains activity despite inhibition.
- Stimulus 3: here, we observe competition between two stimuli who have same amplitude, and who are both supported by lateral input as they were part of the training stimuli. Competition occurs as they never appeared together during training, and thus did not form mutually supporting lateral connections. Competition finally suppresses one of the two Gaussians in a random fashion. By putting a very slight Gaussian preshape, i.e., a pre-activation of the field potential with amplitude $A = 0.05$ to one of the two sites, the randomness disappears and the site with the preshape always wins the competition. Preshape can thus effectively play the role of a prior probability distribution.
- Stimulus 4: same as for stimulus 3 except that the Gaussian at site B is weaker than 1.0, so it has a competitive disadvantage and is suppressed.
- Stimuli 5, 6: here, we observe competition between two stimuli as they were never seen together during training. however the stimulus at site C was seen during training and thus supports itself via lateral input, whereas the stimulus at site D does not, and is therefore quickly suppressed depending on its amplitude.
- stimuli 7, 8: interaction between stimuli that were always seen together during training, and therefore have mutually supporting lateral connections. Not only does the weaker stimulus "survive", it is even boosted depending on its original strength.

We therefore observe that the field always converges towards the state that is evidently the most probable one given the training data, see Fig. 4. A second result concerns the continued validity of the space-latency coding now that lateral interactions weights are learned from data statistics. To this end, we consider the stimuli 3, 5 and 7 from Fig. 6. Here, we observe the following interactions between peaks: strong competition for stimulus 3, as both Gaussians are self-supporting via lateral input, weak competition for stimulus 5 as one Gaussian can support itself and the other cannot, and strong cooperation for stimulus 7 as both Gaussians have bilateral exciting connections between them. Stronger competition is intuitively equivalent to lower posterior probability, as competition means that the field and the lateral input (the likelihood) do not match. This is indeed what we observe when we plot the time evolution of the field in response to these three stimuli, magnified so that small latency differences may be perceived (see Fig. 8): stimulus 3 has the largest latency due to strong competition, stimulus 5 has lower latency as competition is weaker, and stimulus 7 has the lowest latency since there is collaboration going on. Summarizing, at least for these simple cases, the space-latency code continues to work correctly by expressing a high level of conflict in a field, corresponding to a low posterior probability, by increased latency. The effect is rather weak here, but by correct parameter choices it can be magnified to visible time scales.

V. Discussion, conclusions and future work

Continuing our previous work on the subject, we have presented a neural model that gives a complete account of how neural systems may process and transmit probabilistic information in a way that is applicable to real-world processing. In particular, the presented model is capable of learning from observations, creating its own internal models of posterior distributions that allow Bayesian inference. We have made it very plausible, by means of Lyapunov analysis, that, if the model learns a true posterior distribution from data, the final attractor state will closely resemble a MAP estimate. Lastly, we have verified for some simple cases that the field indeed converges towards the MAP estimate, and that the latency of the attractor state is a measure of competition in the field, or conflict to the learned posterior distribution.

In Sec. I, three fundamental questions are posed:

1) How can distributions be represented on a neural level?
2) How can the multiplications required for manipulating them be performed, not to speak of the necessary renormalizations after each such step?
3) How can neural populations take optimal decisions based on distributions?

In the light of the results and methods presented here, the answers we give are the following: first of all, no distributions are represented at the neural level, just an approximation given by their dominant mode and its associated probability (represented by latency). Then, as only the dominant mode of the resulting "multiplied" distribution needs to be represented by the following layer, multiplications are indeed unnecessary and can be replaced by a simple competitive method of determining the dominant mode while taking into account the associated probabilities. And lastly, decision making occurs when competitive dynamics "selects" the dominant mode and suppressed the others.

It may be criticized that we give up the idea of a fully "Bayesian brain" in favor of an approximation that involves the MAP estimate and its posterior probability. However, in a behavioral context, it is often only the most likely interpretation $\vec{\mathcal{T}}^*$ of an input that is required to take an action. Furthermore, sub-leading interpretations are just suppressed but continue to exist, and can resurface if additional external input is provided. In addition, we also compute a measure of the posterior probability associated with the MAP estimate, which effectively realizes an unimodal approximation of the posterior distribution. If this probability is low, we know that the other modes of the distribution are non-negligible and we would do well not to take the MAP estimate too seriously. This manner of thresholding and, if need be, of discarding information is very relevant for intelligent agents, as it is in general better to wait for additional inputs than to take decisions based on inputs that are known to be inconclusive.

Although we do not present a spiking neural network model, we nevertheless claim high biological plausibility for the proposed model. Just as the original DNF model, it is formulated at a lower spatial "resolution" and thus not on the single-neuron level but rather on the population level (e.g., minicolumns or macrocolumns). In this context, introducing spiking models would be very inconsistent and is therefore avoided. Moreover, what we primarily aim to model is the dynamical interplay of lateral excitatory and inhibitory influences and its computational significance, especially since it is well-known that lateral connections are subject to learning as well [14]. In this sense, we feel it is well justified to claim biological plausibility for our model.

It is very evident that the next step must be a rigorous proof of the MAP property. Or rather: it must be shown that the lateral connection weights learn a posterior distribution (which will depend largely on the used learning rule). Subsequently, it must be established to what extent, and to what precision, the field converges to the maximum of this distribution. it would be very helpful if, at the same time, some statements could be made about the time to convergence (i.e., the latency) and find a rigorous link to probability here as well.

References

[1] A Gepperth. Processing and transmission of confidence in recurrent neural hierarchies. *Neural Processing Letters*, 2013.

[2] W. Erlhagen and G. Schöner. Dynamic field theory of movement preparation. *Psychological review*, 109(3):545, 2002.

[3] GepperthA and M Lefort. Latency-based probabilistic information processing in a learning feedback hierarchy. In *International Conference on Artificial Neural Networks 2014*, 2014.

[4] A Gepperth and M Lefort. Latency-based probabilistic information processing in recurrent neural hierarchies. In *International Joint Conference on Neural Networks (IJCNN)*, 2014.

[5] WJ Ma, Latham P Beck, J, and A Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11), 2006.

[6] R. S. Zemel, P. Dayan, and A. Pouget. Probabilistic interpretation of population codes. *Neural Comput*, 10(2):403–430, Feb 1998.

[7] J. Gold and M. Shadlen. Neural computations that underlie decisions about sensory stimuli. *Trends Cogn Sci*, 5(1):10–16, Jan 2001.

[8] David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *Trends Neurosci*, 27(12):712–719, Dec 2004.

[9] Raymond H. Cuijpers and Wolfram Erlhagen. Implementing bayes' rule with neural fields. In *ICANN '08: Proceedings of the 18th international conference on Artificial Neural Networks, Part II*, pages 228–237, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] Rajesh P N Rao. Bayesian computation in recurrent neural circuits. *Neural Comput*, 16(1):1–38, Jan 2004.

[11] Shun-ichi Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1441–1463, 1990.

[12] Derek P Atherton. *An introduction to nonlinearity in control systems*. Bookboon, 2011.

[13] Shigeru Kubota and Kazuyuki Aihara. Anayzing global dynamics of a neural field model. *Neural Processing Letters*, 21, 2005.

[14] Risto Miikkulainen, James A Bednar, Yoonsuck Choe, and Joseph Sirosh. *Computational maps in the visual cortex*. Springer Science & Business Media, 2006.