Universitat de les Illes Balears

Facultat de Ciències

Memòria del Treball de Fi de Grau

Modelo de redes neuronales de la criticalidad en la retina

Bartomeu Pou Mulet

Grau de Física

Any acadèmic 2015-16

DNI de l'alumne: 43172131G

Treball tutelat per Claudio Mirasso
Departament de Física

S'autoritza la Universitat a incloure aquest treball en el Repositori Institucional per a la seva consulta en accés obert i difusió en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor		Tutor	
	Sí	No	Sí	No
	X		X	

Paraules clau del treball:
Retina, criticality, artificial neural network , ...

Contents

Chapter 1

Introduction

Since the ancient Greece vision has been object of study, with theories such as "emission" in which vision occurs when rays emanate from the eyes and are intercepted by visual objects.

Nowadays we have much more knowledge about vision but it is a complex phenomenon not yet completely understood. Vision is the predominant sense in the human body, starting with light penetrating into the eye and arriving to the retina, where photosensitive neurons (rods and cones) process visual information, and culminating in deep structures in the brain that recognize and categorize objects in order to interact with the environment.

The retina is the first layer of neural processing of images. A major problem in visual perception is that what people see is not simply a translation of retinal stimuli (i.e., the image on the retina). Thus people interested in perception have long struggled to explain how visual processing creates what is actually seen and perceived.

To understand better the dynamics of the retina the team of William Bialek at Princeton (1) took measures of the retina under natural stimulation. In these measures it was found that the retina operated near a regime of criticality, that is, a regime that separates ordered and chaotic-like dynamics and that exhibits very specific statistical properties.

The goal of this work is to model and study the criticality regime found in retinal neurons. To do so we implement a model of the retina with artificial neural networks to understand the observations described in (1) and test the emergence of criticality in the neural model, in particular respect to the input statistics.

Before showing our results we will describe some background information about thermodynamics and retina to understand Bialek work (1), and about machine learning to describe the context of our model. Later in the Methods section we will explain how our artificial neural model (restricted Boltzmann machine) works and explain how to measure criticality. After our analyses are described in the results section we will conclude with some discussion about the interpretation of the results and future lines to continue this work.

Chapter 2

Background information

2.1 Thermodynamics and signatures of criticality in a network of neurons

The retina *"is the inner coat of the eye which is a light-sensitive layer of tissue. Light striking the retina initiates a cascade of chemical and electrical events that ultimately trigger nerve impulses. These impulses are sent to various visual centres of the brain through the fibres of the optic nerve."* (2)

To understand it better, William Bialek team (1) made an experiment with a vertebrate retina from a salamander (retina from the vertebrates are similar). They stimulated the retina with naturalistic gray scale movies of fish swimming in a tank while recording from 100 to 200 retinal ganglion cells. From the raw data they identified spikes from 160 neurons whose activity passed their quality check.

The aim of this experiment was to see how well the Ising model fits the raw data. Translating spikes from the different neurons to an Ising model consist of three steps: I. discretize a signal, II. estimate measures from data, III. fit the model.

I. First of all define binary variables σ_i.

The experiment collected a total of $\approx 2 \cdot 10^6$ spikes and time was discretized in bins of duration Δt=20 ms. With that, they created binary variables.

In the interval Δt=20 ms:

$$\sigma_i \leftarrow \begin{cases} +1 & \text{if the neuron emitted at least one spike ,} \\ -1 & \text{otherwise} \end{cases}$$

II. Second having the variables computed, it can be estimated the mean probability of each neuron generating a spike $< \sigma_i >$ and the correlation between spiking in pairs of neurons $< \sigma_i \sigma_j >$.

III. And finally fit an Ising model with the energy and Boltzmann probability distribution. To do that they adjusted the coefficients J and h that matched better the estimated measures.

$$P(\sigma_i) = \frac{1}{Z} exp[-E(\sigma_i)]$$

$$E(\sigma_i) = -\sum_{i}^{N} h_i \sigma_i - \frac{1}{2} \sum_{i}^{N} J_{ij} \sigma_i \sigma_j$$

The resulting Ising model fitted the data statistics very well.

Additionally, as the Boltzmann distribution in physics is affected by temperature their idea was to change a so called 'effective temperature' and fit again. Doing this for a range of temperatures they estimated with a criticality marker (specific heat) in which regime of an Ising model was the system, the result was the retina was operating near the critical regime (criticality near the original temperature, T=1).

2.2 Unsupervised machine learning

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed (3). There are three types supervised, unsupervised and reinforcement learning. In this thesis we are using artificial neural networks trained under unsupervised learning.

Unsupervised learning is "*the task of inferring a function to describe hidden structure from unlabeled data a (e.g. modeling the probability distribution or finding interesting directions in the data such as with principal component analysis)*" (4).

The most natural way of doing unsupervised learning is to use generative models (5). A generative model is a model for randomly generating observable data values (6). For our analyses in the Results section we will use a generative model based on neural networks called restricted Boltzmann Machines.

Chapter 3

Methods

3.1 Artificial neural networks

An artificial neural network (ANN) is a collection of simple trainable units that collaborate together to compute a complex non linear function. They are compatible with supervised, unsupervised and reinforcement learning. It is loosely inspired on the brain, e.g. the input/output relation of the nodes is similar to what we think occurs in real neurons.

The main characteristic of an ANN is that they contain a set of adaptive weights that are tuned by a learning algorithm and that implements the capability of approximating non-linear functions of their inputs. (7)

Neural networks consists normally of three layers: input (or visible), output and hidden as seen in figure 3.1.

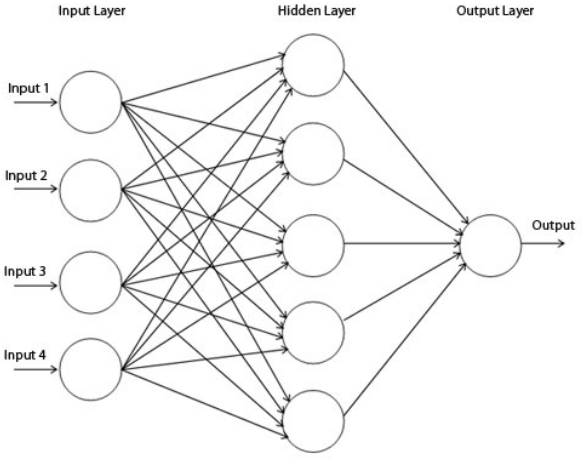

FIGURE 3.1: Example of neural network (19)

In our case we are interested in unsupervised learning because we want to create a generative model of the probability distribution measured in the retina. Therefore, we will use a special case

of unsupervised ANNs called restricted Boltzmann machine. We will explain two types of neural networks needed to understand the concept of restricted Boltzmann machine: the Boltzmann machine and Hopfield networks.

3.1.1 Hopfield networks

A Hopfield network *"is a form of artificial neural network popularized by John Hopfield in 1982. The units in Hopfield network only take two different values for their states ,that depends on whether the units exceeds their threshold or not"* (8). We can see a representation of Hopfield network in figure 3.2.

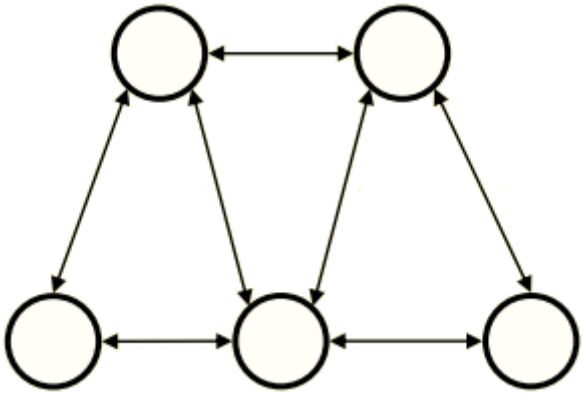

FIGURE 3.2: Diagram of Hopfield network

The connections in a Hopfield network (8) have the following restrictions.

$w_{ii} = 0 \; \forall \; i$

$w_{ij} = w_{ji} \; \forall \; i, j$

Updating one unit (node in the graph simulating the artificial neuron) in the Hopfield network (8) is performed using the following rule:

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \; b_i, \\ -1 & \text{otherwise.} \end{cases}$$

where:

w_{ij} is the strength of the connection weight from unit j to unit i.

s_j is the state of unit j.

b_i is the bias or threshold of the unit i.

Usually the updates in a Hopfield network are made in an asynchronous way: only one unit is updated at a time and its picked at random or using a predefined order.

It can be seen that neurons attract or repel each other just by looking at the weights (8). Consider the connection weight w_{ij} between two neurons i and j. If $w_{ij} > 0$, the updating rule implies that $s_j = 1 \quad (-1)$, the contribution of j in the weighted sum is positive(negative). Thus, s_i is pulled by j towards its value $s_j = 1 \quad (-1)$.

Hopfield networks (8) have a scalar value associated with each state of the network referred to as the "energy", E, where:

$E = -\sum_{i,j} w_{ij} s_i s_j - \sum_i b_i s_i$

This value is called the "energy" (8) *"because the definition ensures that when units are randomly chosen to update, the energy E will either lower in value or stay the same. Moreover, under repeated updating the network will eventually converge to a state which is a local minimum in the energy function. Thus, if a state is a local minimum in the energy function, it is a stable state for the network".*

Hopfield proposed that this net could be used as a memory network by training the weights in a way the input pattern (what the net should remember) would be an energy minimum (9).

The training process (8) *"involves lowering the energy of states that the network should "remember". This allows the network to serve as a content addressable memory system, that is, the network will converge to a "remembered" state if only part of the state is given. The network can be used to recover from a distorted input to the trained state that is most similar to that input. This is called associative memory because it recovers memories on the basis of similarity. If you train on many images it will have a lot of energy minima and it may converge to the wrong one".*

Hopfield networks implement two core ideas (10): find a local minimum of the energy function by using a network of symmetrically connected binary threshold units and the idea this energy minimum corresponds to some memories.

There is a different way of using the ability to find local minima (10), instead of using the net to store memories we can use it to construct interpretations of the sensory input or we can understand it as learning new features from the sensory input. For this we add hidden units to the Hopfield networks as shown in figure 3.3, the interpretation of the sensory input is represented by the states of the hidden units, the "badness" of the interpretation is represented by the energy.

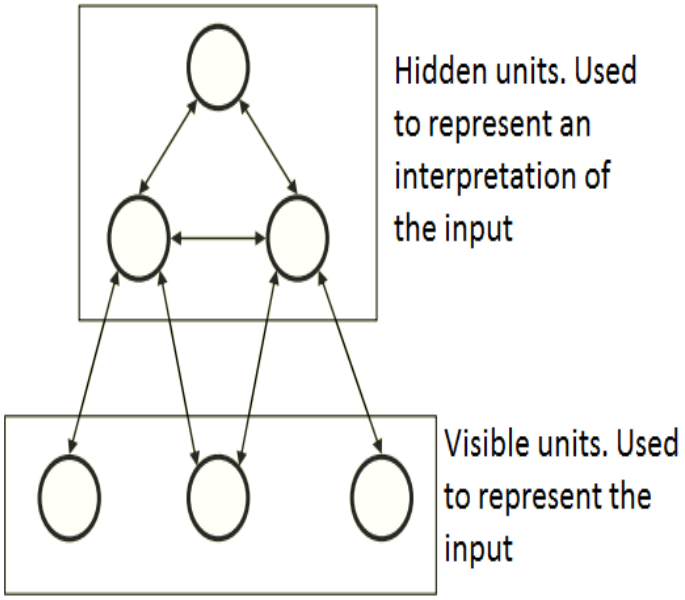

FIGURE 3.3: Adding hidden units allow us to learn representation of the sensory input (10)

This raises two difficult issues:

- We would like to have a mechanism to escape from local minima to get good interpretations.

- How do we learn the weights of the connections to the hidden units? We need a simple algorithm to adjust the weights to get sensible interpretation.

3.1.2 Boltzmann machine

A Hopfield network is deterministic, it always makes updates to the states that reduce the energy so it makes it impossible to escape from local minima. To overcome this limitation it is possible to use stochastic units (noise) to escape from poor minima.

A Boltzmann machine "*is a type of stochastic recurrent neural network and a Markov Random Field invented by Geoffrey Hinton and Terry Sejnowski in 1983 . Boltzmann machines can be seen as the stochastic, generative counterpart of Hopfield nets. They are named after the Boltzmann distribution in statistical mechanics, which is used in their sampling function*" (11). As with Hopfield networks we can use hidden units to learn new features from the input.

A Boltzmann machine (11), like a Hopfield network, is a network of units with an "energy" defined for the network. It also has binary units, but unlike Hopfield nets, Boltzmann machine units are stochastic. The global energy, E, in a Boltzmann machine is identical in form to that of a Hopfield network. If we include hidden units the energy is:

$$E(v,h) = -\left(\sum_{ivis} v_i\, b_i + \sum_{khid} h_k\, b_k + \sum_{i<j} v_i\, v_j\, w_{ij} + \sum_{i,k} v_i\, h_k\, w_{ik} + \sum_{k<l} h_k\, h_l\, w_{kl}\right)$$

Energy with configuration v on the visible units and h on the hidden units

The first (second) term is the contribution of the visible (hidden) and its bias to the energy. The third and fifth term are the weights between units in the same layer (visible with visible and hidden with hidden), and the fourth term is the connection between hidden and visible nodes. All units are connected.

The weights in a Boltzmann machine have the same restrictions as the Hopfield networks. The probabilities (using the Boltzmann distribution) of the joint distribution between hidden and visible states *P(v,h)* and configuration of states *P(v)* summing all over all possible hidden states are:

$$P(v,h) = \frac{\exp(-E(v,h))}{\sum_{U,g}\exp(-E(u,g))} = \frac{\exp(-E(v,h))}{Z}$$

$$P(v) = \frac{\sum_{h}\exp(-E(v,h))}{\sum_{U,g}\exp(-E(u,g))} = \frac{\sum_{h}\exp(-E(v,h))}{Z}$$

The term in the denominator is the partition function and if there are more than a few hidden units we cannot compute the partition function because it has a large number of terms.

We shall assume that the purpose of the learning is to create a good generative model of the set of training vectors in order to determine the parameters that maximize the probability of the observed data (12). To determine these parameters, we use the gradient descent on the log of the probability function. The result is:

$$\frac{\partial logp(v)}{\partial w_{ij}} =< s_i h_j >_{data} - < s_i s_j >_{model}$$

$$\Delta w_{ij} \propto < s_i s_j >_{data} - < s_i s_j >_{model}$$

Where s indicates any state hidden or visible.

The first term in the learning rule says raise the weights in the product of the activities the units have when you are presenting data , Donald Hebb suggested a similar firing method for neurons in 1949 (13):

" *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased*".

With only that rule the synapses will keep getting stronger and the weights will raise and the whole system will blow up. The second term in the equation prevents this blow up to occur (14).

The basic meaning of the learning algorithm is that we learn a model and we adapt it contrasting to the presented data.

The problem with Boltzmann machine is that the partition function is intractable so we cannot compute $< s_i s_j >_{model}$

3.1.3 Restricted Boltzmann machine

A restricted Boltzmann machine (RBM) (figure 3.4) is a Boltzmann machine with no connections between hidden units. Only connections between the visible and the hidden layers are assumed.

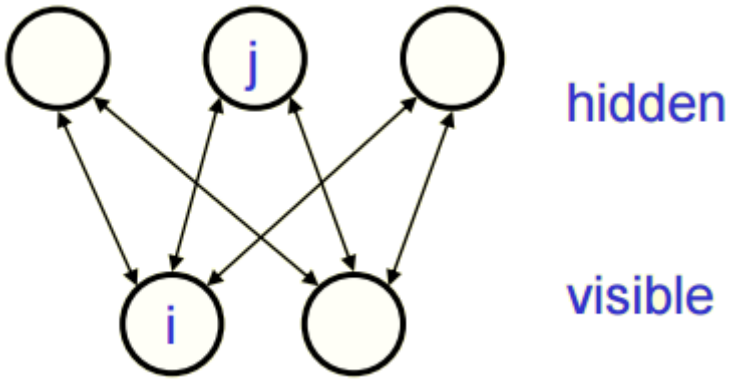

FIGURE 3.4: Representation of a restricted Boltzmann machine, the limited connections helps the computation problem (14)

Because we do not have connections between hidden units and visible units the terms of the energy involving correlations in the hidden and visible layer disappears:

$$E(v,h) = -\left(\sum_{i=1}^{V} v_i\, b_i + \sum_{k=1}^{H} h_k\, b_k + \sum_{i,k} v_i\, h_k\, w_{ik}\right)$$

Like before we use a Boltzmann distribution to express probabilities:

$$P(v,h) = \frac{\exp(-E(v,h))}{\sum_{U,g}\exp(-E(u,g))} = \frac{\exp(-E(v,h))}{Z}$$

$$P(v) = \frac{\sum_h \exp(-E(v,h))}{\sum_{U,g}\exp(-E(u,g))} = \frac{\sum_h \exp(-E(v,h))}{Z}$$

As high energy are associated with low probabilities if the bias terms are negative we will express a preference for the binary values being 0 and if the bias are positive we would have preference for the units being 1.

Still Z is intractable, consequently we could only use an approximation. However there are other types of inferences that are tractable in a RBM (because there are no connections in the same layer). In this case conditional inference takes the form:

$$p(h_k = 1|v) = \frac{1}{1 + exp(-(b_k + \sum_i v_i w_{ik}))}$$

$$p(v_i = 1|h) = \frac{1}{1 + exp(-(b_i + \sum_k h_k w_{ik}))}$$

3.1.3.1 RBM with Gaussian distribution

A RBM in which the visible units can only take binary values is very inconvenient for modeling real-valued data such as pixel intensities (15). To better model real-valued data, we add a negative quadratic term for the visible layer in the energy function:

$$E(v, h) = -\left(\sum_{k=1}^{H} h_k b_k + \sum_{i,k} v_i h_k w_{ik} - \sum_{i=1}^{V} \frac{(v_i - b_i)^2}{2}\right)$$

The only change is that the probabilities for the visible units are now a Gaussian function with mean given by the parameters $(h_k, v_i, b_i, b_k, w_{ik})$ and an identity covariance matrix, while the hidden units remain the same binary units.

$$P(v|h) = \prod_{i=1}^{V} \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}(v_i - b_i - \sum_{j=1}^{H} h_j w_{ij})^2)$$

3.1.3.2 Training

To train the RBM we will perform gradient descent on the log of the probability function:

$$\frac{\partial logp(v)}{\partial w_{ij}} = < s_i s_j > - < s_i s_j >_{model}$$

This time we can change s_i for v_i and s_j for h_j because there are no connection between the same type.

The main idea behind training is to start with a training vector on the visible units. Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel until infinity (with this we have a model of the presented data), then adapt the weights (change the model) to make them similar to the presented data. The change on weights will take the form:

$$\Delta w_{ij} = \epsilon(< v_i h_j >_0 - < v_i h_j >_\infty)$$

The problem of this algorithm is that we have to run it for a long time until it reaches the equilibrium. A much faster approach was invented by Hinton (16). Instead of doing the process infinite times until it reaches equilibrium, we do it only a set of times: starting with the training vector update all the hidden units in parallel then update all the visible units to get a "reconstruction" and then update the hidden units again, this is called contrastive divergence. We can repeat the process successive times to get better precision.

The change in a weight is then given by:

$$\Delta w_{ij} = \epsilon(< v_i h_j >_0 - < v_i h_j >_{recon})$$

With this the computation time is much lower, we update the hidden and the visible layers separated using the conditional probabilities $p(v|h)$ $p(h|v)$ and then we can compute $< v_i h_j >_{recon}$.

3.1.3.3 Filters and reconstruction error

The reconstruction error is the square difference $(v - v')^2$ between the data vector ,v, and the vector produced by sampling the hidden units given the values of visible unit and then sampling the visible units given hidden ones (reconstruction of the presented data). A good training should result in a decreasing error after every epoch.

To visualize the connections and what kind of features the hidden units represent in the case of images we use the plot called filters. We take each hidden unit and plot a gray scale image of the intensity of the weights with all the pixels in the input image. A white pixel means that the weight is positive so it increases the probability of $P(h = 1)$, a black pixel means that the weight is negative so it decreases the probability of $P(h = 1)$ and a grey pixel means that the weight is zero thus it will not impact on the probability.

A good generative model of natural images extracts useful high-level features, such as the locations and orientations of contours. Such a model would have some connection to a physical realities of human vision , as the visual cortex contains neurons that detect high-level features such as edges (15). In edges the image brightness changes sharply. To detect them we need to see a border between a block of white and a block of black. In the figure 3.5 we can see an example of a noisy filter which does not learn high level features and an edge detector in which we can see a difference between white and black.

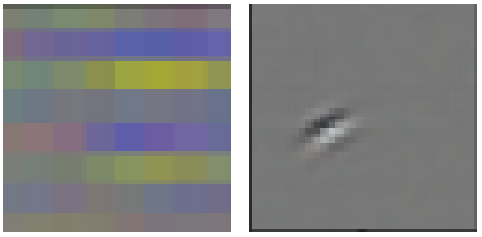

FIGURE 3.5: Examples of noisy filter and edge detector filter (15).

3.1.3.4 Sampling

Having trained and visualized the restricted Boltzmann machine we will need samples from the energies to evaluate the thermodynamics. With the weights fixed from the training we start from a random visible state. Then we sample the hidden state from the conditional probability of hidden given the visible one and then we sample again the visible state from the conditional probability of visible given the hidden one. For each iteration we compute the energy and we calculate the variance in energy, $\delta\epsilon$, between iterations.

For the sampling we use a parameter called burning time to start computing energies after a number of samplings in which the system is more stable.

With this now we can calculate a response function which distinguishes in which regime is our system, called specific heat:

$$C = \frac{< (\delta\epsilon)^2 >}{T^2}$$

In order to show the statistics of our system we need to calculate the C at different temperatures. To do that we follow Bialek idea (1) and create the effect of temperature in the Boltzmann distribution by dividing the energy by a factor T. In the next steps we deduce the effect of the temperature when sampling from the network via calculating the conditional probabilities once the effective temperature has been modified.

Starting with the conditional probability of hidden given visible:

$$P(h|v) = \frac{p(v,h)}{\sum_{h'\in[0,1]} p(v,h')} = \frac{exp((-\cancel{\sum_{i=1}^{V} \frac{(v_i - b_i)^2}{2}} + \sum_{k=1}^{H} h_k b_k + \sum_{i,k} v_i h_k\, w_{ik})/T))/\mathcal{Z}}{\sum_{h'\in[0,1]} exp((-\cancel{\sum_{i=1}^{V} \frac{(v_i - b_i)^2}{2}} + \sum_{k=1}^{H} h'_k b_k + \sum_i \sum_{i,k}^{V,H} v_i h'_k w_{ik})/T)/\mathcal{Z}}$$

$$= \frac{\prod_{k=1}^{H} exp((h_k b_k + \sum_{i=1}^{V} v_i h_k w_{ik})/T)}{\sum_{h'\in[0,1]} \prod_{k=1}^{H} exp((h'_k b_k + \sum_{i=1}^{V} v_i h'_k w_{ik})/T))}$$

$$= \frac{\prod_{k=1}^{H} exp((h_k b_k + \sum_{i=1}^{V} v_i h_k w_{ik})/T)}{(\sum_{h' \in [0,1]} exp((h'_k b_k + \sum_{i=1}^{V} v_i h'_k w_{ik})/T)...(\sum_{h' \in [0,1]} exp((h'_k b_k + \sum_{i=1}^{V} v_i h'_k w_{ik})/T)}$$

$$= \frac{\prod_{k=1}^{H} exp((h_k b_k + \sum_{i=1}^{V} v_i h_k w_{ik})/T)}{\prod_{k=1}^{H} \sum_{h' \in [0,1]} exp((h'_k b_k + \sum_{i=1}^{V} v_i h'_k w_{ik})/T)}$$

In this step we substitute all the values allowed of $h' \in [0, 1]$

$$= \prod_{k=1}^{H} \frac{exp((h_k b_k + \sum_{i=1}^{V} v_i h_k w_{ik})/T)}{1 + exp((b_k + \sum_{i=1}^{V} v_i w_{ik})/T)}$$

$$= \prod_{k} p(h_k|x)$$

We have the probability for the full network, if we want the probability of one unit ,k, being: $h_k = 1$

$$p(h_k = 1|v) = \frac{exp((b_k + \sum_{i=1}^{V} v_i w_{ik})/T)}{1 + exp((b_k + \sum_{i=1}^{V} v_i w_{ik})/T)}$$

$$p(h_k = 1|v) = \frac{1}{1 + exp(-(b_k + \sum_{i=1}^{V} v_i w_{ij})/T)}$$

As we are using the Gaussian distribution we will have for the visible part a continuous space. We will have integrals instead of discrete sums for the visible layer.

$$P(v|h) = \frac{\frac{e^{-(E(v,h)/(T))}}{\cancel{Z}}}{\int_u \frac{e^{-(E(v,h)/(T))}}{\cancel{Z}} du} = \frac{e^{(\sum_{k=1}^{H} h_k\, b_k + \sum_{i,k} v_i\, h_k\, w_{ik} - \sum_{i=1}^{V} \frac{(v_i - b_i)^2}{2})/T}}{\int_u e^{(\sum_{k=1}^{H} h_k\, b_k + \sum_{i,k} u_i\, h_k\, w_{ik} - \sum_{i=1}^{V} \frac{(u_i - b_i)^2}{2})/T} du}$$

$$= \frac{\cancel{e^{(\sum_{k=1}^{H} h_k\, b_k)/T}} \prod_{i=1}^{V} e^{(\sum_{k=1}^{H} v_i\, h_k\, w_{ik} - \frac{(v_i - b_i)^2}{2})/T}}{\cancel{e^{(\sum_{k=1}^{H} h_k b_k)/T}} \int_u \prod_{i=1}^{V} e^{(\sum_{k=1}^{H} u_i\, h_k\, w_{ik} - \frac{(u_i - b_i)^2}{2})/T} du}$$

First of all we will compute the integral in the denominator using the well known result and knowing that as a continuum $u \in [-\infty, \infty]$.

$$\int_{-\infty}^{\infty} e^{-fx^2 + gx + h} dx = (\frac{\pi}{f})^{1/2} exp(\frac{g^2}{4f} + h)$$

$$f = \frac{1}{2T} \qquad h = \frac{-(b_i)^2}{2T} \qquad g = \frac{b_i}{T} + \frac{1}{T}\sum_{k=1}^{H} h_k w_{ik}$$

$$\int_u e^{(\sum_{k=1}^{H} u_i\, h_k\, w_{ik} + \frac{(u_i - b_i)^2}{2})} du = \sqrt{2\pi T} e^{\frac{1}{2T}((\sum_{k=1}^{H}\ h_k\, w_{ik})^2 + 2b_i(\sum_{k=1}^{H}\ h_k\, w_{ik}))}$$

$$P(v|h) = \frac{\prod_{i=1}^{V} e^{(\sum_{k=1}^{H} v_i\, h_k\, w_{ik} - \frac{(v_i - b_i)^2}{2})/T}}{\prod_{i=1}^{V} \sqrt{2\pi T} e^{\frac{1}{T}(\frac{1}{2}(\sum_{k=1}^{H} h_k w_{ik})^2 + b_i(\sum_{k=1}^{H}\ h_k\, w_{ik}))}}$$

If we move up the exponential in the denominator:

$$P(v|h) = \prod_{i=1}^{V} \frac{1}{\sqrt{2\pi T}} e^{\frac{1}{2T}(-(v_i - b_i)^2 - (\sum_{k=1}^{H} h_k w_{ik})^2 + 2(v_i - b_i)(\sum_{k=1}^{H} v_i h_k w_{ik}))}$$

$$P(v|h) = \prod_{i=1}^{V} \frac{1}{\sqrt{2\pi T}} e^{-\frac{1}{2T}(v_i - b_i - \sum_{k=1}^{H} h_k w_{ik})^2}$$

3.2 Signature of criticality

The fundamental variables of thermodynamics are the energy, temperature and entropy but for a network of artificial neurons the last two are, in principle, meaningless (1).

In statistical mechanics all thermodynamic quantities are derivable from the Boltzmann distribution. The probability that the system will be found in a particular state:

$$P\{\sigma_i\} = \frac{1}{Z} exp[\frac{-E\{\sigma_i\}}{T}]$$

$$Z = \sum_s exp(\frac{-E_s}{k_B T})$$

To understand the thermodynamics in our system we will use the concept of energy and we will modify the probability of the Boltzmann function to make the effect of the "effective" temperature.

One interesting quantity in statistical mechanics systems is the specific heat, which is connected to the variance in energies of our system (1):

$$C = N\frac{< (\delta\epsilon)^2 >}{k_B T^2}$$

In some physical systems there are some points (called critical points) at which the behaviour of whole system might change its statistical properties .*"For some systems and phase transitions these effects can be noted from the divergence of the correlation length or the slow down of the dynamics. Critical phenomena include scaling relations among different quantities, power-law divergences of some quantities (such as the magnetic susceptibility in the ferromagnetic phase transition) described by critical exponents, universality, fractal behaviour"* (17)

At the critical regime (17) the system is highly sensitive to external perturbations, and the heat capacity which represents the thermal response function, diverges at the criticality point. For example, we can see the response of specific heat to the temperature in the figure 3.6.

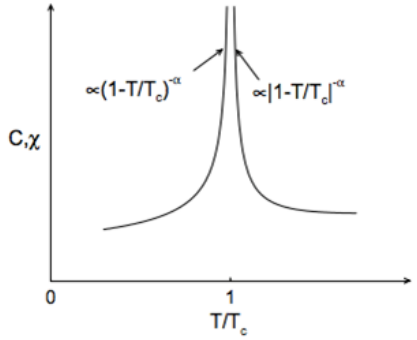

FIGURE 3.6: Response of specifc heat to temperature

The heat capacity can be used to detect criticality in other systems where an energy function and temperature are well defined.

The thermodynamics of natural images have been long studied (18). We will use the ideas mentioned before to study the properties of natural images and for images corresponding to noise.

3.3 Datasets

As in the study of Bialek (1) we used natural images to drive the dynamics of the neurons. The input to our neural network models consist of natural images with each pixel corresponding to the input to each node in the input layer of the network.

For most of the analysis the training set consisted of 200 481x321 natural images from Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500). From these images we create patches of size 32x32 so the input consists of 1024 pixels. We set those RGB 32x32 patches to gray scale. To control for the effect of other types of inputs we have also used images corresponding to random noise as an input.

Chapter 4

Results

As mentioned in methods, our objective with the RBM training is the search for a good generative model of natural images and in addition we seek a model that is capable of extracting useful features from images, such as the locations and orientations of contours. Once a good generative model has been fitted to explain the natural images input, then we will test in which phase our RBM operates (order, disorder, or near criticality).

As to comprehend if the statistics of the input affect our results we will work with the original data set and random noise.

4.1 Natural images

In this section we use a set of naturalistic images to feed the RBM. The goal is to test whether or not natural images driving our generative neural network model present critical signatures.

4.1.1 Initial attempts

We initially trained a RBM with 256 hidden units and training set of 25000 (patches of 32x32 pixels). As we can see in the figure 4.1 the reconstruction error gets smaller values after each epoch so we infer that it is trained well. However we cannot know (with only this training) if the last epoch reconstruction error is good enough.

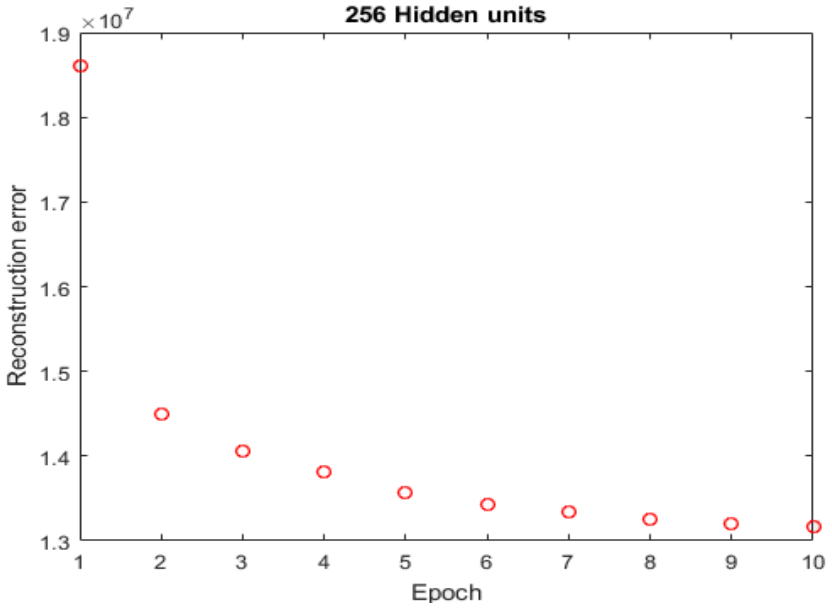

FIGURE 4.1: Reconstruction error for a RBM with 256 hidden units and training set of 25000.

Despite the good convergence of the reconstruction error this model developed a lot of filters like the ones shown in figure 4.2. Most of them are random noise and some of them learn localized 'features', point-like identity functions. Our objective is high level features such as edge detectors, a difference between black and white in one place.

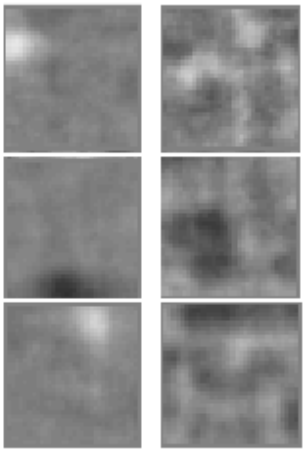

FIGURE 4.2: Example of filters learned by an RBM with 256 hidden units and training set of 25000: on the left side there is localization ,on the right side there is randomness

Although having obtained this model we sampled from it to compute the energy variances and plotted one marker of criticality, specific heat versus temperature in figure 4.3.

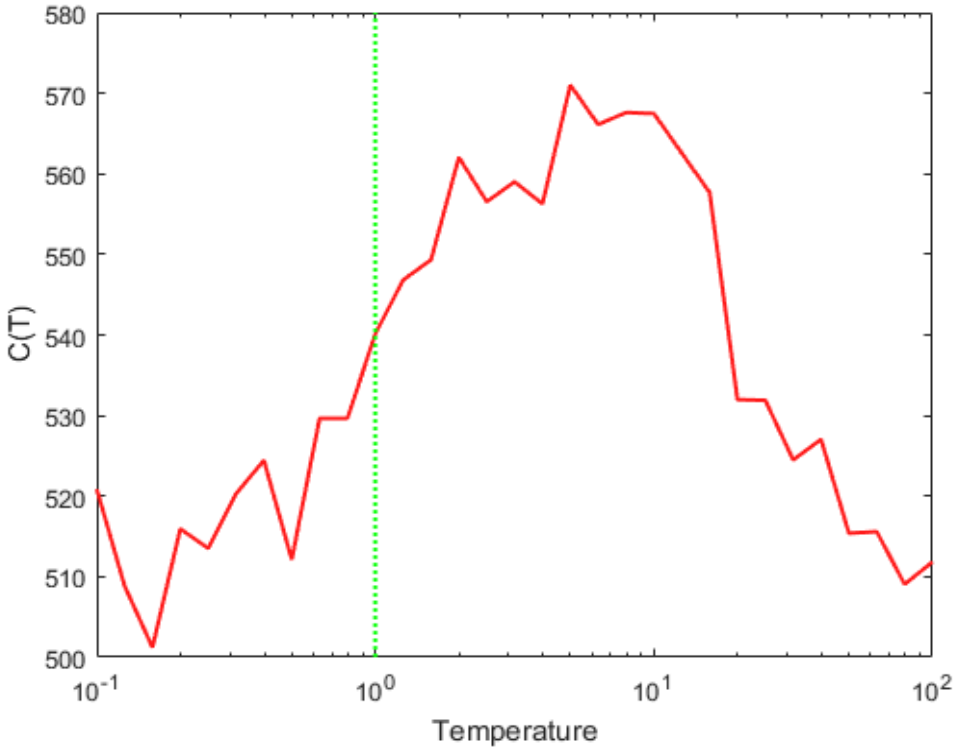

FIGURE 4.3: Plot of specific heat vs temperature for 256 hidden units and a training set of 25000. We set the original fit (T=1,without dividing by an effective temperature) with a green label.

The peak of the specific heat is located near T=10 and the specific heat variance divided by the temperature is not smooth. We clearly have not the similar results as observed in (1). This led to the question of whether a RBM is good enough to model the retina?

4.1.2 Making the RBM to work

First, we need to make the filters learn meaningful features and if possible decrease the reconstruction error. In order to do that we tried to change the most basic thing: the training set. Originally we only had 200 images cut in 32x32 patches and it was not possible to find new data sets. To overcome this limitation we took overlapping patches of the images. With this we increased our training set to 10^5.

Increasing training set allowed us to reduce the reconstruction error starting in the first epoch as seen in figure 4.4.

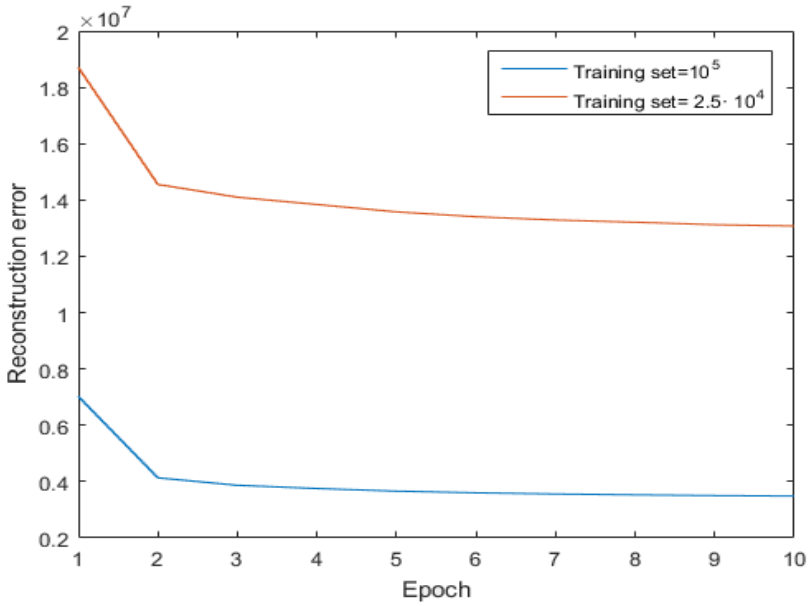

FIGURE 4.4: Plot of reconstruction error for 100000 and 25000 training set. Increasing the training set improves the reconstruction error

We can see the improvement in the figure 4.5, most noisy filters started to disappear and the result was having most of the filters as point-like identity features.

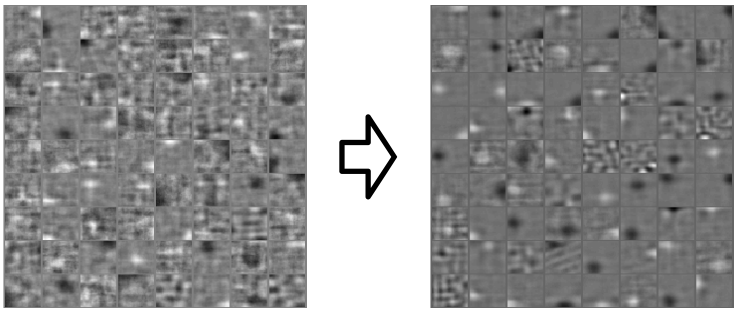

FIGURE 4.5: Filters improvement when increasing the training set

As has been pointed out from our results, having a big training set is necessary to make the RBM learn useful features. However this is not enough, we need a way to make the localized point-like features something like edge detectors. The idea was to give the RBM more nodes so it could learn a more complex model. Therefore, we tried to monitor its behaviour with different number of hidden neurons. We did this for the large training set containing 10^5 image patches.

To know the effect that the hidden nodes produce on the error we plotted the last Epoch reconstruction error vs number of hidden nodes in figure 4.6.

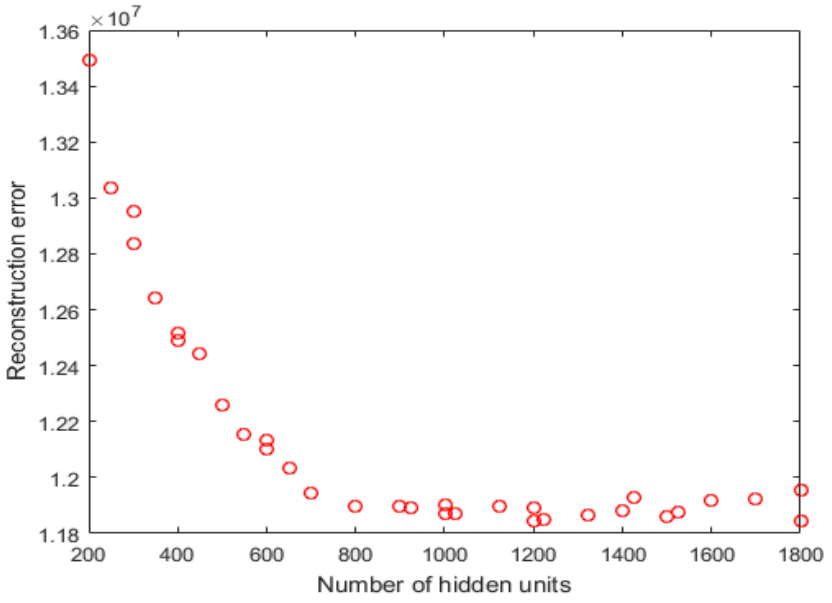

FIGURE 4.6: Plot of last epoch reconstruction error in function of the number of hidden nodes. Hidden nodes decrease reconstruction error as it can learn a better model of the sensory data

The reconstruction error in the final epoch decreases while increasing the number of hidden units in the RBM until it reaches a plateau. We infer from our results that the optimal value (at least from the reconstruction error perspective) is 800.

We can see how the filters change with more hidden nodes in figure 4.7.

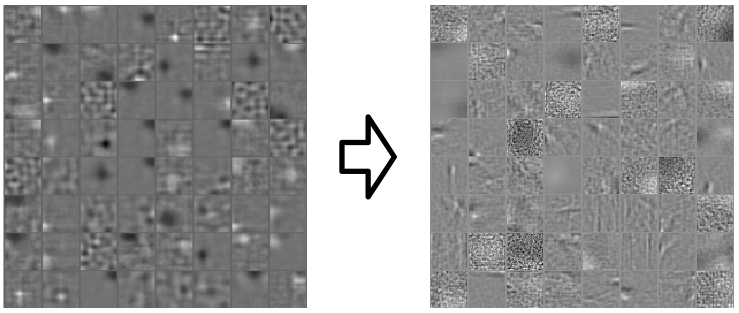

FIGURE 4.7: Filters improvement when increasing the hidden nodes. 300 hidden units on the left, 1500 on the right

In this plot we see the filters for 300 hidden units and for 1500 hidden units. Increasing the hidden nodes converts the point like filters into edge detectors, and the quality of those edge detectors improve for all experiments we have tried (we are only able to try until 1900 hidden nodes due to the specification of our system). However increasing hidden units (with the training set constant) also produces more noisy filters.

Finally as seen in figure 4.8 we plot the specific heat for a couple of RBMs.

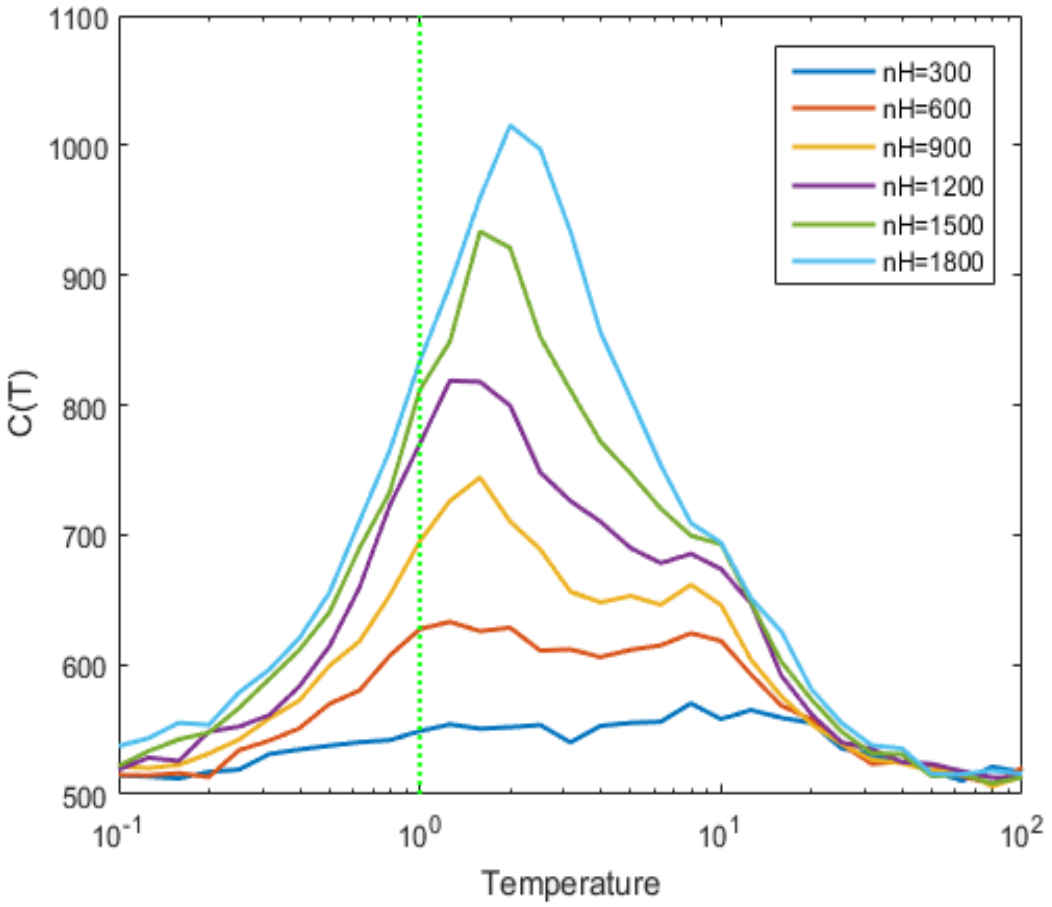

FIGURE 4.8: Peak of criticality for different size of RBM

Initially increasing the hidden units makes the criticality signature peak to occur near $T = 1$ but then it starts to move away from it. Our guess is that increasing the number of hidden nodes, means a lot more parameters so the RBM will require more training set to adjust those parameters. At one point the training set starts to not to be enough to make the RBM work. This can be easily proved by training an RBM with 50000 and 100000 patches and see that the specific heat peak departs more from $T = 1$.

With this information we can conclude that a suitable number of hidden nodes with our training set will be around 1200-1500.

4.1.3 Final training

Taking this into account we set the RBM hidden units to 1500 and use the training set of 10^5.

First of all we plot the reconstruction error per epoch in figure 4.9 to verify it converges.

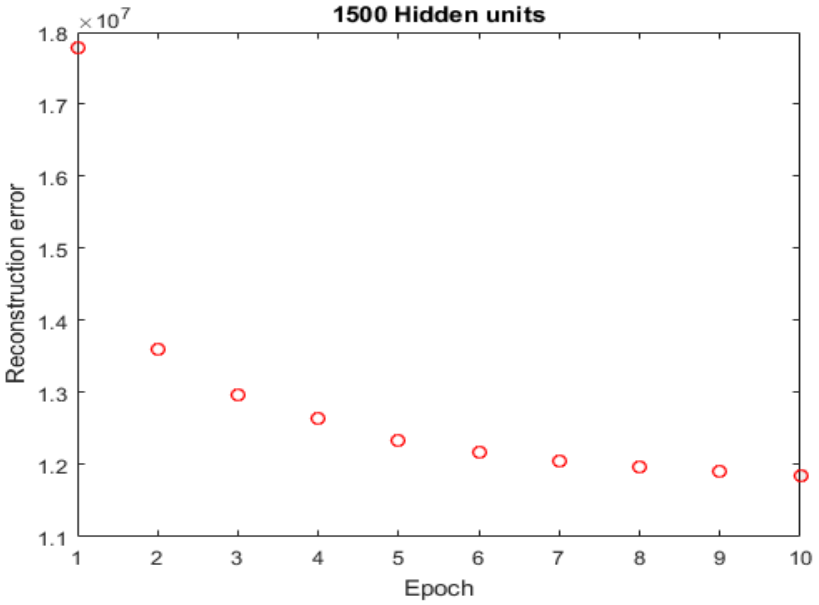

FIGURE 4.9: Reconstruction error per epoch for 1500 hidden units and 10^5 training set

Moreover, the point-like filters, as seen in the evolution figure, transformed into edge detectors as is shown in the figure 4.10. Our RBM is now extracting better features and acting more like the retina and first stages of visual processing.

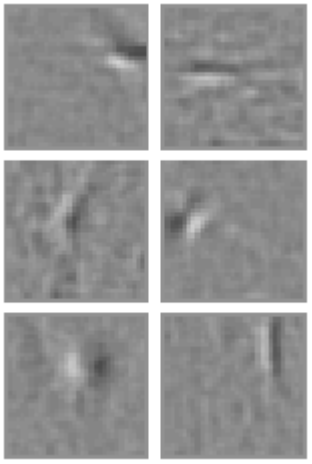

FIGURE 4.10: Examples of filters resulted from our RBM

As the reconstruction error gets lower we expect to get a similar picture if we feed the RBM with an image and go one time to hidden and then go back to the visible layer. To test this we feed the RBM with the picture of a bear and plot it after computing the hidden node activations and from them the visible nodes. The result can be seen in figure 4.11.

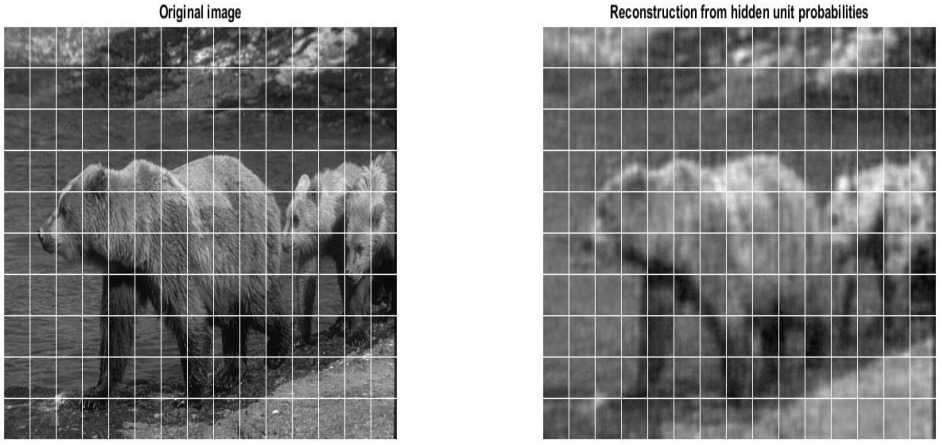

FIGURE 4.11: Reconstruction test of a bear image after one iteration of sampling in the RBM

As we can see we get the same picture a little bit modified as we expected.

To understand a little bit better our RBM statistics we plot the weights in figure 4.12.

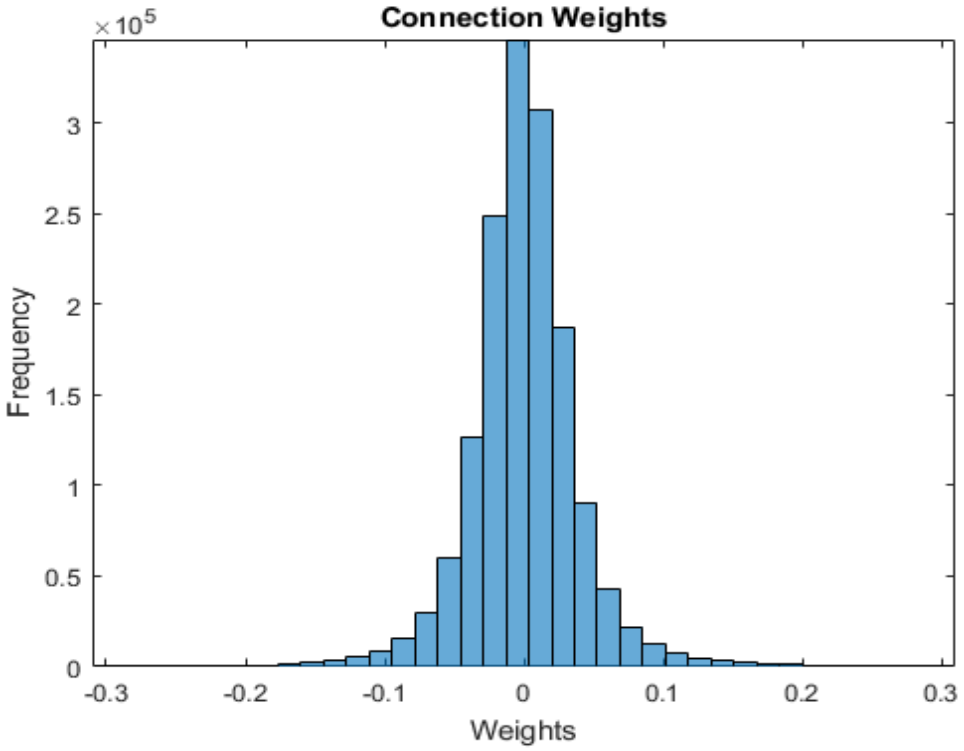

FIGURE 4.12: Histogram of the connection weights of our RBM. We can see that most are 0, we have seen this already in the filters where most of the them are gray areas (W=0)

Finally with this well trained RBM we monitor how our criticality signature (specific heat vs effective temperature) behaves in figure 4.13.

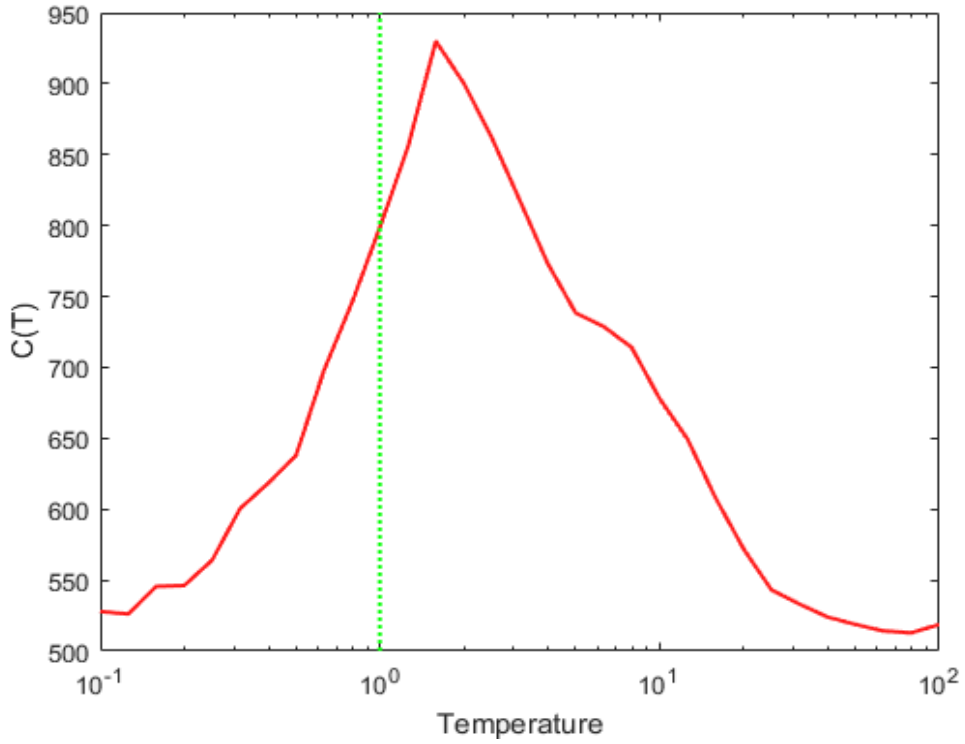

FIGURE 4.13: Criticality peak for 1500 hidden units and a training set of 100.000. It peaks at T = 2.

The peak of specific heat occurs somewhere around $T = 2$, not too far from the original fitting ($T = 1$). This shows that in response to natural images our originally trained RBM operates near the critical regime and on the ordered side of the transition. Now we can see the RBM as a basic model for understanding some statistical features of retinal neurons, how it operates by extracting high level features, and possibly operating in a regime near criticality in response to natural images.

4.2 Artificial images

In this section our objective is to test if the results hold with random noise images or they depend on the statistic of the input. For that we take our original data set and randomize the pixels position in every image. Using the same hidden nodes and training set as with our final training, we trained a new RBM. All filters are noisy (the RBM cannot learn anything more complex from the noise patterns) and the reconstruction error is high. We plot our criticality marker in figure 4.14.

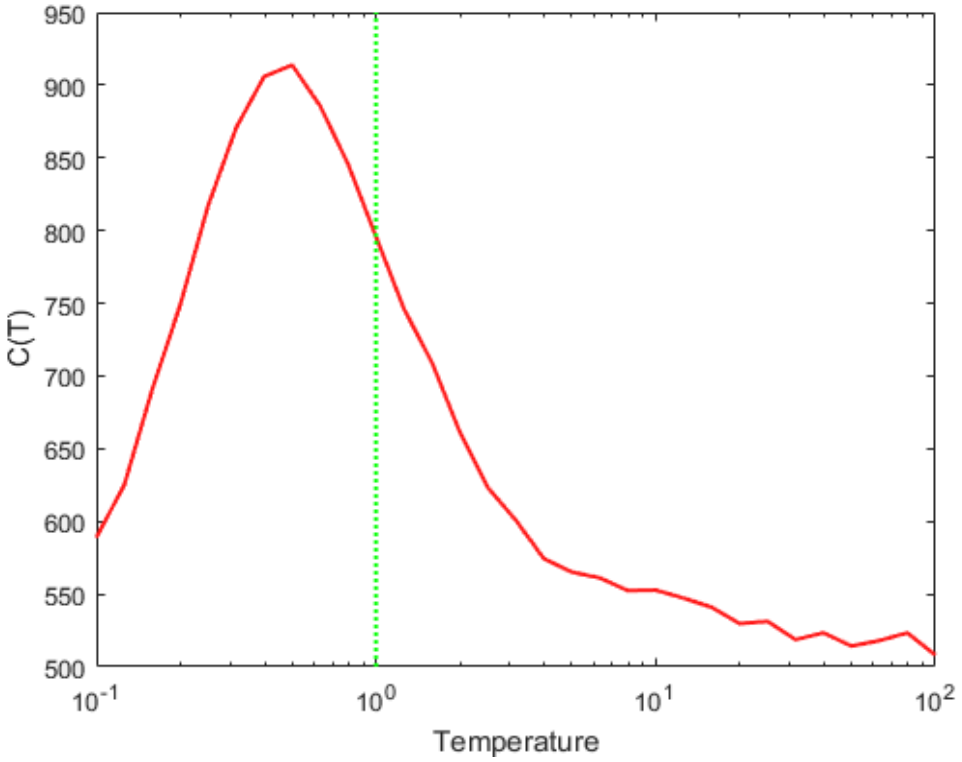

FIGURE 4.14: Peak of criticality for random noise images

In this case the peak of specific heat occurs at the left of our original temperature ($T = 1$). Could this mean that we are in the disorder part of an Ising/RBM model?

Chapter 5

Discussion

5.1 Comparasion with literature

For natural images we have seen that our model with the RBM operates near the criticality regime on the ordered phase. Furthermore our results indicate that under randomized input our generative neural network operates in the high disorder phase. These results occur because the learning rule of the RBM is able to internalize the input statistics into its neural weights and dynamics.

We have also observed that the RBM extract high level features as seen on the retina and next visual processing stages. For that we think it might be a useful model to understand the statistical behavior of the retina under different inputs. Comparing with the results from (1) in figure 5.1., we can see the real fitted data and our generative model display have a qualitatively similar behaviour.

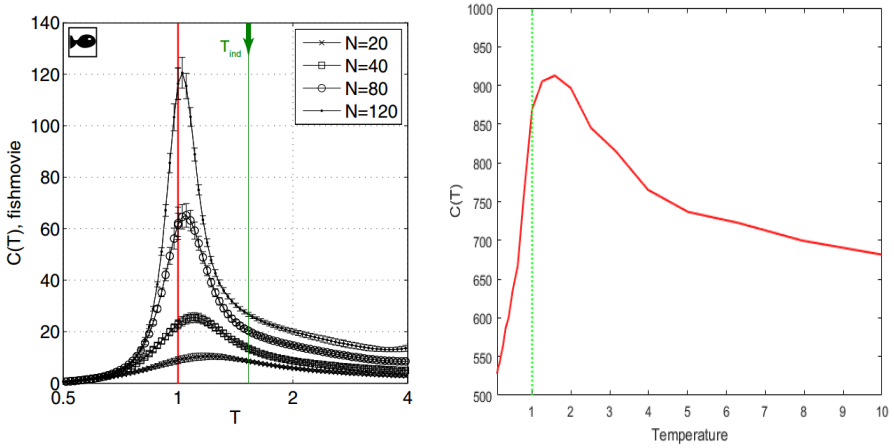

FIGURE 5.1: Comparison of results from Bialek work (1) (salamander retina stimulated with gray scale movie of fish) and our RBM. In their figure they made the fit for a variable number of neurons, N.

5.2 Limitations

We have trained our neural model from patches of only 200 images from BSD 500. As we have seen increasing the training set should have decreased the number of noisy filters.

We did the analyses on Matlab with a i5 3.2 GHz processor with 6 GB of RAM. We should migrate our system to a high performance cluster to handle higher training sets and higher number of nodes.

5.3 Future work

First of all we need to increase the training set in order to reduce the noisy filters that we still have. To do that we plan to use new and larger data sets.

We also plan to study different types of input, including different types of naturalistic inputs (including sound) to test the effects of input in driving the networks toward criticality.

One good possibility is to try a different kind of images, artificial images of textures where we hypothesize that after learning, the RBM will operate in the high order phase. More generally, we will test our results with inputs with well controlled statistics (e.g. by gradually varying the slope of the power spectrum of the input via filtering).

Deep neural networks are getting good results lately and is believed that the image processing of our brain have different layers applying the same kind of algorithm to process images. In order to better mimic that strategy we would also like to increase the depth (number of hidden layers) of the RBMs. This kind of models are called Deep belief networks. Finally lately has resurged an interesting kind of RBM called three-way RBM that hass explicit terms to model the correlation between input pixels. This model and deep belief networks will be subject of future work to inspect a more realistic model of the retina and to assess how well we can explain its remarkable statistical properties with simple models.

Chapter 6

References

(1) Tkaik, Gaper, et al. "Thermodynamics and signatures of criticality in a network of neurons." Proceedings of the National Academy of Sciences 112.37 (2015): 11508-11513.

(2) "Retina" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 20 Jun. 2016. Web. 21 Jun. 2016.

(3) Simon, Phil. Too Big to Ignore: The Business Case for Big Data. Vol. 72. John Wiley Sons, 2013.

(4) "Unsupervised learning" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. Inc. 21 Jun. 2016. Web. 21 Jun. 2016.

(5) Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

(6) "Generative model" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 13 Mar. 2016. Web. 21 Jun. 2016.

(7) "Artificial neural network" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 19 Jun. 2016. Web. 21 Jun. 2016.

(8) "Hopfield networks" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 30 Apr. 2016. Web. 21 Jun. 2016.

(9) Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences 79.8 (1982): 2554-2558.

(10) Hinton, Geoffey. Neural Networks for Machine Learning. Lecture 11. 2012.

(11) "Boltzmann Machine" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 26 May 2016. Web. 21 Jun. 2016.

(12) Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

(13) Hebb, Donald Olding. The organization of behavior: A neuropsychological approach. John Wiley Sons, 1949.

(14) Hinton, Geoffey. Neural Networks for Machine Learning. Lecture 12. 2012.

(15) Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

(16) Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.

(17) "Critical phenomena" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 11 May 2013. Web. 21 Jun. 2016.

(18) Stephens, Greg J., et al. "Statistical thermodynamics of natural images." Physical review letters 110.1 (2013): 018701.

(19) "Artificial neural network" America pink. 21 Jun. 2016