Contents

1 Introduction

In recent years we have assisted to the worldwide spread of AI and, particularly, Machine Learning. The main goal of this discipline is to create and produce intelligent computational devices that can assist humans in performing every day tasks, or even exceed human capabilities, enabling them to perform a wide range of tasks efficiently and effectively [1] [2]. AI applications can be found everywhere: from the very famous ChatGPT, an AI-driven language model able to generate human-like text based on a prompt [3], to platforms like Spotify or Netflix which employ AI tools for internal purposes like building sophisticated recommendation systems, to companies which are more and more moving towards this field in order to automatize monotonous tasks, enhancing efficiency and productivity. Modern architectures, like Multi-Layer Perceptrons, Convolutional Neural Networks and Transformers, were built upon foundational units, originally invented to be computational copies of the most powerful and, in many ways, still mysterious tool humans have: the brain.

In the following sections I will introduce these building blocks theoretically, on a slightly different perspective, explain strengths and limitations and present many variations that have been introduced throughout the years. All these models, despite seeming very different in their formulation, have a common denominator: they have basic units that interact between each other and with themselves, like networks of neurons do through synapses. Moreover, through the analysis of their parameters, we can observe the interplay between chaotic behaviour and convergence to some *attractors*: this mechanism stands at the very core of *associative memory*, an ability so natural for human beings and yet so intricate to grasp fully and replicate computationally.

Other than this mathematical and computational analysis, I will keep moving in the direction of trying to close the gap between Machine Learning and Biology, even more than what the above mentioned interpretation already does. And why do so? The answer is really simple: when AI was born, the very first inspiration was exactly the neuron, with the goal of imitating the human brain. After seeing the potential of these computational devices, the focus have been more and more shifted towards methods which could make them more powerful, including a vast number of parameters that now can run in the order

of $O(N^2)$, where N is the number of *neurons* in each network *layer*. Thus, the computational power modern AI require is huge, demanding an amount of energy which would be unbearable for the human brain. We started from the brain, but we might have outdone the job: if we *turned the power off*, maybe with the aim of being more environmentally friendly, all these devices would very likely stop working. But what if it was possible to create computational devices with the same power of deep CNN, yet using an amount of energy comparable to that needed from the human brain?

Starting from the most basic model of a neuron, I will move towards the first and most common discrete model of associative memory, the Hopfield model. Through the introduction of asymmetry and a thresholds, I will do a first step in the direction of making these models *more biological*. Then, I will switch context, considering $0-1$ units and differentiating between excitatory and inhibitory populations. Afterwards, I will do a brief excursus on continuous models, in particular firing-rate based models, and introduce properly the concept of synaptic plasticity. Lastly, I will go back to the first setting and, through the analysis of the actual learning ability of these networks, try to build a connection with modern AI architectures.

2 The Hopfield Model

In Deep Learning history the first model of an artificial neuron was proposed by McCulloch and Pitts [4], who intended to use it to implement logic gates. The model comprises a set of artificial neurons that can take binary values of 0 (silent) or 1 (active) and are updated through the following dynamics:

$$n_i^{t+1} = \Theta\left(\sum_j J_{ij} n_j^t - \mu_i\right) \tag{1}$$

where n_i^t denotes the state of neuron i at time t, the sum runs over all the neurons that are given in input to the neuron in question, weighted by what can be interpreted as the synaptic strength that connects the two neurons and finally Θ denotes the classical step function:

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

In words, a neuron is active at the next time step, if the synaptic input it receives is higher than some threshold value μ_i. Here there is no learning phase, since the weights and threshold values are chosen *a priori*.

This model was then reintroduced by Rosenblatt [5] in a different context: his objective was to reproduce a physical model of the human brain, able to process raw data and learn to interpret it through examples by adjusting its synapses ("weights"). The objective here falls within the framework of classification problems.

In this context, we can introduce the Hopfield model, a recurrent model of associative memory.

2.1 The model

The term *associative memory* refers to the ability of a system to recall and retrieve information based on associations or patterns [6]. For humans it is a fundamental aspect of cognitive functions, such as learning, reasoning, and decision-making, while in artificial intelligence, it is crucial for tasks such as pattern recognition and information retrieval. Therefore, understanding and modelling associative memory systems can lead to advancements in artificial intelligence, enabling machines to emulate cognitive abilities which are

proper of human beings. Nevertheless, modeling associative memory is not at all an easy task: not only is the brain itself a very complex biological tool to model, but short-term and long-term memory are also very elaborate mechanisms to even understand theoretically, let alone to replicate computationally. Despite being still far from this goal, the Hopfield network can be considered as a huge step towards it.

The Hopfield model [7] [8] is aimed at storing a set of P patterns ξ^μ, where $\mu = 1, ..., P$, in such a way that, when the network is presented with a new pattern, it is able to identify and produce the stored pattern that most closely matches the new input. Each pattern can be seen as a highly dimensional vector of N neurons which evolve through the following dynamics:

$$S_i^{t+1} = sgn\left(\sum_j J_{ij} S_j^t\right) \tag{2}$$

where we set a threshold of 0. As in the other models briefly specified above, the dynamics evolves in discrete time with neurons taking binary values (in this case $S_i = \{+1, -1\}$, denoting the *on* and *off* state of the neuron respectively) and *sgn* is the function:

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

The update can be carried out in two different ways:

- *Syncronously*: all neurons are updated at the same time at every time step
- *Asyncronously*: either one neuron is chosen randomly to update, or each neuron independently chooses to update itself at every time step

The starting random patterns are fundamental to build the couplings (synaptic efficacies) through what is called in the literature the *Hebb rule*:

$$J_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu \tag{3}$$

If this rule is put in place, it can be proved that, not only are the memorized patterns fixed point of the dynamics, but they are also stable. Formally, a fixed point of the system is such that the state of every neuron remains unchanged in the subsequent steps

of the dynamic's update (i.e. equation 2 would loose its time dependency and it would be imposed that it is satisfied $\forall i$). Instead, a fixed point is also stable if small *errors*, namely flipping $< \frac{1}{2}$ neurons, get automatically corrected. We can interpret an associative memory as a set of attractors with the respective basin of attraction: this mean that, when a new point is given to the system, it is able to carry it through the basin towards the attractor it initially stored (memorized).
This model is strictly related to the Sherrington and Kirkpatrick model of spin-glass [9]. Indeed, thanks to the Hopfield contribution, it is easy to introduce an energy function which takes the role of a Lyapunov function or Hamiltonian in statistical physics, and can be thought of as an energy landscape above all the possible configurations of the system:

$$H = -\frac{1}{2}\sum_{i,j} J_{ij} S_i S_j \tag{4}$$

In this framework, the attractors or *retrieval states* are local minima of the energy. This energy function perfectly recalls the one of spin-glass, where the J_{ij} are quenched independent random variables, which means that they are chosen independently at the beginning according to some probability distribution (typically a Gaussian with 0 mean and variance $\frac{1}{N}$), but then remain fixed over time.
When given a new point, the probability of making an error while trying to retrieve the "closest" pattern can guide us in the direction of understanding the storage capacity of the network, i.e. how many patterns can the network remember maintaining an acceptable performance. It was shown [10] [11] that, when $\frac{P}{N} =: \alpha \lesssim 0.14$ most of the patterns are recalled in a desirable way.

2.2 Limitations

The problem is that, also due to the rugged landscape, the energy attains local minima in the so called *spurious states* in a close neighbourhood of the attractors. These spurious state are sometimes retrieved in the minimization process but they are not the originally stored patterns and they can be of three different kinds:

- *Reversed states*, due to the symmetric nature of the system (see also below)
- *Mixed states*, which are linear combinations of an odd number of memorized states

- *Spin glass states*, which, instead, are completely uncorrelated with the initial patterns

Furthermore, there are also a number of other problem connected with this original and initial formulation of the Hopfield model: despite trying to mimic the function and behaviour of real neurons, the model is very different from a biological network. Indeed, it is entirely symmetric, which violates Dale's law by not distinguishing between excitatory and inhibitory synapses; moreover, if two neurons take values -1 the interaction between them is unrealistically reinforced.
Therefore, the first step is to loose the symmetry assumption and introduce some randomness which makes the model asymmetric.
Before doing so, I just want to mention that many of the models I will analyze, including the Hopfield network, have been solved analytically using statistical physics techniques which, in the majority of the cases, were firstly employed for solving the model of spin-glass (more on this in section 5). Nevertheless, I will always focus on their numerical analysis: I will start from a formal introduction, followed by personal implementations of the dynamics in Python to study behaviours and/or properties, and finally, I will display and briefly comment on the main results obtained.

3 Recurrent Asymmetric Neural Networks: the discrete case

I will now abandon the energy interpretation to return to analyze the system in a dynamical way, in order to be able to simulate the behaviour of the network against the variation of some parameters and analyze the phase transitions.

3.1 Model 1: auto-interaction coefficient

The first model I will study is a slight modification of the above mentioned classical Hopdfield model, which differs in the asymmetry and randomness of the synaptic efficacies (which is again a quenched disorder remaining fixed after initialization) and the introduction of an auto-interaction coefficient, from now on called J_D. In formula:

$$S_i = sgn\left(\sum_{j\neq i} J_{ij}S_j + J_D S_i\right) \tag{5}$$

Note that here we loose the time dependency since the focus will be centered around the study of the fixed points of the system.
Again, we consider N binary neurons $S_1, ..., S_N$ which can take value ± 1. Furthermore, the J_{ij} are drawn randomly and independently from $\{+\frac{1}{\sqrt{N}}, -\frac{1}{\sqrt{N}}\}$ and each value is chosen with probability $\frac{1}{2}$.
It is trivial to notice that, given the role of the *sgn* function in the equation, with great values of J_D every point is a fixed point. The system exhibits three different phases:

1. $J_D \lesssim 0.1$: only unstable isolated fixed points

2. $0.2 < J_D \lesssim 0.8$: the system presents a high-dimensional *cluster* of fixed points which co-exists with the same isolated fixed points as in 1.

3. $J_D > 0.8$: every point is a fixed point. This is given by the fact that the sum in 5 is $O\left(\frac{1}{\sqrt{N}}\right)$ whether J_D is $O(1)$, resulting in a dominating behaviour as I mentioned earlier.

In Figure 1 I plot the evolution of the percentage of fixed points found for an increasing number of neurons, since usually phase transitions are analysed in the limit of $N \to \infty$. Clearly, there are computational constraints: at every iteration we perform matrix-vector multiplication with a matrix of size NxN; thus, as N increases, simulations become slower. I initialized 500 possible random initial configurations of N neurons and let the network evolve through its dynamics. Indeed, we can see a phase transition for a value of J_D between 0.5 and 0.8, which means that below this critical value the network is not able to converge and the percentage of fixed points found is zero, while above it every point converges to its own fixed point in a few number of iterations.

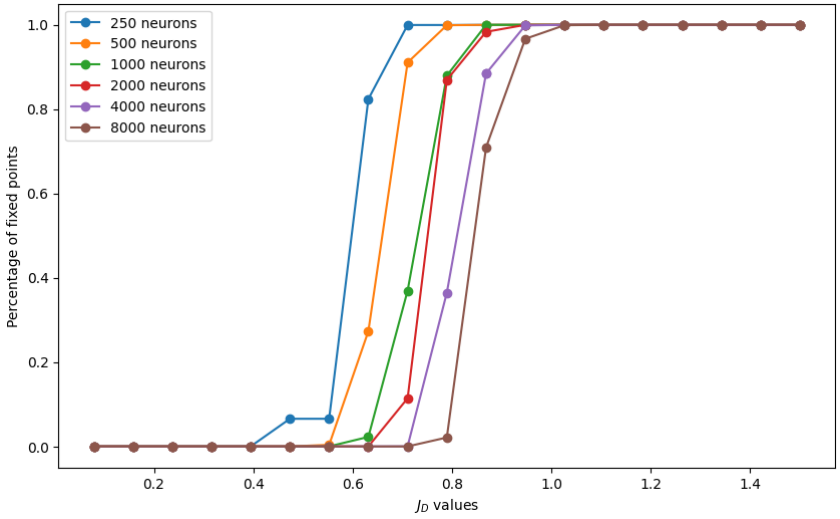


Figure 1: Percentage of fixed point found against different J_D values. Simulations are carried out for 250, 500, 1000, 2000, 4000 and 8000 neurons.

Despite the fact that even below the critical value of $J_D^c \approx 0.8$ there was proven to be an exponential number of fixed points [12] which grows up until a point of saturation when $J_D \approx 2$, no simulation is able to find it. Computationally, we can see only one phase transition at this precise value. What would also be interesting to determine, since it cannot be really understood from Figure 1, is whether this critical value keeps growing as a function of N or the fact that it changes as N grows is just a finite-size effect resulting, instead, in this critical value reaching a *plateau* for large N. I decided to plot (see Figure 2) the evolution of J_D^c as a function of $\frac{1}{N}$.

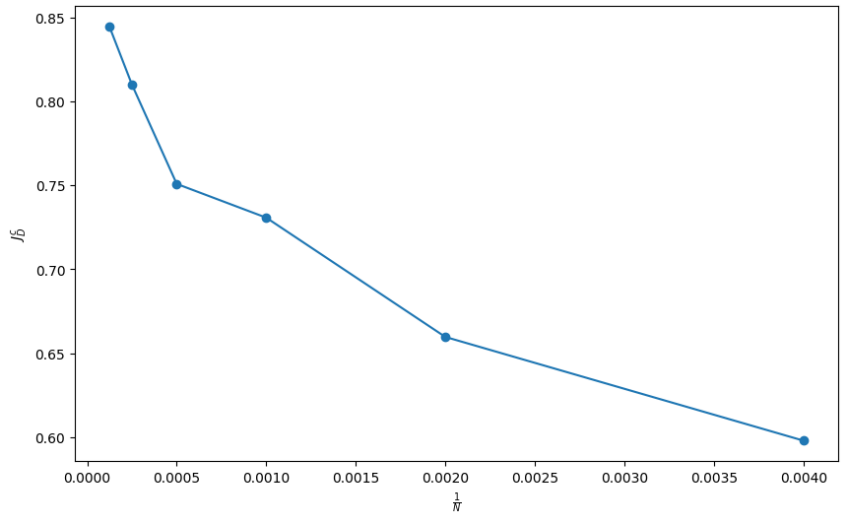


Figure 2: J_D^c as a function of the number of neurons considered in the network. We see that there might not be a finite limit as $N \to \infty$.

From the simulations it more likely seems that we can expect the following behaviour:

$$\lim_{\frac{1}{N} \to 0} J_D^c = \infty \tag{6}$$

J_D is, therefore, treated as an external parameter which is equal for all neurons and represents a self-coupling variable, which has the role of intensifying the value of each neuron at every iteration: being it positive, active neurons will be intensified in the positive direction, while silent neurons will become *more negative*. One might wonder what would happen if, alongside with the neurons, also the couplings changed based on their own dynamics.

3.1.1 Synaptic plasticity of the auto-interaction coefficient

We see that the value for which the network converges to fixed points is quite high, $J_D^c \approx 0.7$ in the cases considered. In attempt to lower this value to a more comparable scale to the one of the random J_{ij} and also inspired by the existence of synaptic plasticity for biological synapses between neurons (concept which will be more deeply discussed in section 5.1.2), I want to introduce a dynamic that updates the neurons and the respective auto-interaction coefficient, based on the input the neuron receives. So 5 becomes:

$$
\begin{aligned}
S_i &= sgn\left(\sum_{j\neq i} J_{ij}S_j + J_D(i)S_i\right) \\
J_D(i)^{t+1} &= \begin{cases} J_D(i)^t + f\left(\sum_{j\neq i} J_{ij}S_j\right) & \text{if } J_D(i)^t < J_D^{max} \\ J_D(i)^t & \text{otherwise} \end{cases}
\end{aligned} \tag{7}
$$

where in the system I explicitly underlined the dependency on time for clarity, f is a non-linear function that has the role of stabilizing the value of J_D to the desired one and J_D^{max} is set to see the different behaviour of the network in relation to it. Now that the auto-interaction becomes site-dependent, $J_D \to J_D(i)$, one thing we can do is compare the value of the J_D^c we found in the basic model with the average of the $J_D(i)^c$ for which this new dynamic brings the network to convergence. Results with a candidate function $f(h) = \frac{1}{1+e^{\beta|h|}}$ are displayed in Figure 3.

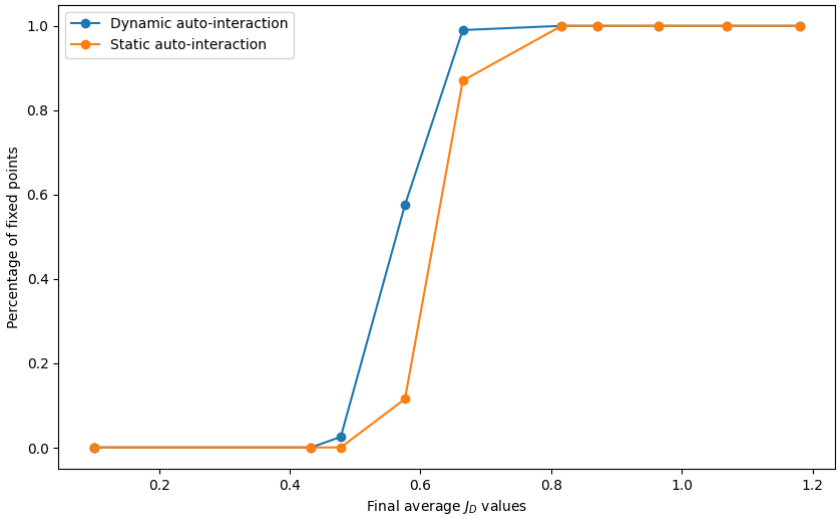


Figure 3: I let the network evolve for different values of J_D^{max} and, every time, recorded the value of the average $J_D(i)$ at the end of the dynamic. These quantities are then used in the original basic network as the fixed J_D values. I display here the percentage of fixed point obtained through the two dynamics as a function of those values. J_D^c is, indeed, slightly improved to a lower value. (Simulations are carried out in the case of 500 neurons and $\beta = 1$ i.e. $f(h) = \frac{1}{1+e^{|h|}}$).

For simplicity, I abandon for now the matter of synaptic plasticity and go back to

the fixed J_D case. In section 2 I mentioned that in simple models of neurons a threshold is usually included (which has been considered as equal to 0 up to now), so in the next section I will modify this model to include a constant threshold.

3.2 Model 2: threshold

I augment the model in section 3.1 to include a constant threshold value γ. The model is modified as follows:

$$S_i = sgn\left(\sum_{j\neq i} J_{ij}S_j + J_D S_i - \gamma\right) \tag{8}$$

This means, in words, that the neuron will be active only if its input (which includes all other neurons weighted by the respective values of J_{ij} plus itself weighted by the auto-interaction coefficient) is greater than the value of γ. As before, I repeat the same analysis on the phase transition of the percentage of fixed points, but this time as a function of both J_D and γ.

Simulations are carried out for different number of neurons as before, but this time we do not see explicitly the dependencies between the parameters values and the percentage of fixed points found. Instead, in Figure 4 I decided to plot, for every couple $J_D - \gamma$ considered, the ones in which the network started converging; thus, the line indicates the separation between a no-convergence zone (below) and the convergence one (above).

Again, it would be interesting, fixing J_D, to understand how the value of the critical γ changes as a function of $\frac{1}{N}$.

I performed the same simulations as before (see Figure 5), this time fixing the value of $J_D = 0$, in order to focus on the relationship between chaos and convergence as a function of just γ. We recover here a similar behaviour, with a critical $\gamma^c \approx 2.5$ and

$$\lim_{\frac{1}{N}\to 0} \gamma^c = \infty \tag{9}$$

The model becomes increasingly complicated having an explicit dependence on multiple parameters. As a consequence, also the convergence behaviour gets more intricate. Within the frameworks described above, we always start with 500 initial random patterns which evolve through a particular dynamics and sometimes converge to another point in

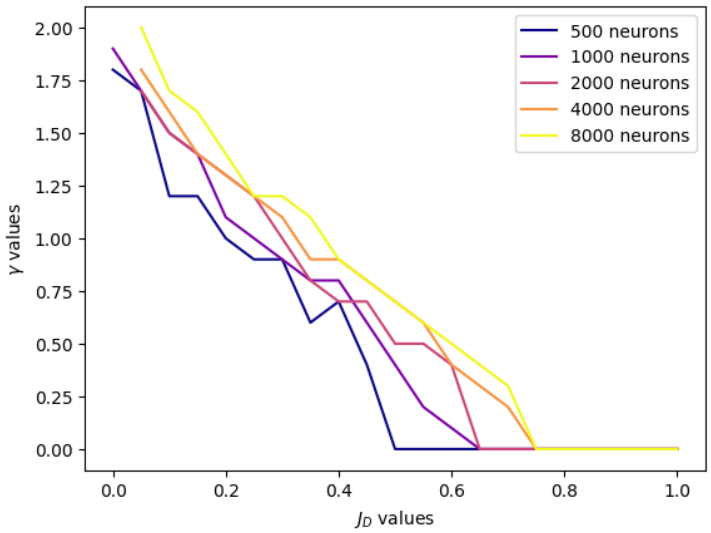
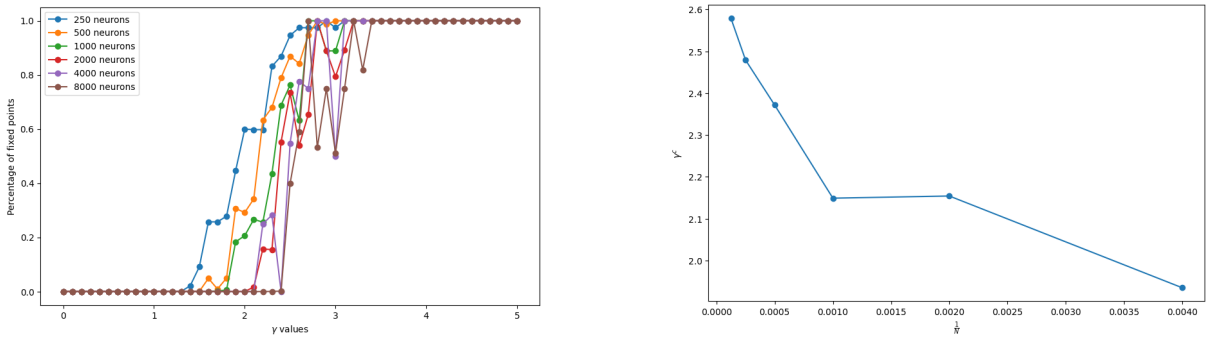


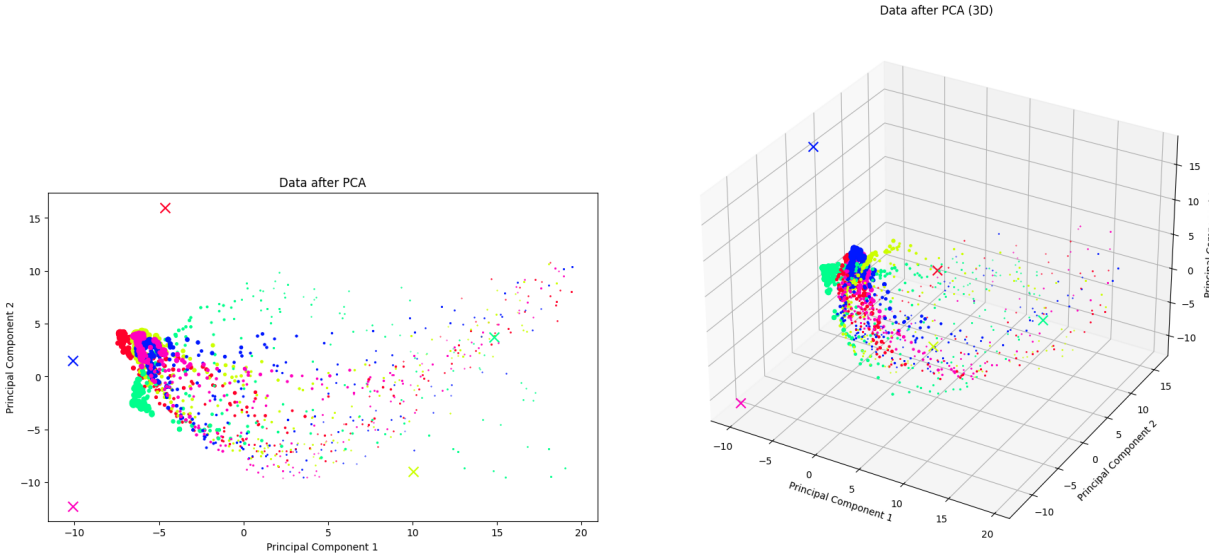
Figure 4: Transition lines separating zones of chaos (below) and convergence (above): the $J_D - \gamma$ couples for which the network starts converging to fixed points.



(a) (b)

Figure 5: In figure 5a I am analysing the convergence behaviour as a function of the parameter γ keeping J_D fixed and equal to 0. In figure 5b I display γ^c as a function of the number of neurons considered in the network. We can see that there is not a finite limit as $N \to \infty$.

the same high-dimensional space. Visualizing the actual positions of these points would provide valuable insights into the geometry of these complex spaces. Furthermore, we could ask ourselves questions like whether there is a convergence zone, or how *distant* the



(a) $2D$ PCA (b) $3D$ PCA

Figure 6: 5 out of the initial 500 points are indicated with a cross. Convergence points are related through colour to the initial ones, with a distinction based on the different values of parameters associated, underlined through dimension. In figure 6a $2D$ PCA is displayed, in figure 6b the $3D$ one.

initial points are from the converged ones (even though I do not define a proper notion of distance, focusing on the more intuitive concept of visual distance) and whether there is some correlation regarding the relative position. Motivated by these reasons, I performed PCA in both 2 and 3 dimensions. Figure 6 illustrates the following situation: I selected 5 out of the initial 500 random points in the N-dimensional space (where I chose a general $N = 500$ to balance computational efficiency and simulation efficacy), since including more points would result in a more accurate but visually cluttered representation. These initial points are marked with crosses. Each point, under different parameter values, converges to another point in the space; this correspondence is enhanced by the use of the same colour. The size of each point approximates the strength of the parameters, in particular, it is set as the re-scaled L^2 norm of γ and J_D.

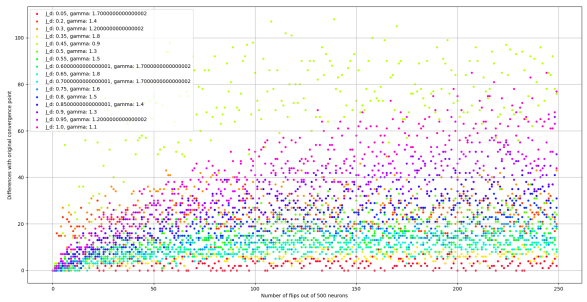
Unfortunately, the explained variance is very low (around 40% and 50% respectively): this means that the selected components are insufficient to capture the full high-dimensional

structure, with the variance being more vastly spread across different directions. Nevertheless, this still provides a rough sense of the point distribution and offers a visual understanding of the situation. One might want to experiment with different methods like UMAP or t-SNE, hopefully obtaining better results. From both the 2 and $3D$ cases, we can extract a couple of pieces of information: firstly, the network does not converge to trivial points, and secondly, with higher values of the parameters, points tend to concentrate around a high dimensional cluster.

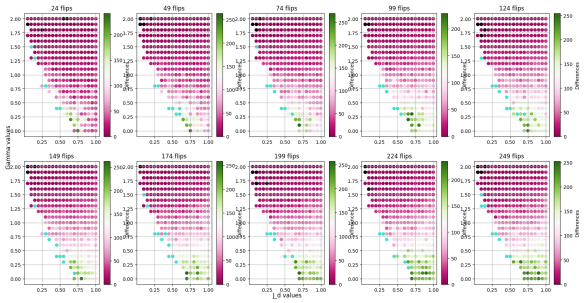
Up to this point I did not mention anything about the stability of the fixed points. It is very important in the context of associative memory to understand whether small variations to the fixed points of the dynamic get automatically corrected, which would mean that, indeed, the points are also stable. In order to do so, I performed the following steps with 500 and 1000 neurons:

1. I initialize the network in a random point
2. I select couples of J_D and γ for which the network is able to converge to some fixed point
3. For every couple I register the point to which the network converged, call those ξ^μ
4. I started flipping some neurons in random positions in regular steps up to $\frac{1}{2}$ of the number of neurons (maximum threshold needed for stability)
5. For every new point obtained after step 4., I initialize the network in this new point and let it evolve. If it converges again, I register this point as ζ^μ
6. I record how many differences there are between the ξ^μ and ζ^μ

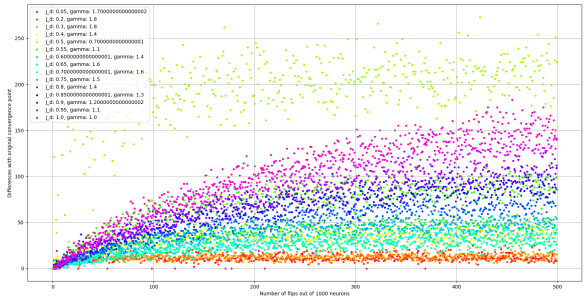
The results of this whole procedure are displayed in Figure 7. We can see that, in general, for higher values of J_D the points tend to show a higher number of differences. This might relate to what we were discussing in section 3.1: when J_D is very high it means that every point is a fixed point, so these fixed points are not at all stable because every neighbour is also a fixed point, no matter how close we are to the original one. Instead, we find a higher stability for γ high. Furthermore, we can see the following general trend: as we increase the number of flips, we tend to find more differences between the two convergence points, accounting for the higher noise we are introducing.



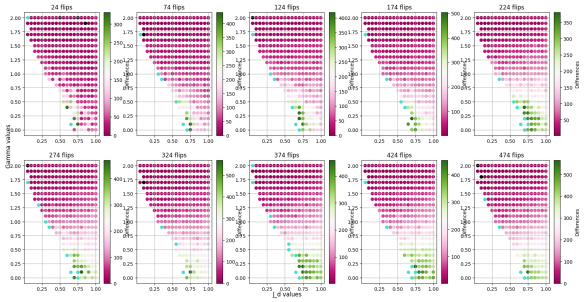
(a) 500 neurons



(b) 500 neurons



(c) 1000 neurons



(d) 1000 neurons

Figure 7: In figures 7a and 7c I displayed on the x-axis the number of neurons I flipped out of the total number of neurons (500 and 1000 respectively). On the y-axis I chose to indicate the number of differences between the components of the vectors of the original convergence point and the point to which the "flipped" network converged. The different colours refer to different values of the couple J_D - γ as shown in the legend. Instead, in figures 7b and 7d (where again we are analysing 500 and 1000 neurons respectively) I decided to replicate the style of the phase transition diagram I displayed in Figure 4. A single subplot represents the situation for a number of flips indicated above (I chose them in a regular step in a symbolic way, to be able to capture the general trend as the number of flips increases). Moreover, the x-axis represents different J_D values, whereas the y-axis refers to γ. This time the color accounts for the number of differences between the two convergence points, in particular, the heatmap goes from green (high number of differences) to pink (low number of differences) with the additional use of a black colour when the number of differences is zero and a light-blue one in the case in which the flipped point did not reach convergence anymore.

4 Towards a more "biological" network: the 0 - 1 model

As explained in the Introduction 1, the issue with these models remains: they are still far from resembling biological neuronal networks. One of the main problems is that we consider neurons in a ± 1 state, without distinguishing between excitatory and inhibitory functions. Additionally, this value is merely symbolic, just representing the presence or absence of a *spike*: to accurately represent a real neuron, we should also only consider positive values, since a neuron with a value of -1 lacks a concrete interpretation. Moreover, our model is just a discretization of what in computational neuroscience is described through systems of differential equations that account for the evolution of the firing rates over time, in recurrent or feed-forward networks of excitatory-inhibitory neurons.
One might wonder why this distinction matters when associative memory and deep convolutional neural networks perform quite well in practice. The answer lies in the computational power required by these models, which would be unbearable for the human brain. However, the human brain is one of the most efficient and effective computational instruments we know, which makes us very interested in replicating its capabilities and functionalities as closely as possible. We can start moving towards a "more biological" network by following a path described in the following sections, repeating the analysis carried out up to now, but with a slightly modified purpose in mind.

4.1 A possible mathematical connection

Our first objective becomes translating the network analysed up to now to a network where neurons take values in $\{0, 1\}$. We can make a first observation: we can define binary $0 - 1$ variables x_i in the following way:

$$S_i = 2x_i - 1 \Longrightarrow x_i = \frac{1}{2}(S_i + 1) \tag{10}$$

This way, $S_i = 1 \to x_i = 1$ and $S_i = -1 \to x_i = 0$ which is what we wanted. By using the substitution in 10 we can rewrite the model in 5 as follows:

$$2x_i - 1 = sgn\left(\sum_{j\neq i} J_{ij}(2x_j - 1) + J_D(2x_i - 1)\right)$$

$$x_i = \frac{1}{2}\left(1 + sgn\left(2\sum_{j\neq i} J_{ij}x_j - \sum_{j\neq i} J_{ij} + 2J_D x_i - J_D\right)\right)$$

We can see that the step function can be used here to rewrite the equation as follows:

$$x_i = \Theta\left(\sum_{j\neq i} J_{ij}x_j + J_D x_i - \alpha_i\right) \tag{11}$$

where $\alpha_i = \frac{1}{2}\left(\sum_{j\neq i} J_{ij} + J_D\right)$ is a site dependent quantity.

4.2 The models

This brings us to the next model I want to analyze. Imagine having a network of real neurons which are excitatory and inhibitory. We consider here a total number of N neurons of which $N-1$ are excitatory and 1 is inhibitory. Excitatory neurons are connected with each other and to the inhibitory neuron as before, while the inhibitory neuron gives a constant negative input to all other neurons. Thus, we can separate the x_i to distinguish between the two different roles of neurons in the following way:

$x_i \longrightarrow \begin{cases} x_i \\ y \end{cases}$ where now x_i indicate just the excitatory neurons so $i = 1, ..., N-1$ and y indicates the inhibitory neuron in the network.

4.2.1 Simplest (discrete) model of an excitatory-inhibitory population

Taking into consideration this simple distinction, the dynammics described in 11 becomes the following:

$$\begin{cases} x_i = \Theta\left(\sum_{j\neq i} J_{ij}x_j + J_D x_i - Iy - \alpha_i\right) \\ y = \Theta\left(\frac{1}{N}\sum_j x_j\right) \end{cases} \tag{12}$$

where $I > 0$ is the coefficient of all the outgoing connections from the inhibitory neuron, which clearly appears with a minus sign in the equation because inhibitory neurons

have the role of lowering the activity. J_{ij} here indicate again the synaptic efficacy between neuron i and neuron j, with the only difference that (before normalization) $J_{ij} = \{0, 1\}$. A thing we can do in some cases to simplify the model is to consider a constant threshold instead of a site dependent one, so to use $\alpha_i \approx J_D$ We can even remove the step function in the case of the inhibitory neuron, so to make it evolve more freely. So 12 becomes:

$$\begin{cases} x_i = \Theta\left(\sum_{j \neq i} J_{ij} x_j + J_D x_i - I y - \alpha_i\right) \\ y = \frac{1}{N} \sum_j x_j \end{cases} \tag{13}$$

4.2.2 The problem of the trivial fixed points

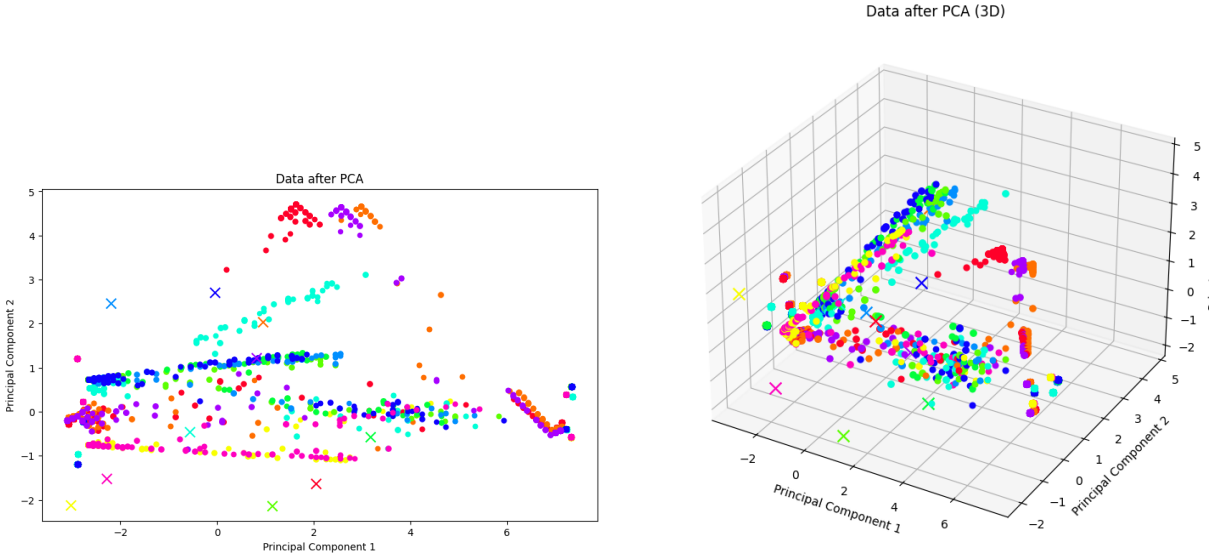
With networks like 12 and 13 we incur in the following problem: from simulations we can either see a convergence to the trivial fixed points, which are globally stable points of the dynamics (for example where neurons take value of all 0 or all 1), or no convergence, which at this point is more likely due to an insufficient number of iterations of the dynamics rather than actual chaotic behaviour.

4.2.3 Spiking threshold

Also motivated by the biological behaviour, what we can do is to introduce a threshold value γ which, as before, would account for the fact that neurons spike only when their input is higher than a certain quantity: doing so, we advance towards our primary goal of approximating biological systems, while also aiming to destabilize the network from trivial fixed points. The dynamic becomes the following:

$$\begin{cases} x_i = \Theta\left(\sum_{j \neq i} J_{ij} x_j + J_D x_i - I y - \alpha_i - \gamma\right) \\ y = \frac{1}{N} \sum_j x_j - \gamma \end{cases} \tag{14}$$

The percentage of points which reach convergence is usually very high and a lot of points still get stuck in the trivial fixed points. Nevertheless, as we can see in Figure 8, where I display the PCA on the initial and convergence points, despite the presence of a lot of regularity, not all the points are equal, meaning that, indeed, the introduction of a threshold managed to destabilize the network a bit. This time, the explained variance is



(a) $2D$ PCA (b) $3D$ PCA

Figure 8: PCA on 10 out of the initial 200 points. In figure 9a $2D$ PCA is displayed, in figure 9b the $3D$ one.

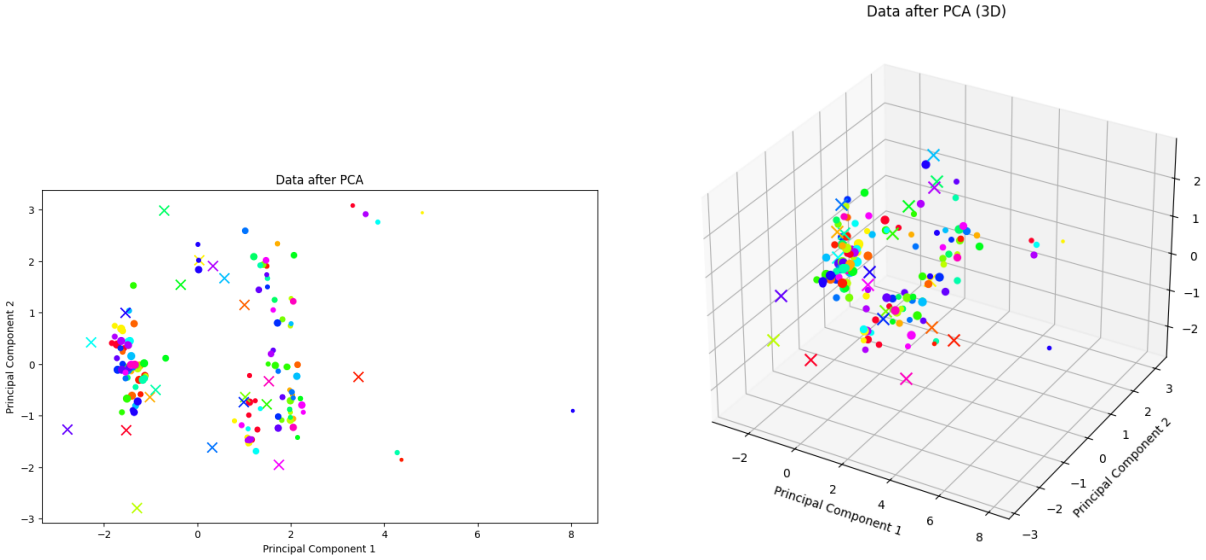
around 95% and 97% for the $2D$ and $3D$ cases respectively, which means that what we see is quite a good representation of the points in the 100-dimensional space. The general reasoning behind shape, color and dimension is the same as in Figure 6.

4.2.4 A more realistic distinction between neuronal roles

If we think about the behaviour of excitatory and inhibitory neurons, we would like that, indeed, inhibitory ones spike when the activity is *too high* (i.e. exceeds a certain quantity), but on the other hand, excitatory ones should behave in the opposite fashion, which is spike when the activity is *too low*. Therefore, we modify once again the dynamics to account for this consideration:

$$\begin{cases} x_i = \Theta\left(-\left(\sum_{j\neq i} J_{ij}x_j + J_D x_i - Iy - \alpha_i\right) + \gamma\right) \\ y = \Theta\left(\frac{1}{N}\sum_j x_j - \gamma\right) \end{cases} \tag{15}$$

Here, the relationship between the parameters result in being way more irregular and



(a) $2D$ PCA

(b) $3D$ PCA

Figure 9: PCA on 20 out of the initial 200 points. In figure 9a $2D$ PCA is displayed, in figure 9b the $3D$ one.

difficult to capture than before. In fact as before, I believe it is interesting to focus the attention more on the space, rather than the parameter themselves. I considered a network of $N = 100$ neurons and, again, I was able to deviate the network a bit from the trivial convergence points. In order to see this concretely I performed the PCA (9) on the initial and convergence points. Again, the explained variance is high, around 80% and 90% for the $2D$ and $3D$ ones respectively, but lower than before, which, in this case, means that with this model the network tends to converge less to the trivial fixed points and patterns are more widely and irregularly spread in the space. Since this time fewer points reach convergence, to get a better visual sense of the situation, I selected 20 of the initial patterns.

4.2.5 A different distribution for J_{ij}

In the subsequent sections I will introduce more realistic models of neuronal dynamics. Those are continuous models where the focus is shifted on the firing rate and its evolution

through differential equations, rather than being limited to the binary variables x_i which are able to just account for the spike - no spike relationship. Inspired by this discussion in a way that will become clearer later, I decided to experiment again with the synaptic efficacies J_{ij}, not through their evolution, rather with their initial distribution which, up to now, have always been chosen in the following ways:

either

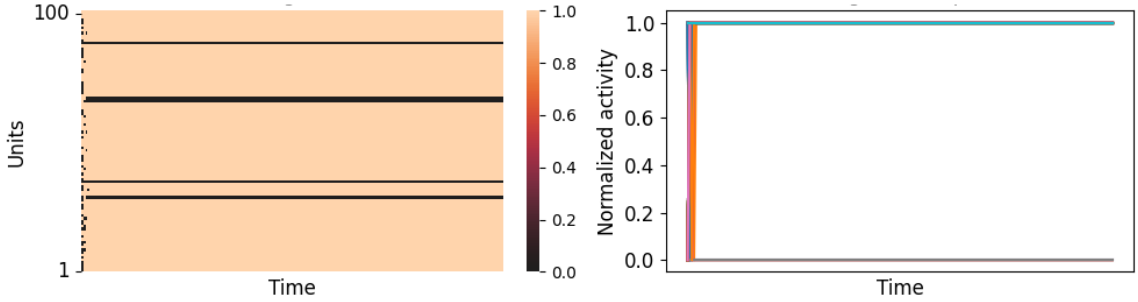
$$\begin{cases} +\frac{1}{\sqrt{N}} & \text{with probability} \frac{1}{2} \\ -\frac{1}{\sqrt{N}} & \text{otherwise} \end{cases}$$

or

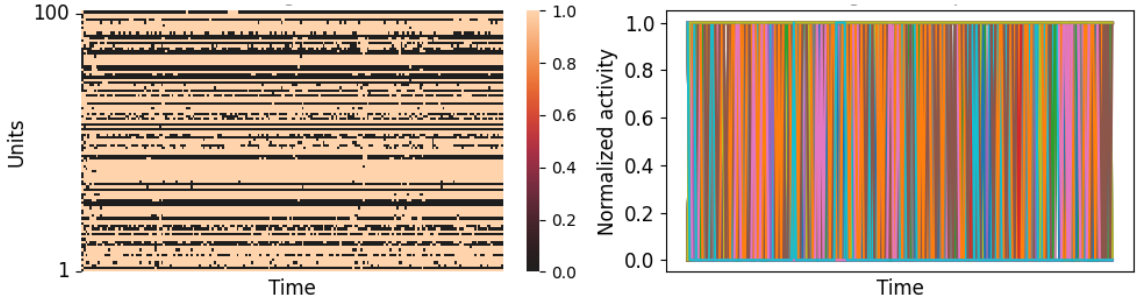
$$\begin{cases} +\frac{1}{\sqrt{N}} & \text{with probability} \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Within the last introduced dynamics, we can instead consider a different distribution, in particular, we can assume the weights are drawn independently and identically distributed from a Gaussian with mean 0 and variance $\frac{g^2}{N}$, where g is a control parameter. The main reason why I decided to introduce this modification, which directly affects the weights, is to incorporate some sort of heterogeneity, because until now the main changes to the model were centered around different types of dynamic and spiking activity. Instead, in living beings, synapses themselves display significant structural and functional variability, even among synapses of the same type, leading to a wide range of functional capabilities [13] [14]. I introduce here a slightly different type of analysis which will become useful in the following sections when I present continuous models. Up until now we have just focused our attention on the random initial points and the final points to which the dynamics converge, when there exist one. All this, either in relation to the "geometry" of the space, or to different combinations of parameters in order to be able to see the transition to chaotic behaviour. We have not very much taken into consideration explicitly the time dependency: what I decided to display here is the evolution in time of the value of single neurons, once certain parameters have been fixed.

The results are displayed in Figure 10 where the difference between chaos and conver-



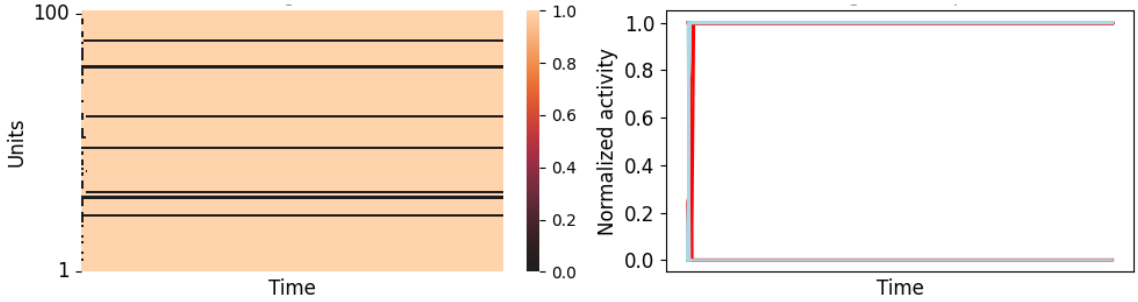
(a) Convergence behaviour: $I = 2$ and $g = 5$



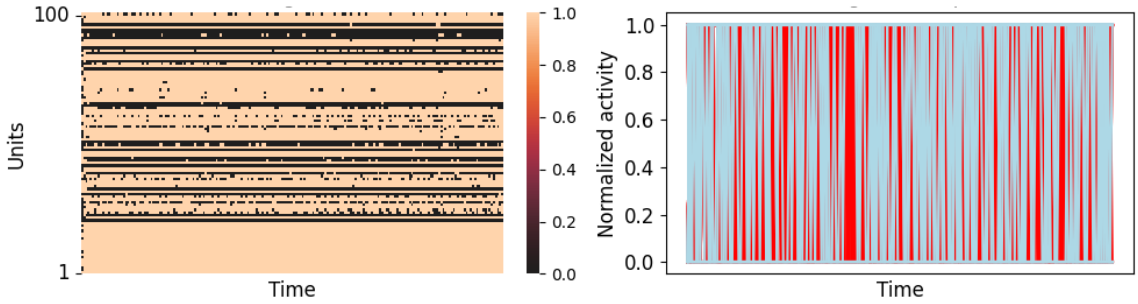
(b) Chaotic behaviour: $I = 2$ and $g = 10$

Figure 10: On the left a raster plot in which each row represents the activity of one single neuron through time. On the right we can see the evolution of the activity of single neurons (each neuron represented with a different colour) through time. In figure 10a we can see that the activity of every neuron remains fixed through time, which stands for convergence to a fixed point. Instead, in figure 10b units keep oscillating between 0 and 1 displaying a chaotic behaviour.

gence is highlighted clearly . I should clarify that the kind of activity displayed here is different from the one considered in the next section, where it will be closely related to the "Firing rate". In fact, in discrete scenarios we cannot talk about a firing rate which is typically an instantaneous measure, defined as the probability density of firing averaged over multiple trials [15]. Instead, here, we always simply refer to the presence or absence of spikes within a fixed time interval Δt (which essentially corresponds to the number of iterations in the dynamic's update process) and, since we are utilizing once again the step



(a) Convergence behaviour: $I = 0.1$ and $g = 5$



(b) Chaotic behaviour: $I = 0.1$ and $g = 10$

Figure 11: On the left the raster plot, on the right the evolution of the activity of single neurons (inhibitory activity in blue, excitatory in red) through time. In figure 11a we see convergence to a fixed point. Instead, in figure 11b we observe chaotic behaviour.

function, the only possible values are limited to 0 and 1.

One could also try to increase the number of inhibitory neurons with a double objective: firstly, to see whether activity gets balanced or destabilized in different ways and secondly, to consider a network comparable to the ones we find in real networks of neurons in the brain, in which pre-synaptic input is received in a ratio of excitatory-inhibitory neurons of 8:2. I display results of a network of 100 neurons (20 inhibitory and 80 excitatory) in Figure 11.

5 Recurrent Neural Networks: the continuous case

The models discussed so far closely resemble theoretical and computational models commonly used in neuroscience to describe interacting neuronal systems [15]. Building on this foundation, we can now introduce a new family of models that are more closely aligned with these biological networks, but which can also be analyzed using the same methods previously applied, allowing us to explore how the model parameters influence whether the system exhibits chaotic behavior or converges to a fixed state. There are two main classes [16] of models I wish to mention in the following sections: the first uses a Gaussian distributed connectivity matrix and transforms the synaptic input to an output firing rate through a transfer function; the second class, which is even more closely inspired by biology, obeys Dale's law differentiating between excitatory and inhibitory populations and neurons are either binary variables which take value in $\{0, 1\}$ or *LIF* neurons [17] [18].

5.1 Firing rate models

The models belonging to the first class are generally referred to with the term *firing-rate models*, since the exact spike sequence of neurons is replaced by the firing rate. An important remark regards the fact that it is the actual spike sequences that in reality affects the network's dynamics behaviour; thus, in order to be sure that the substitution we are making is justified we should understand if the firing rate (averaged over trials) is a valid approximation of this quantity and this is typically true if we consider that each neuron receives a huge amount of inputs. While individual synaptic inputs exhibit high trial-to-trial variability, the collective input from many uncorrelated pre-synaptic neurons averages out, making the firing rate a reliable approximation. This averaging effect occurs because the mean total input increases linearly with the number of synapses, whereas the variability only increases with the square root of the number of synapses. However, if pre-synaptic neurons are highly correlated (e.g. if they fire synchronously), the firing-rate model may fail to accurately describe the network's dynamics.

5.1.1 The simplest rate-based smooth dynamics

The first model I want to analyze consists of N neurons which, this time, are continuous variables denoted as $\{S_i(t)\}$, where $i = 1, ..., N$ and $-1 \leq S_i \leq 1$. Each neuron has an associated local field which obeys the following relation: $S_i = \phi(h_i)$ and $-\infty \leq h_i \leq \infty$.

$\phi(x)$ is a nonlinear gain function representing the neuron's input-output characteristics and it is assumed to have a sigmoid shape i.e. to present the following characteristics:

1. $\phi(\pm\infty) = \pm 1$
2. It is odd: $\phi(-x) = -\phi(x)$

Thus, a common choice is to use $\phi(x) = \tanh(x)$.
The dynamic of the network is described by N coupled first-order differential equations [19]:

$$\frac{\partial h_i}{\partial t} = -h_i + s\phi(h_i) + g\sum_j J_{ij}\phi(h_j) \tag{16}$$

The synaptic couplings are random, independent variables Gaussian distributed with mean 0 and variance $\frac{1}{N}$. The self-coupling term s and the parameter g, which represents the strength of the couplings (also defined *network gain*), determine the different long-term behaviour of the network.
I will focus on the particular case in which $s = 0$. Thus, the dynamic becomes the following:

$$\frac{\partial h_i}{\partial t} = -h_i + g\sum_j J_{ij}\tanh(h_j) \tag{17}$$

As I did in section 2, I want to give some quick intuition about the analytical solution of these equations, before moving on to the results I obtained through numerical analysis. The model has an exact solution as $N \to \infty$, found through the dynamical mean-field theory (DMFT) [20]. The basic idea of this approach is to replace the last term on the right-hand side of 17 with a time-dependent Gaussian random variable $\eta_i(t)$. This allows to perform different realizations of the matrix $(J_{ij})_{i,j}$ and compute the properties of the network as an average over those. Doing so, all the neurons in the network become

equivalent, so the system of differential equations can be replaced by a single stochastic self-consistent differential equation, bringing us to a simpler setting:

$$\frac{\partial h}{\partial t} = -h + \eta(t). \tag{18}$$

Call the solutions of this equation $h(t;\eta)$ to denote explicitly the dependence on both time and the particular realization of η. We are interested in two quantities: the first moment of η (which is trivially zero) and its second moment. To compute the latter self-consistently, we need to define the auto-correlation function (averaged over realizations of $(J_{ij})_{i,j}$). Using the identity $\langle J_{ij}J_{kl}\rangle_J = \frac{1}{N}\delta_{ik}\delta_{jl}$ we get:

$$\begin{aligned} C(\tau) &= \frac{1}{N}\sum_{i=1}^{N}\left\langle \sum_{j=1}^{N} J_{ij}\tanh(h_j(t)) \sum_{k=1}^{N} J_{ik}\tanh(h_k(t+\tau))\right\rangle_t \\ &= \frac{1}{N}\sum_{j=1}^{N}\langle \tanh(h_j(t))\tanh(h_j(t+\tau))\rangle_t \end{aligned} \tag{19}$$

where the variable over which we are averaging is indicated as a subscript of the angle brackets. Therefore the second moment of η becomes:

$$\langle \eta(t)\eta(t+\tau)\rangle_t = g^2 C(\tau) \tag{20}$$

These quantities are fundamental because, if they match the first and second moments of the term they replaced in 17, then the distribution of $h_i(t)$ that solves the original equation 17, when averaged over realizations of the stochastic matrix $(J_{ij})_{i,j}$, will exactly correspond to the set of solutions $h(t;\eta)$ over the distribution of η.

A similar study can be done to understand the stability of these systems, either locally through the analysis of the eigenvalues of a particular stability matrix, or globally through the largest Lyapunov exponent, which represents, intuitively, the rate of the exponential growth of the distance between two points when they undergo a certain temporal dynamic. A system which, for example, is highly sensitive to initial conditions will have a positive exponent, accounting for the fact that small differences in starting conditions lead to exponential divergence. Vice-versa for a negative one.

Let us now move on to the simulations I carried out.

It was shown that the system exhibits a phase transition at critical $g^c = 1$ (Figure 12): for $g < g^c$ the activity converges to 0, while for $g > g^c$ we observe chaotic behaviour.

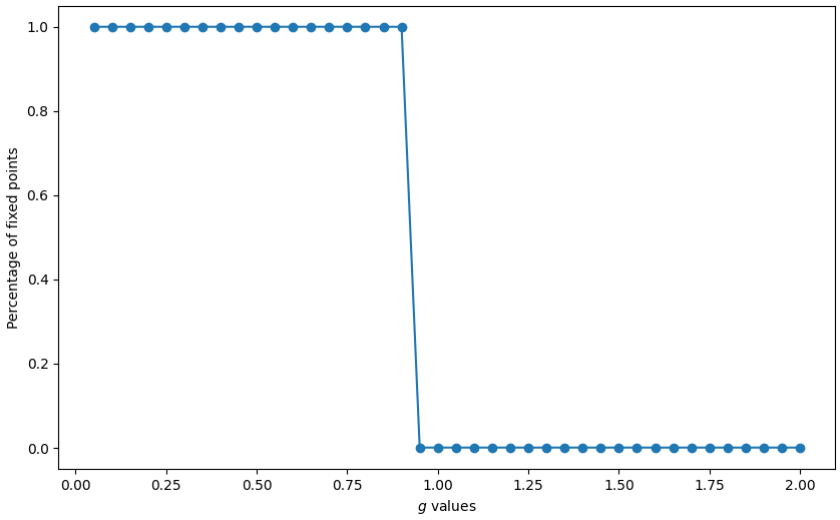


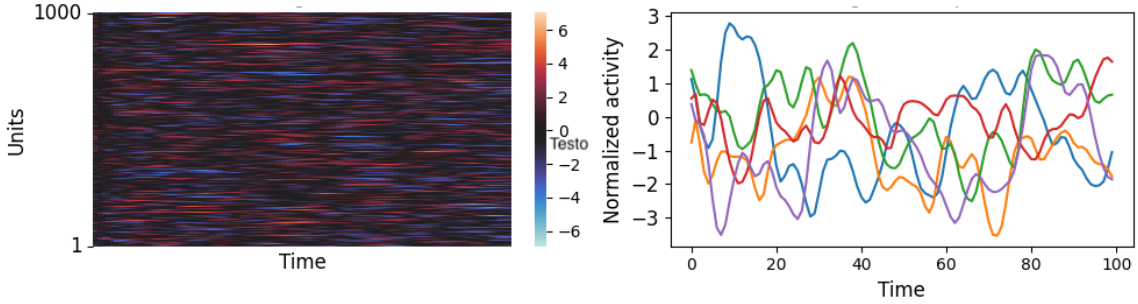
Figure 12: Phase transition at the critical value of the synaptic gain g^c. On the x-axis different values of the parameter g are displayed, on the y-axis the percentage of fixed points is plotted. In this case, the fixed point always corresponds to the activity decaying to zero.

I simulated the systems with two different methods: the first one is the Euler method [21], which performs a discretization of the differential equation making it suitable for coding, considering the discrete time step Δt as iterations of the function which performs the dynamic update; the second one is the Python library *scipy.integrate* [1] which has routines already implemented to perform integration of differential equations. We can clearly see the difference between convergence and chaos selecting two values of g for which the network exhibits these two types of long-term behaviour and plotting the evolution in time of the h_i (Figure 13).

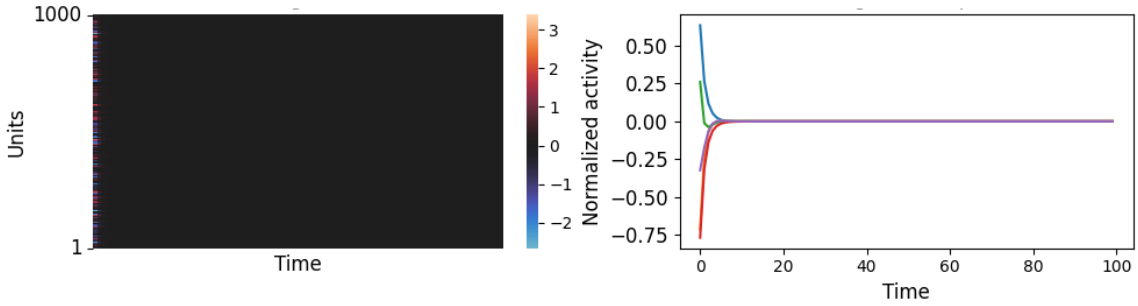
5.1.2 Local Hebbian and anti-Hebbian plasticity: dynamic couplings

A crucial point I started bringing up sparsely up to now, is that of the importance of the weights. I now wish to dig deeply into this matter.

[1] https://docs.scipy.org/doc/scipy/reference/integrate.htmlscipy.integrate documentation



(a) Chaotic behaviour: $g = 2$



(b) Convergence behaviour: $g = 0.3$

Figure 13: On the left, the continuous correspondent of a raster plot in which each row represents the activity of one single neuron through time. On the right, the evolution of the activity of a sample of neurons (each neuron represented with a different colour) through time. In figure 13a we can see a highly variable activity, typical of chaos. Instead, in figure 13b there is a uniform decay to the 0 fixed point. (These results are obtained with the *scipy.integrate.odeint* routine).

Up to now the attention has mainly been focused on the neurons. Indeed, the dynamic is centered around the evolution of the neuronal activity, whereas the parameters, and in particular the synaptic efficacies, are quenched variables, which, after being drawn randomly, are fixed once they enter the dynamics. However, as already mentioned in 3.1.1 and 4.2.5, synapses do not merely transmit information, they also play an active role in information processing, being highly variable in type and timescales [22]. Long-term synaptic evolution relates more to learning and memory, while short-term changes

support various types of computations. Thus, these different forms of synaptic plasticity are essential for allowing single neurons to transmit a wide range of signals. The model becomes the following [23]:

$$\frac{\partial h_i}{\partial t} = -h_i + \sum_j W_{ij}(t) \tanh(h_j) \tag{21}$$

where I explicitly displayed the dependency of W_{ij} on time to highlight the fact that also weights now are evolving in time. In particular, we define them as the sum of quenched and varying terms as follows:

$$W_{ij}(t) = g \cdot J_{ij} + A_{ij}(t) \tag{22}$$

where $J_{ij} \sim \mathcal{N}(0, \frac{1}{N})$ as before, while the $A_{ij}(t)$ evolve in time through the following local plasticity rule:

$$p\frac{\partial A_{ij}}{\partial t} = -A_{ij} + \frac{k}{N} \tanh(h_i) \tanh(h_j) \tag{23}$$

The timescale of synaptic decay is usually taken as $p > 1$ since, biologically, it is probably not shorter than the neuronal timescale (which is here equal to 1). k is the strength of the local plasticity: in particular, $k > 0$ represents *Hebbian plasticity*, which refers to an increase in the strength of the connection when there is simultaneous activity, whereas $k < 0$ stands for *anti-Hebbian plasticity*, which determines a decrease in the synaptic strength when activity occurs separately i.e. the opposite process [24] [25] [26]. I fix the parameter $p = 2.5$ and analyze the long-time behaviour of the dynamic letting the other parameters vary; the different types of behaviour are explained in Figure 14.

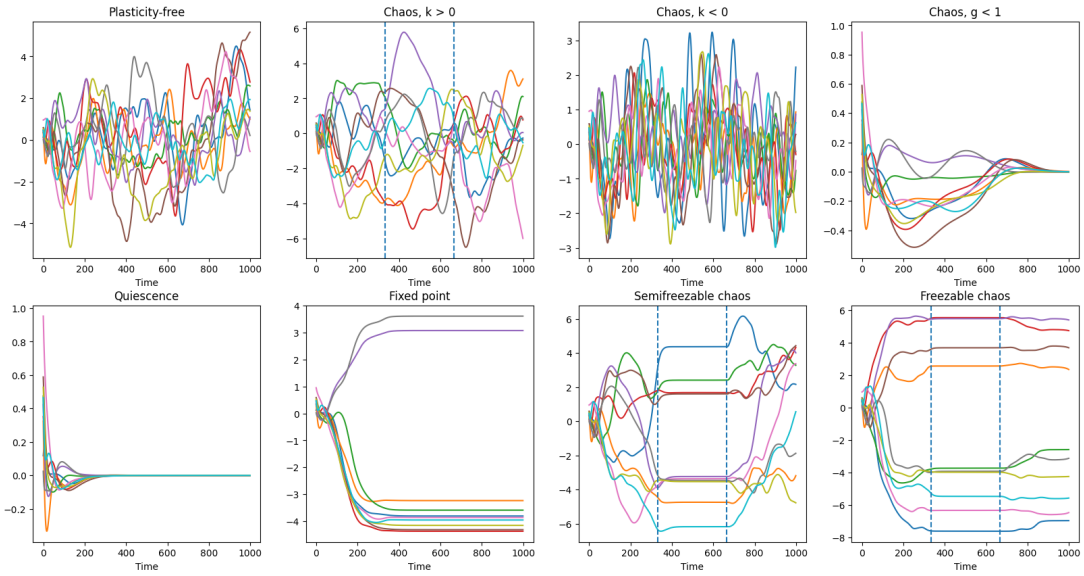


Figure 14: Experiments carried out on different values of the parameters to explore different types of behaviour. In the top row: if $k = 0$, $g > 1$ we retrieve the behaviour of the basic model without dynamical update in the weights, in particular, we see chaotic behaviour; in the second and third subplots $g > 1$ remains the same while I display the difference between $k > 0$ (Hebbian plasticity) and $k < 0$ (anti-Hebbian plasticity) which has the role of decelerating and accelerating the activity respectively. In the bottom row I fixed $g < 1$ and recover the instantaneous attenuation of the activity to zero, however, if we use a k large enough we can inject some noise into the system which might induce a destabilization from the globally stable fixed point (top row, last subplot); in the second subplot we can see convergence to a non-trivial fixed point for a strong Hebbian plasticity; finally I employed a *halt - release* dynamic, meaning I let the network evolve through its dynamic up to a point when I fix the weights and interrupt their evolution through time; after a while, I begin updating the synaptic efficacies again. The situations in which I employed this mechanisms, differentiated from increasingly stronger values of k, are highlighted through the explicit use of additional vertical dashed lines which mark the beginning and end of the synaptic update interruption. In the top row, second subplot, we can see that this has no effect on the overall dynamic; instead, in the last two subplots, with the difference in the presence of more variability, we see that the state is retained by the system, therefore constituting a model of short-term memory.

5.2 Randomly diluted networks

The last models I want to rapidly introduce regard the interplay of excitatory-inhibitory populations [16], connected in a way that is motivated by biological constraints: output connections of excitatory neurons are non-negative, while the opposite is true for inhibitory neurons. The general formulation in case of multiple populations of neurons is the following:

$$\frac{\partial h_i^k}{\partial t} = -h_i^k + \sum_{l=1}^{P} \sum_{j=1}^{N} J_{ij}^{kl} \phi(h_j^l) + h_0^k m_0 \tag{24}$$

where we consider a new index $l = 1, ..., P$ to keep trace of the population we are considering and introduce the parameters h_0^k which denote some external activity given as constant input to population k. J_{ij}^{kl} now denotes the synaptic efficacy between neuron i of population k and neuron j of population l. This is considered to be a *randomly diluted* network, which means that J_{ij}^{kl} can be $\begin{cases} \neq 0 & \text{with probability} \frac{K}{N} \\ 0 & \text{otherwise} \end{cases}$.

We can replace this coefficient with our usual matrix of Gaussian distributed connections: $J_{ij}^{kl} \to \frac{\bar{g}^{kl}}{N} + g\mathcal{J}_{ij}^{kl}$ where, indeed, $\mathcal{J}_{ij}^{kl} \sim \mathcal{N}(0, \frac{1}{N})$, $\frac{\bar{g}^{kl}}{N}$ representing the contribution of the mean connections. Thus 24 becomes:

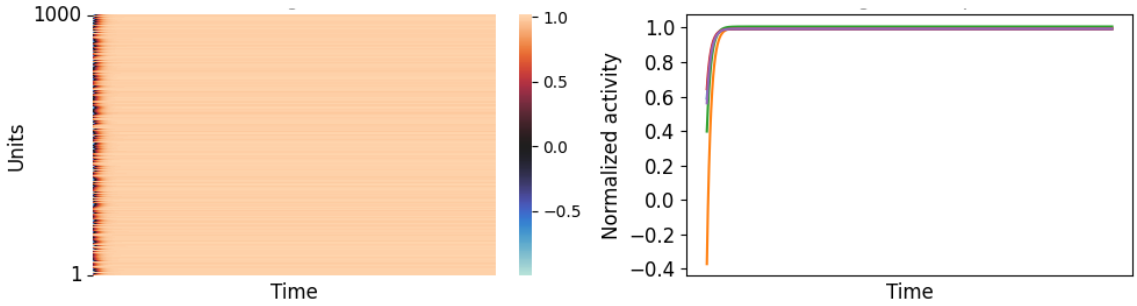
$$\frac{\partial h_i^k}{\partial t} = -h_i^k + g \sum_{l=1}^{P} \sum_{j=1}^{N} \mathcal{J}_{ij}^{kl} \phi(h_j^l) + \sum_{l=1}^{P} \frac{\bar{g}^{kl}}{N} \sum_{j=1}^{N} \phi(h_j^l) + h_0^k \tag{25}$$

5.2.1 The single inhibitory population case

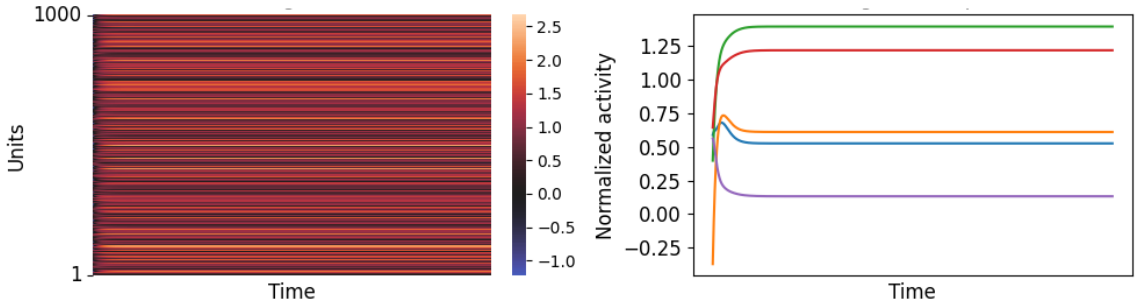
Firstly, I will focus on the most simple model, which is a network composed of a single population of excitatory neurons, which receives constant external excitatory input $h^E > 0$ and is connected within itself through a self-coupling gain parameter $\bar{g} < 0$. In formula:

$$\frac{\partial h_i}{\partial t} = -h_i + g \sum_{j=1}^{N} \mathcal{J}_{ij} \phi(h_j) + \frac{\bar{g}}{N} \sum_{j=1}^{N} \phi(h_j) + h_0^E \tag{26}$$

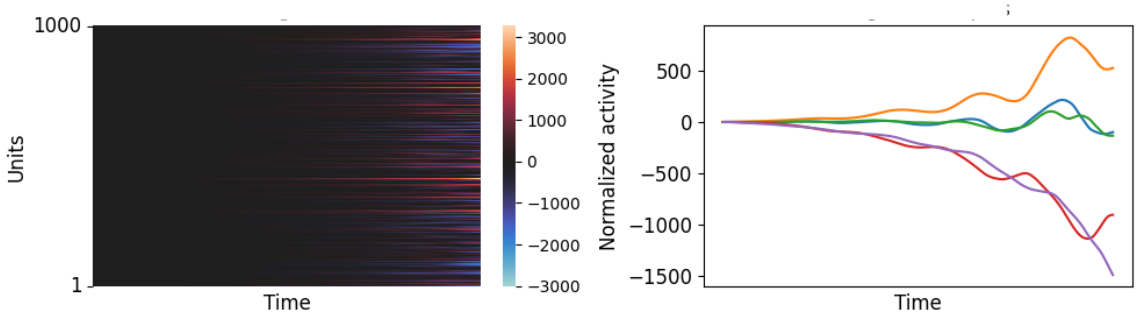
In particular, I will employ $\phi(x) = max(x, 0)$. The existence of two different phase transitions, one from fixed point to chaos, the other from chaos to divergence, has been



(a) $g = 0.5, \bar{g} = -5$



(b) $g = 4, \bar{g} = -25$



(c) $g = 7, \bar{g} = -5$

Figure 15: The general framework of the plots is the same as in 13. The difference here is that we get to observe a divergence behaviour 15c in which activity seems to gradually and increasingly move away from 0.

proven to exist based on different values of the couple $g - \bar{g}$. Some examples are displayed in Figure 15.

5.2.2 Two populations case

We can augment the above mentioned model to analyse the case of two populations of neurons, one excitatory and one inhibitory. Going back to the randomly diluted network formulation in 24, we get the following dynamic:

$$\begin{cases} \frac{\partial h_i^E}{\partial t} = -h_i^E + \sum_{j=1}^{N} J_{ij}^{EE}\phi(h_j^E) + + \sum_{j=1}^{N} J_{ij}^{EI}\phi(h_j^I) + h_0^E \\ \frac{\partial h_i^I}{\partial t} = -h_i^I + \sum_{j=1}^{N} J_{ij}^{IE}\phi(h_j^E) + + \sum_{j=1}^{N} J_{ij}^{II}\phi(h_j^I) + h_0^I \end{cases} \tag{27}$$

It directly follows [27] [28] that we can rewrite this intricate process focusing on the mean activity of the populations, instead of single neurons, using what can be interpreted as an average synaptic efficacy between the two populations (see Figure 16 for a visual interpretation).

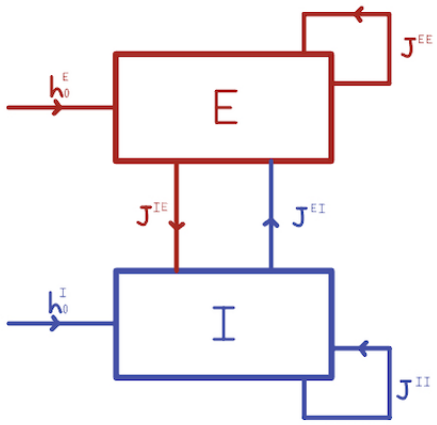


Figure 16: A visual representation of the interaction between two representative units of the average activity of a two-population network of excitatory and inhibitory neurons.

With a slight abuse of notation, considering populations of neurons as single units and replacing every quantity with its respective average, h^E will be used to denote the average activity of the excitatory population and the same for h^I with the inhibitory one; external incoming activity will be denoted with h_o^E and h_0^I respectively. Furthermore, we can consider the *average self-coupling term* defined as $J^{EE} > 0$ and $J^{II} < 0$. Lastly, the signal from the inhibitory population will be weigthed through a negative *average coupling*

term J^{EI} and the opposite is valid for the excitatory population with J^{IE}. What we get is the following dynamic:

$$\begin{cases} \frac{\partial h^E}{\partial t} = -h^E + J^{EE}\phi(h^E) + J^{EI}\phi(h^I) + h_0^E \\ \frac{\partial h^I}{\partial t} = -h^I + J^{II}\phi(h^I) + J^{IE}\phi(h^E) + h_0^I \end{cases} \tag{28}$$

where ϕ remains the power law transfer function (with unit exponent) employed before. For simplicity, I fixed external activity to a positive value in the case in which it is given as input to the excitatrory population, and to a negative one for the inhibitory case. Moreover I set $J^{EE} = J^{IE} = \alpha g$ for some parameter α, and $J^{II} = J^{EI} = -g$. There exist a phase transition from fixed point to chaos as a function of the couple $\alpha - g$, displayed in Figure 17.

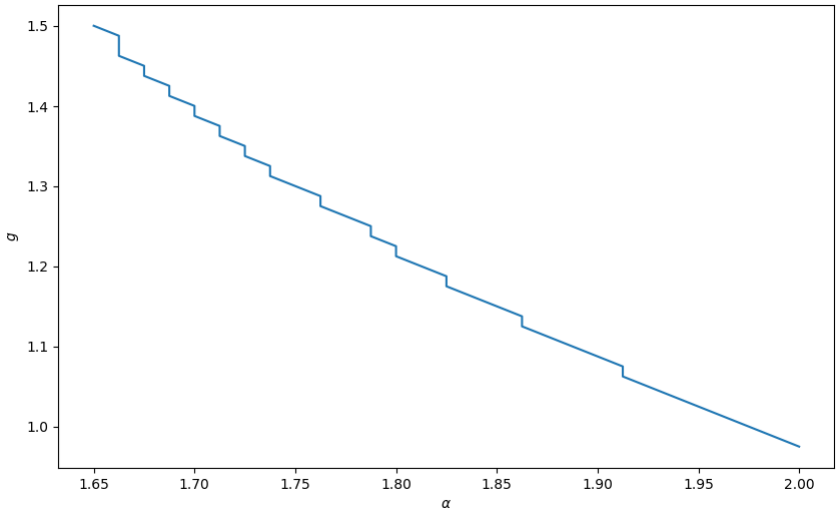


Figure 17: On the x-axis different values of α; on the y-axis different values of g. Separating line between the convergence zone (below) and the chaotic one (above).

6 Towards new frontiers for AI: conclusion and further inspiration

Through my work I wanted to give a general overview of associative memory models and dynamics of recurrently connected networks of neurons. The main purpose was to build a connection between mathematical models of neural networks and the modelling of human networks of neurons and synapses, based on a biological interpretation which could allow to almost not distinguish the two anymore. I started from the simplest model of a neuron and built a recurrent asymmetric network of neurons upon it, inspired by the Hopfield model. Then, I gradually moved towards a more biological model, through the use of an additional spiking threshold, separating neuronal roles and introducing a greater focus on the synaptic efficacy. Moreover, inspired by computational neuroscience, I started analyzing firing-rate based continuous models, in their simplest form and with the addition of synaptic plasticity, to reach a more realistic dynamic involving populations of excitatory and inhibitory neurons. Other than this main direction, I tried to analyse the interplay between chaos and convergence, in particular, how those might arise or disappear as a function of external parameters. I also focused on the evolution of activity through time and sometimes on the "geometry" of the space of fixed points and their stability. In the end, the main takeaway is that we were able to build fundamentally random models that, through some dynamic and dependently on some externally fixed parameters, were able to converge independently to their own fixed point. This is a fundamental step in the context of associative memory, but also in Machine Learning in general, when considering reliability, efficiency, and interpretability of the systems. Furthermore, as I hope will become clear through the next (final) discussion, these computational devices can turn out to be way more powerful than we may think.

6.1 An *upgraded* perceptron

In section 2 I implicitly introduced *the perceptron*, one of the first computational tools developed in the context of classification tasks mapping an input vector $\bar{x}^\mu$ to an output binary value y^μ. Through the following algorithm, the perceptron learns the correct

weights and *bias* to create a linear separator of the training data.

Algorithm 1 Perceptron Algorithm

```
Initialize the weights w_i = 0 ∀i and the bias b = 0
while stopping condition not met do
    for each pattern μ where μ = 1, ..., P and P total number of data points do
        Compute: ŷ^μ = sgn(w̄ · x̄^μ + b)
        if ŷ^μ ≠ y^μ then
            w̄ ← w̄ + y^μ x̄^μ
            b ← b + y^μ
        end if
    end for
end while
```

It was proven [29] that the maximum number of patterns a perceptron can learn is closely related to the dimension of the space of the training data. In particular, if input vectors are N-dimensional, a perceptron can learn up to $2N$ patterns.
In fact, if we give a closer look to the computation used to calculate $\hat{y}^\mu$ in the perceptron algorithm, it closely resembles the discrete dynamics described in sections 2 - 3. Thus, imagine having a discrete discrete dynamic like the one in 5 and do the following:

1. Select a J_D for which we are sure the network converges (e.g. $J_D = 0.9$)
2. Select P initial random patterns ξ^μ for $\mu = 1, ..., P$ to use as initialization for the network
3. Let the network evolve until convergence to fixed points ζ^μ, for $\mu = 1, ..., P$

We can interpret the mapping $\xi^\mu \to \zeta^\mu$ as a *clever* rearrangement of the points, made independently by the network itself through its internal dynamic. If, for every pattern, we extract the first bit and consider it as a label, we get two different *training sets*: $\left(\bar{x}^\mu_{(\xi)}, y^\mu_{(\xi)}\right)_{\mu=1,...,P}$ and $\left(\bar{x}^\mu_{(\zeta)}, y^\mu_{(\zeta)}\right)_{\mu=1,...,P}$ where each $\bar{x}^\mu$ has dimension $N-1$. Passing the first (random) dataset to a perceptron, we see that the perceptron theorem applies: as soon as $P > 2$ the accuracy drops significantly. Instead, a strange phenomenon arises when

we pass the second dataset (see Figure 18): since these points have already undergone a dynamic similar to the perceptron rule, it is possible that the latter's job gets simplified and the critical value of $\frac{P}{N}$ becomes ≈ 5, after which the accuracy (still high) drops below 95%. To be completely honest, despite still being a powerful result, it is not really a process of learning, since the label are not imposed from the outside, yet learned through the dynamic.

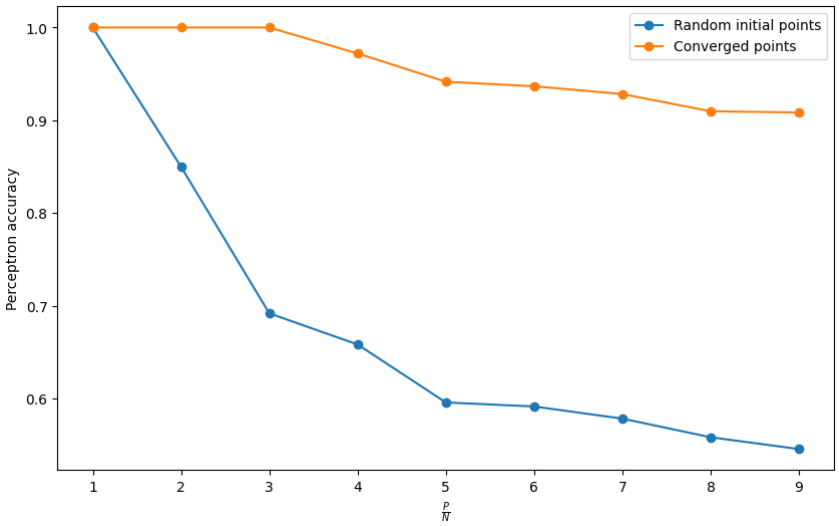


Figure 18: The accuracy of a perceptron on random data and the respective fixed point obtained through the dynamic in 5 with $J_D = 0.9$. On the x-axis the evolution of the ratio $\frac{P}{N}$, on the y-axis the perceptron's accuracy.

A further improvement one could try to implement is to consider a similar setting to the one analysed above, but instead of removing a neuron and considering it as a target a posteriori, one could decide to consider a dynamic with $N+1$ neurons, where the additional $N^{th}+1$ neuron, call it y, immediately assumes this specific role, thus being inter and self-connected in a slightly different manner (the network will approximately be like in Figure 19).

The dynamic will be the following:

$$\begin{cases} S_i = sgn\left(\sum_{j\neq i} J_{ij}S_j + J_D S_i + W_i^0 y\right) \\ y = sgn\left(\sum_{i=1}^{N} w_i S_i + b\right) \end{cases} \tag{29}$$

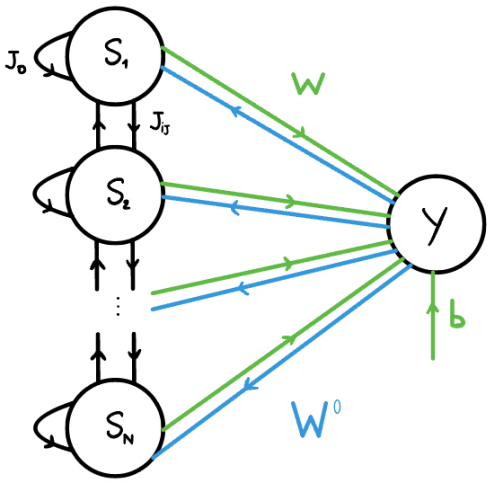


Figure 19: Visual intuitive representation of a single pattern of the newly described network, which will be the basis upon which the **Algorithm 2** will act.

where the additional parameters W_i^0 are quenched and chosen in the same way as the J_{ij} and denote the outgoing connections from y to the N original neurons; instead, differently from those, y is not self-connected and its incoming weights become learnable: we can consider $\bar{w}$ and b as weights and bias of a perceptron respectively, which tries to learn the internal representation and correctly classify the labels y^μ (one for each pattern μ) acting upon the successfully completion of a *pre-processing* of the data points through the dynamics in 29. The algorithm will be something like **Algorithm 2**.

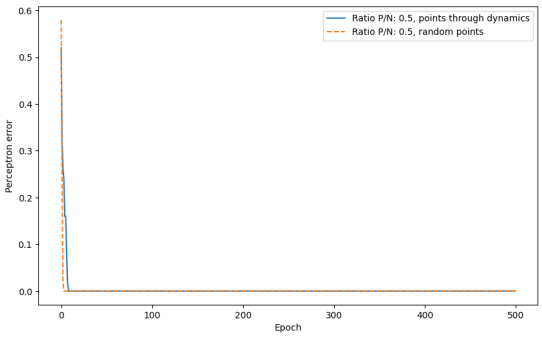
Through the first attempts of implementation, the *upgraded* perceptron seems to perform at least as good as the traditional one, with the difference that it is a bit slower for smaller ratios $\frac{P}{N}$, but shows a slight improvement for higher values (see Figure 20). Despite having significant room for improvement, I believe it has a huge potential for further research.

In the simplest model of (deep) neural networks we have an *input layer* of N neurons that represents the features of the input data, a *hidden layer* with pre-activations computed through a linear transformation of the input layer then passed through a non-linear activation function, and an *output layer* which might be the label of a classification task. The linear transformation is the most computationally expensive, since two nearby layers are fully connected through a matrix of weights (which are the parameters that

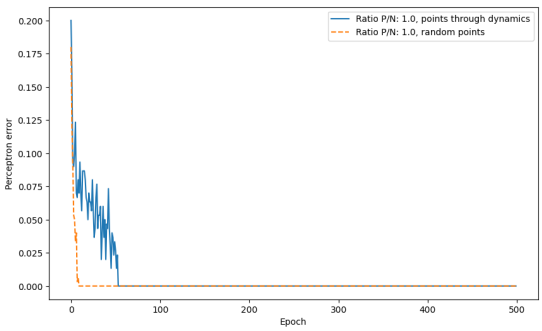
Algorithm 2 Post-dynamic perceptron algorithm

1: Initialize random couples $\left(\xi^{\mu}, y^{\mu}_{(\xi)}\right)_{\mu=1,...,P}$, where each ξ^{μ} is composed of N binary units S_i for $i = 1, ..., N$. These are our data points.
2: Fix the true labels you want the system to learn: $y^{\mu}_{\text{true}} = \{\pm 1\}$ for $\mu = 1, ..., P$, and a value of J_D for which the network is known to converge.
3: **while** stopping condition not met **do**
4: **for** each pattern μ **do**
5: Let the network $\left(\xi^{\mu}, y^{\mu}_{(\xi)}\right)$ evolve through 29. Doing so, we get a set $\left(\zeta^{\mu}, y^{\mu}_{(\zeta)}\right)$ of fixed points to which the dynamics converged.
6: **if** $y^{\mu}_{(\zeta)} \neq y^{\mu}_{\text{true}}$ **then**
7: $\bar{w} \leftarrow \bar{w} + y^{\mu}_{\text{true}} \zeta^{\mu}$
8: $b \leftarrow b + y^{\mu}_{\text{true}}$
9: **end if**
10: **end for**
11: **end while**

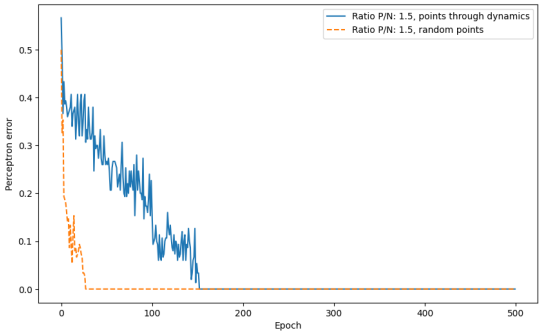
the network should learn) of order $O(N^2)$. If we interpret subsequent layers as states of the same system evolving in time, we recover the idea of the recurrent network above described, with the difference that the *learnable weights* now run in the order of $O(N)$. If, through an algorithm similar to the one above, we managed to push the boundary of the perceptron and be able to learn $P > 2N$ patterns, we would have an incredibly powerful system: it would have a potential closer to the one of the MLP, but way less computationally expensive. Without any doubt, this would be a huge step towards the goal of creating a more biological computational counterpart of the human brain.



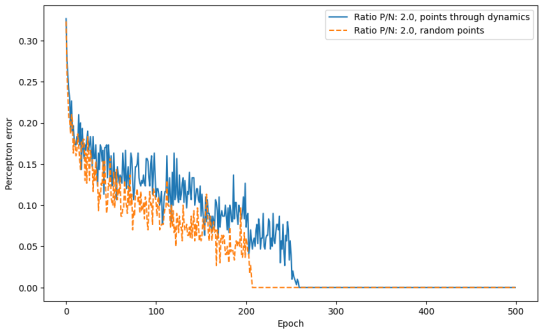
(a) $\frac{P}{N} = 0.5$



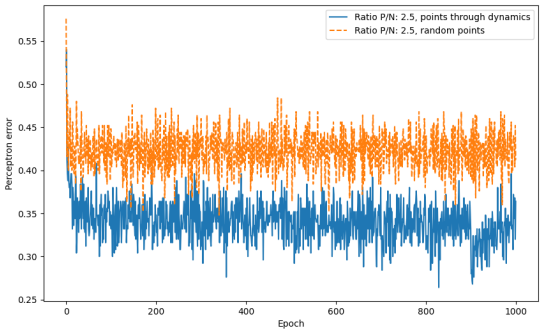
(b) $\frac{P}{N} = 1.0$



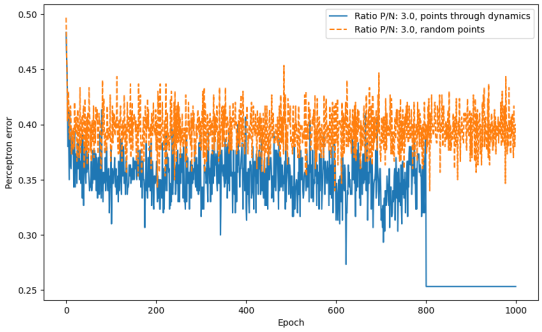
(c) $\frac{P}{N} = 1.5$



(d) $\frac{P}{N} = 2.0$



(e) $\frac{P}{N} = 2.5$



(f) $\frac{P}{N} = 3.0$

Figure 20: A first attempt in the implementation of **Algorithm 2**. Comparison of its performance with the one of the classical perceptron on the random initial points, for increasing values of the ratio $\frac{P}{N}$.

References

[1] Russell Stuart and Norvig Peter. Artificial intelligence: a modern approach, 1995.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 52:99–115, 1990.

[5] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[6] Teuvo Kohonen. *Associative memory: A system-theoretical approach*, volume 17. Springer Science & Business Media, 2012.

[7] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[8] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[9] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

[10] Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

[11] Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Spin-glass models of neural networks. *Physical Review A*, 32(2):1007, 1985.

[12] Mattia Scardecchia. Statistical physics of learning. *Replicating the work of professors Baldassi, Mezard, Zecchina remained unpublished as of today*, 2023.

[13] Harold L Atwood and Shanker Karunanithi. Diversification of synaptic strength: presynaptic elements. *Nature Reviews Neuroscience*, 3(7):497–516, 2002.

[14] Carolin Wichmann and Thomas Kuner. Heterogeneity of glutamatergic synapses: cellular mechanisms and network consequences. *Physiological reviews*, 102(1):269–318, 2022.

[15] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[16] Jonathan Kadmon and Haim Sompolinsky. Transition to chaos in random neuronal networks. *Physical Review X*, 5(4):041030, 2015.

[17] Richard B Stein. A theoretical analysis of neuronal variability. *Biophysical journal*, 5(2):173–194, 1965.

[18] Bruce W Knight. Dynamics of encoding in a population of neurons. *The Journal of general physiology*, 59(6):734–766, 1972.

[19] Merav Stern, Haim Sompolinsky, and Laurence F Abbott. Dynamics of random neural networks with bistable units. *Physical Review E*, 90(6):062710, 2014.

[20] Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. Chaos in random neural networks. *Physical review letters*, 61(3):259, 1988.

[21] Richard L Burden and J Douglas Faires. *Numerical analysis*. Brooks Cole, 1997.

[22] LF Abbott and Wade G Regehr. Synaptic computation. *Nature*, 431(7010):796–803, 2004.

[23] David G Clark and LF Abbott. Theory of coupled neuronal-synaptic dynamics. *Physical Review X*, 14(2):021001, 2024.

[24] Richard E Brown. Donald o. hebb and the organization of behavior: 17 years in the writing. *Molecular Brain*, 13(1):55, 2020.

[25] Anders Lansner, Florian Fiebig, and Pawel Herman. Hebbian fast plasticity and working memory. *arXiv preprint arXiv:2304.06626*, 2023.

[26] John Lisman. A mechanism for the hebb and the anti-hebb processes underlying learning and memory. *Proceedings of the National Academy of Sciences*, 86(23):9574–9578, 1989.

[27] Carl Van Vreeswijk and Haim Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996.

[28] Carl van Vreeswijk and Haim Sompolinsky. Chaotic balanced state in a model of cortical circuits. *Neural computation*, 10(6):1321–1371, 1998.

[29] Hans-Dieter Block. The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34(1):123, 1962.