Plagiarism Declaration:

1. I know that plagiarism is a serious form of academic dishonesty.
2. I have read the UCT document Avoiding Plagiarism: A guide for students, am familiar with its contents and have avoided all forms of plagiarism mentioned there.
3. Where I have used the words of others, I have indicated this by the use of quotation marks.
4. I have referenced all quotations and properly acknowledged other ideas borrowed from others.
5. I have not and shall not allow others to plagiarise my work.
6. I declare that this is my own work.

Signature: [illegible]

Hopfield Networks

Jeremy du Plessis
University of Cape Town
Department of Mathematics & Applied Mathematics

November 2020

Abstract

In 1982 John Hopfield proposed a computational model of associative memory in biological neural networks, now referred to as Hopfield networks, catalysing a resurgence of academic interest in artificial neural networks which persisted over the subsequent three decades. The model proposed by Hopfield, which is part of a now broader class of models called energy based models, is capable of both storing and retrieving patterns - functionality which is of great importance in the field of machine learning. In this research report we explore the evolution of the Hopfield model, examining the mathematical properties each of the major variants and confirming the theoretical results with computational simulations. Of utmost importance is the most recent work published on Hopfield networks which illustrates the mathematical equivalence between the update rule of modern Hopfield networks and the attention mechanism used in modern transformer models. We end the report by examining the results of some basic experiments which illustrate the properties and functionality of modern Hopfield networks applied to the MNIST data set, namely, pattern storage, retrieval and separation.

1 Early Hopfield Networks

1.1 Invention

In his now famous 1982 paper titled *"Neural Networks and Physical Systems With Emergent Collective Computational Abilities"* [1] John Hopfield proposed a simple model of associative memory in biological neural networks - the ability of the brain to access memories given only partial reference information - which he suggested could also be implemented as a hardware-based content addressable memory (CAM) system[1]. The simple model proposed by Hopfield in [1], now known as a Hopfield network, is part of a broader class of models called 'energy-based models' which derive their properties from a global energy function. In the paper, Hopfield proposed the idea of

[1]As opposed to random access memory (RAM), where in order to access data the OS provides the address where the data is stored, content addressable memory (CAM), also known as associative memory, executes search queries given the data itself and returns a memory address. CAM is typically implemented as a hardware-based search engine (it's the hardware analogue of an associative array or hash map in certain programming languages) and is used in situations where high-speed associative search is required, for example in routing tables.

’memories’ as local minima of the energy function governing the dynamics of the system defined by the collective properties of the model. The high-level idea was that memories, corresponding to certain states of the system, could be retrieved at a later time by beginning in a similar state (i.e. given partial information) and evolving the system according to an update rule so as to move ’downhill’ in energy until the desired state (memory) corresponding to an energy minima is reached[2].

Formally, the associative memory model described in [1] is defined by a set of d artificial neurons, termed *binary threshold units*, where the activation of each neuron may take on a value of -1 (not firing) or 1 (firing at maximum rate) at any given time[3]. We will represent the state of the network at some time t as as $\boldsymbol{\xi}^t \in \{-1, 1\}^d$. A network is defined by it’s adjacency/weight matrix $W \in \mathbb{R}^{d\times d}$, where the $(i, j)^{th}$ entry, w_{ij}, represents the strength of the connection between neurons i and j, and a threshold vector $\mathbf{b} \in \mathbb{R}^d$, where b_i is the threshold value for the i^{th} neuron. At any time t we can compute the incoming stimulus received by the i^{th} neuron as the weighted sum of the activations of all of the neurons adjacent to it in the network, minus its threshold value:

$$H_i(t) = \sum_{j=1}^{d} w_{i,j}\xi_j^t - b_i \tag{1}$$

where ξ_j^t is the activation value of the j^{th} neuron in the network at time t. At each time t, the activation value for each neuron i, $i = 1, \ldots, d$, in the network is updated according to the *binary threshold update rule*[4]:

$$\xi_i^{t+1} = \mathbf{sgn}(H_i(t)) \tag{2}$$

where the **sgn** function assigns $+1$ to input values which are positive or zero, and -1 to input values which are negative. The updates to each individual neuron may happen at some random rate, but in practice updates to each neuron occur one at a time in a random sequence. Importantly, we note that updates may also happen in parallel, according to:

$$\boldsymbol{\xi}^{t+1} = \mathbf{sgn}(W\boldsymbol{\xi}^t - \mathbf{b}) \tag{3}$$

By definition of the update rule (1) the early Hopfield network may be thought of as a special type of recurrent neural network (RNN) - see Figure 1-A for an illustration - although it differs from modern RNNs (e.g. long short-tern recurrent neural networks) in two significant ways. Firstly, the network only takes external input initially (the starting state), where for each subsequent iteration the network only performs updates based on it’s current state. Secondly, network updates are typically not performed synchronously in practice because synchronous updates can lead to periodic trajectories in the phase space - more on this below[5].

Hopfield goes on to show in [1] that in the special case where W is symmetric, that is, $w_{ij} = w_{j,i}$ for all $i, j = 1, 2, \ldots, d$, which is not the case in most well-known RNNs, then the state of the system could be described

[2]Together with the backpropogation algorithm, Hopfield networks were one of the main reasons for the resurgence of academic interest in artificial neural networks in the 1980’s, mainly because Hopfield, being a reputable physicist, described his model in terms of the dynamics of physical systems.

[3]The original Hopfield paper [1] uses binary values $\{0, 1\}$.

[4]hence the term ’binary threshold units’ used to describe theses particular artificial neurons.

[5]Additionally, there is no evidence to suggest that the firing of biological neurons in the brain is synchronised [1].

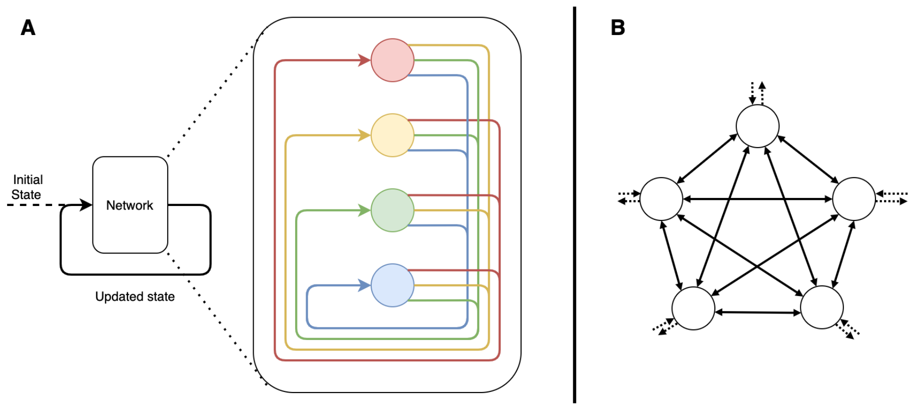

Figure 1: An illustration of a Hopfield network. **A** Illustrates the recurrent nature of the Hopfield network as the update rule is applied, and **B** illustrates a Hopfield network with symmetric weights/bi-directional edges connecting the artificial neurons.

in terms of a global energy function which decreases monotonically as the update rule defined by equation (2) is applied recurrently and **asynchronously** to the neural activations of the network. See Figure 1-B for an illustration of a Hopfield network with symmetric weights. The energy function proposed by Hopfield has the form[6]:

$$E(\boldsymbol{\xi}) = -\frac{1}{2}\boldsymbol{\xi}^T\mathbf{W}\boldsymbol{\xi} + \boldsymbol{\xi}^T\mathbf{b} \tag{4}$$

$$= -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{i,j}\xi_i\xi_j + \sum_{i}^{d}\xi_i b_i \tag{5}$$

We will go on to show in a section below that if we begin in an an arbitrary state, then update the activation of each neuron in the network one by one according to the binary threshold update rule (2), that the resulting sequence of states will correspond to a sequence of energy values which decreases monotonically to an energy minima. Furthermore we will show that, given W has certain properties, these energy minima necessarily correspond to stable states, or fixed points, in phase space, defined as $\boldsymbol{\xi}^{t+1} = \boldsymbol{\xi}^t$, which in Hopfield's language are 'memories' stored by the network. As an example, Figure 2 shows an illustration of the binary threshold rule applied to updating the activation of a single neuron in a simple Hopfield network where the threshold values are zero for each neuron, causing the state of the system to stabilise at a local energy minimum, making the binary pattern represented by the neural activations (on the right) a stored state, or memory, of the network. The last major component of the Hopfield model to consider is the discuss is the process encoding/storing binary patterns. Hopfield proposed a simple, online rule for setting the weights of the network in such a way as to memorise a set of N binary patterns $\{\mathbf{x}_i\}_{i=1}^N$, where each $\mathbf{x}_i$ is a binary vector of length d, which we will express in matrix form as $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, which has the dimensions $N \times d$. The online rule proposed by Hopfield for setting the weights in W is simply the sum of the outer products of each pattern:

$$W = \mathbf{X}\mathbf{X}^T = \sum_{k=1}^{N}\mathbf{x}_k\mathbf{x}_k^T \tag{6}$$

Or, equivalently, for a single entry in the weight matrix we can write:

[6]However, Hopfield neglects to include the bias terms in his original paper [1].

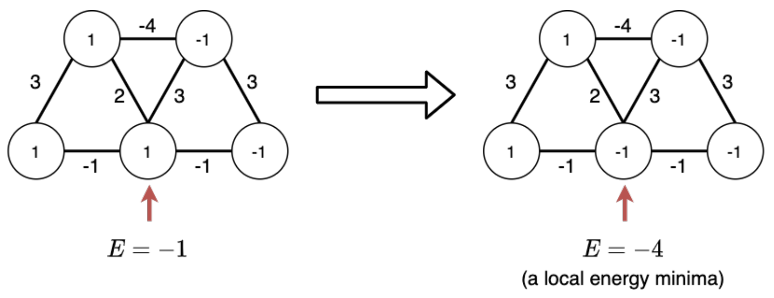

Figure 2: An example of the binary threshold update rule applied to a Hopfield network, causing it to settle at a local minima of the energy function.

$$w_{ij} = \sum_{k=1}^{N} x_i^k x_j^k \tag{7}$$

Where x_i^k is the i^{th} element of the k^{th} pattern vector. Equation (7), known as the Hebbian learning rule [7], gives us a symmetric weight matrix W - this will be important in proving the convergence of the network later on. To see that, on average, a stored pattern corresponds to a local energy minima, Hopfield makes the following argument in [1]. Suppose we want to store N randomly sampled patterns $\{\mathbf{x}\}_{i=1}^{N}$, where each bit in each pattern is either $+1$ or -1 with equal probability. Then, suppose the network is in a state equivalent to the k'^{th} stored pattern. We would then have the incoming stimulus to neuron i being:

$$H_i^{k'} = \sum_{j=1}^{d} w_{ij} x_j^{k'} = \sum_{k=1}^{N} x_i^k \left[\sum_{j=1}^{d} x_j^{k'} x_j^k \right] = x_i^{k'} d + \sum_{k \neq k'} x_i^k \left[\sum_{j=1}^{d} x_j^{k'} x_j^k \right]$$

We observe that the $k = k'$ term gives a constant signal, d, which is the expectation of the quantity $H_i^{k'}$, while the expectation of contribution coming from the $k \neq k'$ terms (the so-called noise terms) is essentially 0, since the patterns are randomly sampled. We therefore get:

$$\langle H_i^{k'} \rangle \equiv x_i^{k'} d \approx \sum_{j=1}^{d} w_{ij} x_j^{k'}$$

which is clearly positive when $x_i^{k'} = 1$ and negative when $x_i^{k'} = -1$, which means that if the state of the network is equivalent to pattern k', and the bias threshold is zero, then on average the state should be stable. Based on this, one would intuitively guess that the stability of stored patterns depends on a number of factors including the number of states one attempts to store in the network, as well as the correlation between states - we will explore this in detail in the sections below.

To illustrate the theory described above we implement a classic Hopfield net and demonstrate the storage and retrieval of black and white images of the Simpsons [8] - see Figure 3 for a sample of the results. The code for the traditional Hopfield net may be found in Appendix C. The energy corresponding to the updates of the model

[7]Introduced by Donald Hebb in his 1949 book The *Organization of Behavior* the Hebbian learning rule, often summarised by the colloquial adage *"Neurons that fire together, wire together"*, states in essence that the strength of the connection between two neurons should be proportional to the product of their activations.

[8]I am reproducing the examples from the blog `ml-jku.github.io/hopfield-layers/`, however, I wrote all code and processed all the images independently.

which produced the results in Figure 3 may be seen in Figure 4 - clearly the stored pattern (Homer) corresponds to an energy minima.

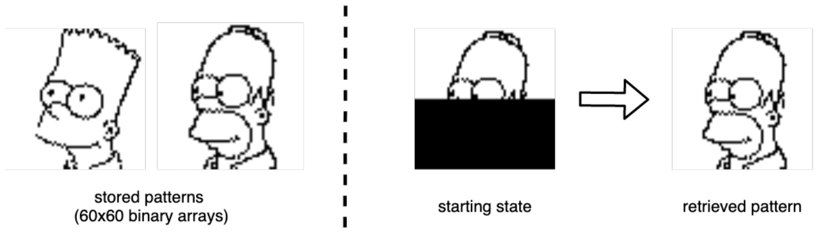

Figure 3: An example of a traditional Hopfield network being trained to memorise and retrieve a binary pattern given only partial information.

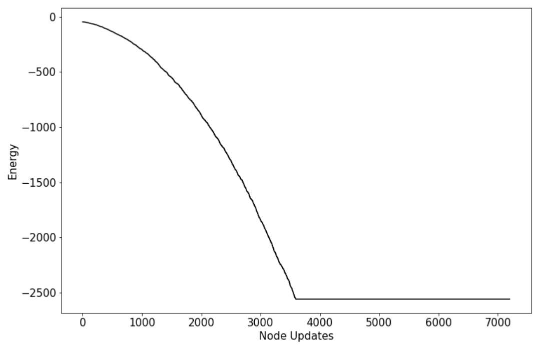

Figure 4: A plot illustrating the decrease in energy when applying the update rule to nodes in a traditional Hopfield network in order to the retrieve the Homer pattern seen in Figure 3

The dynamics of the system arising from the traditional Hopfield model described above are stochastic in nature and depend significantly on a few key elements: the manner in which the neurons of the network are updated, the properties of the weight matrix, the number of patterns stored, the degree of correlation between stored patterns, and the degree to which starting states differ from the stored states which one wishes to retrieve. As we will see, the traditional Hopfield model has very limited storage capacity and is susceptible to large retrieval error rates when the stored patterns are highly correlated. First, we go on to prove some convergence properties of the traditional Hopfield model, whereafter we will examine it's storage capacity, showing how it is highly dependant on the degree of correlations between patterns.

1.2 Convergence Properties

A few key convergence properties of the traditional Hopfield model have been proved over the years ([1], [5], [4]), the most important of which may be summarised as follows. If $f = (W, \mathbf{b})$ is a Hopfield network then:

1. If the neural activations of f are updated asynchronously, and if W is symmetric with non-negative diagonal elements, then f will always converge to a stable state.

2. If the neural activations of f are updated in parallel (i.e. synchronous updates), and if W is symmetric, then f will converge to either a stable state, or a limit cycle of length 2.

3. If the neural activations of f are updated in parallel (i.e. synchronous updates), with W being anti-symmetric and $\mathbf{b} = \mathbf{0}$, then f will always converge to a limit cycle of length 4.

We will prove each of these results in turn, but first it is helpful to have them illustrated with three simple examples. Consider two simple Hopfield networks, f_1 and f_2, defined by the weight matrices W_1, W_2 below, each having zero-valued bias vectors.

$$\hat{W}_1 = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \quad \hat{W}_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Let M_{f_i} denote the set of stable states of some Hopfield network f_i. Then, noticing that W_1 is symmetric, it can be shown that if the neural activations of f_1 are updated in an asynchronous manner, then the network will always converge to a a stable state in $M_{f_1} = \{(-1,1),(1,-1)\}$. On the other hand, if the neural activations of f_1 are updated in a parallel manner, according to equation (3), it can be shown that the state of the network will either converge to a state in M_{f_1} or to a limit cycle of length 2, where the state oscillates between states in the set $\{(1,1),(-1,-1)\}$. These two cases correspond to convergence properties (1) and (2) above. Then, noticing that W_2 is an anti-symmetric matrix, it can be shown that if the neural activations of f_2 are updated in a parallel manner, $M_{f_2} = \emptyset$ and that the state of the network will oscillate between the states in $\{(1,1),(-1,1),(-1,-1),(1,-1)\}$, that is, a limit cycle of length 4. This corresponds to property (3) above. We now go on to formally prove the first two convergence properties listed above. The following theorem is adapted from [1] and [4].

Theorem 1. *Let W be a symmetric matrix with non-negative diagonal elements and $\mathbf{b}$ be a threshold vector defining an traditional Hopfield network $f = (W, \mathbf{b})$. If the neural activations of f are updated asynchronously, the sequence of energy values corresponding to the resulting sequence of states will decrease monotonically. Furthermore, since the energy of f is bounded from below, the sequence will converge and the network will reach a stable state (i.e. there are no limit cycles in the phase space).*

Proof. Let $\Delta E = E(\boldsymbol{\xi}^{t+1}) - E(\boldsymbol{\xi}^t)$ be the difference in energy between the state of the network at time $t+1$ and time t, and let $\Delta\xi_m$ denote the difference in the activation value of the m^{th} neuron after applying the binary threshold rule (2) at time t, then we have that:

$$\Delta\xi_m = \begin{cases} 0, & \text{if } \xi_m^t = \mathbf{sgn}(H_m(t)) \\ -2, & \text{if } \xi_m^t = 1 \text{ and } \mathbf{sgn}(H_m(t)) = -1 \\ 2, & \text{if } \xi_m^t = -1 \text{ and } \mathbf{sgn}(H_m(t)) = 1 \end{cases} \tag{8}$$

Now, by assumption the neurons of the network are updated in an asynchronous manner (i.e. one neuron at a time in random order). Suppose an arbitrary neuron m is updated at time t, from ξ_m to ξ'_m, then $\Delta\xi_m = \xi'_m - \xi_m$ and the resulting energy difference would be:

$$\Delta E = -\frac{1}{2}\left[\sum_{\substack{i=1\\i\neq m}}^{d} w_{i,m}\xi_i\xi'_m - \sum_{\substack{j=1\\i\neq m}}^{d} w_{m,j}\xi_m\xi_j + w_{m,m}(\xi'^2_m - \xi^2_m)\right] + \Delta\xi_m b_m \tag{9}$$

$$= -\frac{1}{2}\left[2\Delta\xi_m \sum_{\substack{i=1\\i\neq m}}^{d} w_{i,m}\xi_i + w_{m,m}(\xi_m'^2 - \xi_m^2)\right] + \Delta\xi_m b_m \quad \text{(since } W \text{ is symmetric)} \tag{10}$$

$$= -\frac{1}{2}\left[2\Delta\xi_m \sum_{i=1}^{d} w_{i,m}\xi_i + w_{m,m}(\xi_m'^2 - \xi_m^2) - 2\Delta\xi_m w_{m,m}\xi_m\right] + \Delta\xi_m b_m \tag{11}$$

$$= -\frac{1}{2}\left[2\Delta\xi_m \sum_{i=1}^{d} w_{i,m}\xi_i + w_{m,m}(\xi_m'^2 - 2\xi_m\xi_m' + \xi_m^2)\right] + \Delta\xi_m b_m \tag{12}$$

$$= -\Delta\xi_m H_m - \frac{1}{2}w_{m,m}\Delta\xi_m^2 \tag{13}$$

Therefore, since $\Delta\xi_m H_m \geq 0$, by equation (8), and since $w_{m,m} \geq 0$, by assumption, we have shown that $\Delta E \leq 0$ for every m. Hence, the sequence of states resulting from the recurrent application of the binary threshold update rule (2) results in an energy sequence which is monotonically decreasing. Finally, it is easy to see that since the values in W and $\mathbf{b}$ are finite, that the energy of f is bounded from below, which means the sequence of energy values will converge on a minimum. This in turn implies that the network will reach a stable state, since no update to any neuron will occur if it causes an increase in energy. This completes the proof of convergence property (1). $\square$

Having now shown that any Hopfield network defined by a symmetric adjacency matrix with non-negative diagonal entries will converge to a stable state if updated in an asynchronous manner, we would like to better understand how updating the neurons in parallel leads to a limit cycle of length 2. To this end, we adapt an elegant proof presented in [4]. But, first, we need to prove an intermediate result, showing that performing parallel updates to a network of the type described above is equivalent to performing asynchronous updates in a larger network with similar properties.

Lemma 2. *Let W be an arbitrary symmetric weight matrix and $\mathbf{b}$ be a threshold vector defining an traditional Hopfield network $f = (W, \mathbf{b})$, with f having d neurons. Let $\hat{f} = (\hat{W}, \hat{\mathbf{b}})$ be a new Hopfield network derived from f, with:*

$$\hat{W} = \begin{pmatrix} 0 & W \\ W & 0 \end{pmatrix} \qquad \hat{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ \mathbf{b} \end{pmatrix}$$

Then, there exists an ordering in which the neural activations of $\hat{f}$ may be updated asynchronously which is equivalent to updating the neural activations of f in parallel (as per equation (3)).

Proof. We first begin by noting that $\hat{f}$ is a bipartite graph with $2d$ neurons, and that $\hat{W}$ is a symmetric matrix having zero-valued diagonal elements. Let S_1 and S_2 be the partite sets of $\hat{f}$, containing the neurons $1, \ldots, d$ and $d+1, \ldots, 2d$ respectively. It is clear from the structure of $\hat{W}$ that no two nodes of S_1 are connected to one another, similarly for S_2. Notice also that the i^{th} neuron in S_1 is incident with a set of edges which have weights exactly equivalent to those of the edges incident with the $(d+i)^{th}$ neuron in S_2. Now, suppose that the network f is in some starting state $\boldsymbol{\xi}^0$ and that $\hat{f}$ is in the starting state $(\boldsymbol{\xi}^0, \boldsymbol{\xi}^0)$, that is, where neurons $1, \ldots, d$ have exactly the same activation pattern as neurons $d+1, \ldots, 2d$. Then, suppose we update the neurons of $\hat{f}$ in the following way. Begin by updating all the neurons in S_1 one by one (in any order), whereafter $\hat{f}$ will be in the state $(\boldsymbol{\xi}^1, \boldsymbol{\xi}^0)$. Then, since each neuron in S_1 is adjacent only to neurons in S_2, updating S_1 in this manner is equivalent to updating all the neurons in f in parallel, whereafter f would be in the state $\boldsymbol{\xi}^1$. After updating the neurons in S_1, we may update all the neurons in S_2 (in any order), after which $\hat{f}$ will be in the state $(\boldsymbol{\xi}^1, \boldsymbol{\xi}^2)$. Again, this is equivalent to performing a second parallel update of the neurons of f, whereafter f would be in the state $\boldsymbol{\xi}^2$. In this way, updating $\hat{f}$ in an asynchronous manner is equivalent to updating f in a fully parallel manner. □

We now go on to state and prove a theorem showing that property (2) above holds.

Theorem 3. *Let W be an arbitrary symmetric weight matrix and $\mathbf{b}$ be a threshold vector defining a Hopfield network $f = (W, \mathbf{b})$. If the neural activations of f are updated in parallel, according to equation (3), then the f will either converge to a stable state, or limit cycle of length 2 (i.e. oscillating between two states continuously).*

Proof. By Lemma 2 there we can construct a network $\hat{f}$ from f such that updating the neural activations of $\hat{f} = (\hat{W}, \hat{\mathbf{b}})$ in an asynchronous manner is equivalent to updating the neural activations of f in parallel. Since $\hat{W}$ is symmetric, having non-negative diagonal entries, by Theorem 1 we know that $\hat{f}$ will converge to a stable state $(\boldsymbol{\xi}^{1*}, \boldsymbol{\xi}^{2*})$ corresponding to the two partite sets of neurons S_1 and S_2. When $\hat{f}$ reaches a stable state, there are two cases:

1. $\boldsymbol{\xi}^{1*}$ is element-wise equivalent to $\boldsymbol{\xi}^{2*}$, in which case f will converge to a stable state which is element-wise equivalent to $\boldsymbol{\xi}^{1*}$.

2. $\boldsymbol{\xi}^{1*}$ and $\boldsymbol{\xi}^{2*}$ are distinct states, in which case the state of f will oscillate between $\boldsymbol{\xi}^{1*}$ and $\boldsymbol{\xi}^{2*}$. That is, f will converge to a limit cycle of length 2.

This completes the proof of convergence property (2). □

The proof of convergence property (3), which is similar to that for convergence property (2), will not be covered here, since in both practice and research most Hopfield networks are typically studied with symmetric weight matrices, but may be seen in [4]. We will now go on to examine the storage capacity of traditional Hopfield networks.

1.3 Storage Capacity

Much analysis has been performed on the dynamics of the traditional Hopfield model in order to asses its critical storage capacity ([3], [6], [10]). The storage capacity of a Hopfield network is measured as the ratio between the number of binary patterns stored, N, and the number of neurons in the network (or equivalently, the pattern length), d, combined into a single metric called the *load parameter*: $\lambda = \frac{N}{d}$. Amit et. al were the first to perform an in depth analysis of the storage capacity of the Hopfield model in 1986 [3], where they added several layers of complexity to the simple model described above in order to model a spin system (we discuss the analogy of the Hopfield Network to the Ising model below), which they then analysed both analytically and numerically using mean field theory (and other tools from statistical mechanics). The results of the analysis conducted and published in [3] and [10] determined that the storage capacity of a traditional Hopfield network for a retrieval of patterns free from errors was:

$$N_c \approx \frac{d}{2ln(d)}$$

And the storage capacity for retrieval of patterns with a small percentage of errors ($\approx 1.5\% - 5\%$) was determined to be:

$$N_c \approx 0.14d$$

Where N_c, measured in number of patterns stored, represents the critical storage capacity of the network. These analytical results have been confirmed by way of a Monte-Carlo simulation, and the results published in several papers including the original paper by John Hopfield [1], as well as in [6] which builds on the work presented in [3]. Figures 4 and 5 illustrate the results of a Monte-Carlo simulation replicating the results shown in [6] which confirm the theoretical results regarding storage capacity.

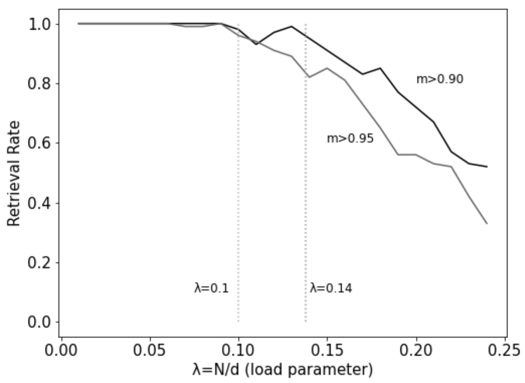

Figure 5: Results from a Monte-Carlo simulation of the traditional Hopfield model. Here the pattern retrieval rate is plotted against the load parameter $\lambda = N/d$ for different values of the overlap threshold $m > m^*$, where $d = 100$ and $\rho = 0.15$.

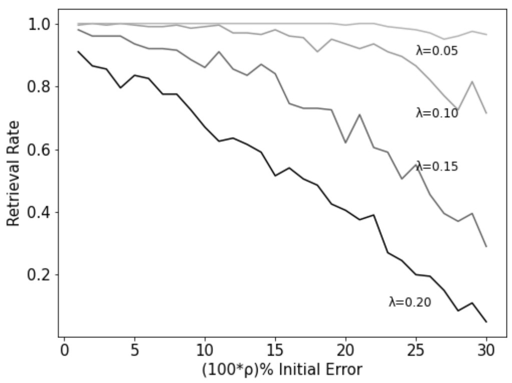

Figure 6: Results from a Monte-Carlo simulation of the traditional Hopfield model. Here the retrieval rate is plotted against the $(100{\cdot}\rho)\%$ initial error (the difference between the starting state and the target state) for different values of the load parameter λ.

The simulation was conducted as follows. We would like to simulate the storage and retrieval of randomly generated binary patterns in a traditional Hopfield network. In order to do this we first need to define what a successful retrieval means, and thus how we measure retrieval error. Suppose we generate N random, distinct binary patterns and store them in the matrix X (as defined above), noting that each of the binary variables in each pattern is selected uniformally from $\{-1, 1\}$. The we select a pattern $\mathbf{x}_0$ as the target pattern, and select ρd elements at random in the target pattern to invert, where $\rho \in [0, 1]$ determines the % initial error, multiplying each by -1 to create a new, altered pattern $\tilde{\mathbf{x}}_0$. We then assign the altered pattern to be the starting state of the system: $\boldsymbol{\xi}^0 = \tilde{\mathbf{x}}_0$. We then recursively apply the asynchronous update rule, updating the nodes of the Hopfield network until the state converges on a fixed point $\boldsymbol{\xi}^*$. We then compute the overlap $m = \frac{1}{d}\mathbf{x}_0^T\boldsymbol{\xi}^*$ to determine the element-wise similarity between the target state and the stable state upon which the network has converged. A retrieval is considered successful when we have $m \geq m^*$, where m^* is an overlap threshold. We then repeat this procedure a large number of times and record the proportion of successful retrievals for the parameters λ, ρ and m^*. The code for the Monte-Carlo simulation can be found in Appendix C.

The results potted in Figure 5 illustrate the retrieval rate as a function of the load parameter λ for a fixed initial error of 15% ($\rho = 0.15$), where two curves have been plotted, one for $m^* = 0.95$ and another for $m^* = 0.9$.

The vertical dotted lines correspond to $\lambda = 0.1 \approx \frac{d}{2log_2(d)d}$, where $d = 100$, the storage capacity threshold for error-free retrieval (left), and $\lambda = 0.14$, the storage capacity threshold for a small percentage error (right). The results of the simulation reinforce the theoretical results; the retrieval rate of the network is ≈ 1.0 for $\lambda \leq 0.1$, decreasing dramatically for $\lambda > 0.14$. Additionally, performance appears to degrade at a faster rate for $m^* = 0.95$ than for $m^* = 0.9$ as λ approaches 0.25. Figure 6 illustrates the retrieval rate plotted against the $(100 \cdot \rho)\%$ initial error for different values of λ. We see from the plotted curves that network exhibits nearly perfect retrieval for $\lambda = 0.05$ up to and above an initial error of 25%, with performance decreasing dramatically for $\lambda > 0.1$. As one would expect, it appears that the effects of increasing λ as well as the $(100 \cdot \rho)\%$ initial error compound to cause an even greater, possibly non-linear, decrease in retrieval rate.

In order to understand the inefficiency of the traditional Hopfield model with respect to storage capacity it is helpful to compare the amount of memory it takes to store the weights and biases of a Hopfield network with d neurons, having a critical storage capacity of $N_c \approx 0.14d$, versus the amount of memory required to store N_c binary patterns in ordinary computer memory. First, notice that there are d neurons and each neuron is incident with at most d weighted edges, along with a total of d biases, giving $\mathcal{O}(d^2)$ weights and biases. Then, notice that if we would like to store N patterns, setting the weights of the network according to equation (7), each weight will lie in the range $[-N, N]$, which is a total of $2N + 1$ possible numbers. Assuming the biases also fall in that range the total memory required to store the weights are biases of the network will be $\mathcal{O}(d^2 log_2(2N + 1))$ bits. On the other hand, the amount of memory required to store $0.14d \approx N_c$ binary patterns, at d bits per pattern, is $0.14d^2$ bits. The important thing to notice here is that the number of bits required to store the weights and biases scales logarithmically with N, the number of patterns stored, whereas the amount of information being stored does not.

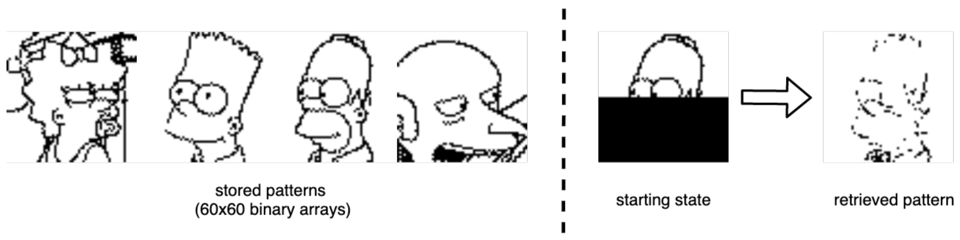

Figure 7: Pattern retrieval error in a Hopfield network.

As mentioned above, another factor which can have large adverse effects on the critical storage capacity of a Hopfield network is the degree to which patterns are correlated with one another [16]. As an example, consider the Hopfield network trained to store and retrieve the Simpsons images, illustrated in Figure 7. Notice that, in this example, the size of the network is $d = 60 \times 60 = 3600$, and so based on the preceding analysis the critical storage capacity should be $N_c \approx 0.14d = 504$ patterns. However, as illustrated in Figure 7, the network fails to retrieve the target image of homer, given half-masked image as the initial state. Instead the network retrieves what appears to be a muddled combination of the patterns stored in the network. The reason for this is because the stored patterns, each containing a illustration of a face, are highly correlated whereas the theoretical storage capacity was computed for random binary patterns. The muddled image retrieved by the network is still a local minima of the energy function, however it is what is termed a 'spurious minima', which occurs when two nearby energy minima, which may be thought of as fixed points in a dynamical system, coalesce to form a new minima which does not correspond to any single stored pattern, but rather something representing a distorted combination of stored patterns. We will see below that this problem may be remedied by the careful reconfiguration of the energy function, the storage procedure and the update rule.

A final note regarding the storage capacity of the traditional Hopfield model is that new research published in 2017 [7] showed that so long as the diagonal elements of the weight matrix are allowed to be non-zero, a number of patterns far exceeding the number of neurons ($N_c >> d$) may be stored as fixed points of the system. However, a paper [8] was published shortly after in response showing that, while the network is able to "store" a much larger number of patterns than previously thought possible, it is not a particularly useful finding since the attraction basins surrounding each fixed point are essentially non-existent, so much so that instabilities were observed for perturbations as small as a single bit. The authors of [9] published a second paper in response in 2019, but by that time John Hopfield, along with a younger researcher named Dmitry Krotov, had published a new and improved model for associative memory which appeared to bring large improvements to the storage capacity without sacrificing the networks ability to successfully retrieve patterns. We will examine Hopfield and Krotov's new contribution in detail below.

1.4 An Analogy To The Classic Ising Model

Aside from being a model for associative memory in the context of biology, or of content-addressable memory in computer science, the traditional Hopfield model has been studied extensively in the context of physics as a simple Ising model [3] [6] [5]. The Ising model is used to model the emergent behaviour that arises from a system involving lots of very small interacting components, abstractly termed "spins". A spin may at any time be in one of two "micro-states", -1 or +1. A spin system may be pictured as a lattice (as opposed to a graph) where at each site on the lattice (as opposed to a vertex) there is an arrow which may point either up (+1) or down (-1) - see Figure 8-B for an illustration. The spin system has an associated energy function, called the Hamiltonian, which computes the total energy of the system as a function of several different components. First and foremost is the collective state of the individual spins, which may be represented entirely by a binary vector, as per the Hopfield model, where energy between two adjacent spins is lowest when they are pointing in opposite directions (i.e. +1 * -1 = -1). Furthermore, each pair of adjacent spins has an associated coupling strength (as opposed to a weighted edge or synaptic connection) which dictates the degree to which the interaction between each pair of spins contributes to the overall energy. Finally, each spin is influenced by an external field (as opposed to a bias or threshold value) which also contributes to the total energy of the system. As per the Hopfield model, the objective is to study the emergent dynamics of the system under different coupling strengths and external fields. An elementary illustrative example may be seen in Figure 8-A where the spins are illustrated as small dipole magnets in the presence of an external magnetic field, and where the coupling strength between adjacent magnets may be controlled. If left to rotate freely, the magnets will always settle in a configuration which corresponds to an energy minima of the system.

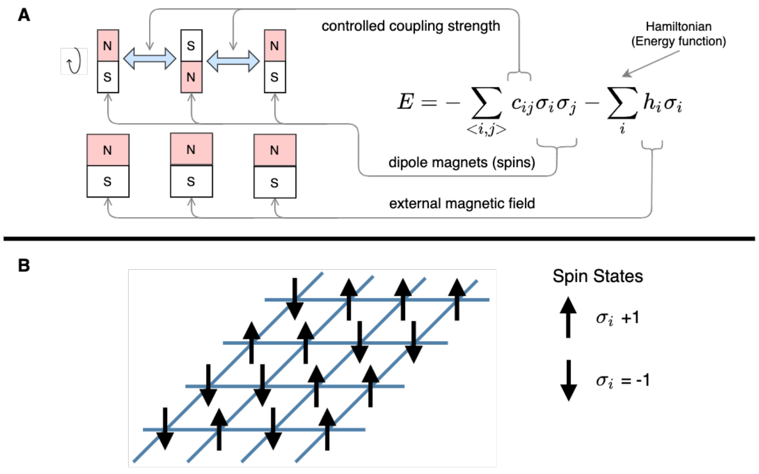
A
controlled coupling strength
Hamiltonian (Energy function)
$E = -\sum_{<i,j>} c_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i$
dipole magnets (spins)
external magnetic field
N
S
B
Spin States
σ_i +1
σ_i = -1

Figure 8: The classic Ising model (an analogy to the Hopfield model) illustrated. **A** A simple system of dipole magnets (spins) where each magnet is free to rotate, having either north pole up (+1) or south pole up (-1). Additionally each magnet is in the presence of a (often much stronger) magnetic field, the larger magnets, and the coupling strength between magnets is controlled by a barrier between each pair. The small magnets will always attempt to arange themselves in a low energy state - a local minima of the Hamiltonian. **B** A theoretical spin system illustrated as a lattice, where the arrows at the vertices of the lattice represent the spins. Spins may point up (+1) or down (-1).

2 Modern Hopfield Networks

2.1 Dense Associative Memory

In a 2016 paper titled "*Dense Associative Memory for Pattern Recognition*" [11] John Hopfield, together with a younger researcher named Dmitry Krotov, proposed a new-and-improved Hopfield model which the authors refer to as a *dense associative memory* model. The major contribution of the paper was to show that, as opposed to the linear critical storage capacity of the original Hopfield model, the dense associative memory model has a critical storage capacity which is polynomial in the size of the network - a major improvement. We will now go on to examine the formal definition of the dense associative memory model proposed in [11].

As opposed to the original Hopfield model, the modern Hopfield model proposed in [11] no longer involved a weight matrix, but is rather based entirely on an energy function of the form:

$$E(\boldsymbol{\xi}) = -\sum_{k=1}^{N} F(\mathbf{x}_k^T \boldsymbol{\xi}) \tag{14}$$

Where $F(x)$, referred to as the *interaction function*, is some smooth function which defines the energy "surface" in the $d-$dimensional state space of the network. In the original Hopfield model the interaction function takes the form $F(x) = x^2$, which allows only for second order interactions between neurons. The authors of [11] observe that the severe increase in retrieval errors which occurs in the original model when one attempts to store $N > 0.14d$ patterns arises because the quadratic form of the interaction function results in the energy changing "too slowly" as the state of the network approaches a point phase space corresponding to a stored memory. The authors then suggest that a polynomial interaction function of the form $F(x) = x^n$ with $n > 2$, allowing for higher order interactions, would result in an energy "surface" [9] wherein each of the stored patterns is more likely to be associated with a deep, convex energy minima about which the rate of change of energy is more extreme. This makes intuitive sense when one considers the input to the interaction function, as seen in equation (14), which is simply the inner product between the network state $\boldsymbol{\xi}$ and a stored pattern $\mathbf{x}_i$, which is a measure of their similarity. In the case that the patterns $\{\mathbf{x}\}_{i=1}^N$ are randomly distributed, if the state of the network $\boldsymbol{\xi}$ is in close proximity to some stored pattern $\mathbf{x}_k$ in state space, the term $F(\mathbf{x}_k^T\boldsymbol{\xi})$ will dominate the energy (14), where the strength of the signal may be increased by increasing the degree n of the polynomial interaction function. Notice that a difference of 1 bit between $\boldsymbol{\xi}$ and $\mathbf{x}_k$ corresponds to a difference of 2 in the inner product, which will in turn be raised to the power of n inside the interaction function. The result of having a high-order polynomial interaction function will mean a faster rate of change in energy and a wider basin of attraction in the regions of the state space about the points which correspond to the stored patterns.

Having defined the energy function, the authors of [11] go on to define a rule for updating individual neurons according to a transition function:

$$\xi_i^{(t+1)} = T_i(\boldsymbol{\xi}^{(t)})$$

The transition function, $T_i(\boldsymbol{\xi})$, makes explicit use of the energy function such that the updated value of the i^{th} bit, ξ_i, clearly corresponds to a decrease in energy. This is accomplished by computing $\mathbf{sgn}()$ of the difference between the energy of the network for $\xi_i = 1$ and $\xi_i = -1$:

$$T_i(\boldsymbol{\xi}^{(t)}) = \mathbf{sgn}\left[\sum_{k=1}^{N}\left(F\left(x_i^k + \sum_{j\neq i} x_j^k \xi_j^{(t)}\right) - F\left(-x_i^k + \sum_{j\neq i} x_j^k \xi_j^{(t)}\right)\right)\right] \tag{15}$$

Now we will examine the critical storage capacity of the model proposed in [11], where the interaction function has a polynomial form, $F(x) = x^n$. What we would like to establish statistically is the number of patterns we can store in such a model before the stored patterns (or rather, the states corresponding to the stored patterns) are at significant risk of becoming unstable points in phase space. To answer this question, consider the case where we randomly sample N binary patterns $\{\mathbf{x}_k\}_{k=1}^N$ such that each bit in each pattern is either $+1$ or -1 with equal probability. Now, suppose the network is in a state $\boldsymbol{\xi}^{(t)} = \mathbf{x}_{k'}$, corresponding to the k'^{th} stored pattern, and consider the change in the energy of the network if the i^{th} bit were flipped (multiplied by -1):

$$\Delta_i E = \sum_{k=1}^{N}\left((x_i^k x_i^{k'} + \sum_{j\neq i} x_j^k x_j^{k'})^n - (-x_i^k x_i^{k'} + \sum_{j\neq i} x_j^k x_j^{k'})^n\right) \tag{16}$$

$$= d^n - (d-2)^n + \sum_{k\neq k'}\left((x_i^k x_i^{k'} + \sum_{j\neq i} x_j^k x_j^{k'})^n - (-x_i^k x_i^{k'} + \sum_{j\neq i} x_j^k x_j^{k'})^n\right) \tag{17}$$

In order for $\boldsymbol{\xi}^{(t)} = \mathbf{x}_{k'}$ to be stable, we need that $\Delta_i E > 0$ for $i = 1, \ldots, d$, which will cause the i^{th} bit to remain unchanged when applying the update rule (15). Notice that if the the $k = k'$ term in equation (16), which we will label E_{signal}, dominates the sum then flipping the i^{th} bit will correspond to $\Delta_i E > 0$, which is what we need for

[9]It is not technically a surface since the state space is discrete, but it is helpful to visualise it this way.

stability. However, it is clearly possible for the terms $k \neq k'$, which we will label E_{noise}, to contribute to the sum in such a way that the change in energy is negative, which would result in the state of the network being unstable. For this to happen we require $|E_{noise}| > |E_{signal}|$ and $E_{noise} < 0$. We can compute the approximate probability of these two events occurring in the following way. If d and N are large enough then we will have, by the central limit theorem, that $\Delta_i E$ is distributed normally with mean $\langle \Delta E \rangle = E_{signal} = d - (d-2)^n \approx 2n \cdot d^{n-1}$ (for large d, by the binomial theorem), and a variance $\Sigma^2 \approx \Omega_n (N-1) d^{n-1}$ (for large d), where $\Omega_n = 4n^2(2n-3)!!$ [10][11]. Now we would like to know the probability that the fluctuations contributed by E_{noise}, which we will denote $Y = \Delta_i E - \langle \Delta_i E \rangle$ and which is also a normally distributed random variable in the limit, will both exceed the magnitude of E_{signal} and be less than zero. Since the normal distribution is symmetric, we can compute the probability of this occurring as:

$$P_{error} = \int_{E_{signal}}^{\infty} \frac{dy}{\sqrt{2\pi\Sigma^2}} e^{-\frac{y^2}{2\Sigma^2}} \approx \sqrt{\frac{(2n-3)!!}{2\pi} \frac{N}{d^{n-1}}} e^{-\frac{d^{n-1}}{2N(2n-3)!!}}$$

where the integral is computed easily using a conversion to polar coordinates. If we require $P_{error} < 0.05$ - that stored patterns correspond to unstable states only 5% of the time when patterns are generated randomly - then we can compute an upper limit on the critical storage capacity as:

$$N_c = \lambda_n d^{n-1} \tag{18}$$

where λ_n is, as in the traditional Hopfield model, the load parameter, but now parameterised by n. Note that Setting $n = 2$, we recover $\lambda_2 \approx 0.14$ which is the value of the load parameter for a small percentage retrieval error in the traditional Hopfield model. The key realisation about the expression (18) is that N_c is polynomial in d and that there is a powerful non-linear relationship between d and n which gives a far larger storage capacity than the original model. We may then extend this condition to $P_{error} < 1/d$ which, in the limit of d, will ensure no stored pattern is unstable, the authors of [11] state, but do not prove, that the critical storage capacity is:

$$N_c = \frac{1}{2(2n-3)!!} \frac{d^{n-1}}{ln(d)} \tag{19}$$

which is slightly more restrictive, but N_c still grows with d in a highly non-linear way. Note that the result expressed by (19) is not derived explicitly in [11], a proof may be seen in [13].

The proposition of the new-and-improved model for dense associative memory prompted a near immediate response in the form of a paper published in 2017 which extended the new dense associative memory model by proposing an exponential interaction function, as opposed to a polynomial one, which was shown to lead to storage capacity which is exponential in d - better still. This extension to the modern Hopfield model is what we will explore next, and is arguably one of the most important results in the literature on Hopfield networks to date.

[10]The double factorial is not, as you would assume, a factorial of a factorial. It is in fact the product of a sequence of numbers which decreases by 2, as opposed to 1.

2.2 Exponential Storage Capacity

While considering the polynomial energy function, and the corresponding approximately-polynomial storage capacity as expressed in (18), it is natural to wonder what happens if we take $n \to \infty$. Hopfield and Krotov suggest in [11] that the performance degrades for very large n, however, the authors of a 2017 paper titled *On a Model of Associative Memory with Huge Storage Capacity* [13] prove in great mathematical detail that this is not the case. The major contribution from [13] is proving that, in fact, using an exponential interaction function, as opposed to a polynomial one, to compute the energy of the network leads to a critical storage capacity which is exponential in the size of the network d whilst maintaining the basins of attraction about the points in phase space corresponding to the stored patterns. We will now go on to state and prove the theorem which is the main contribution of the paper, and which cements the mathematical relationship between:

1. The critical storage capacity N_c;
2. The size of the network d, and;
3. The $(100 \cdot \rho)\%$ initial error required for error-free retrieval.

It is important to keep in mind while considering the theorem below that since the original Hopfield model was proposed the relationship between these three factors had (to my knowledge) never been established prior to [13], despite the large amount of analysis published on the Hopfield model.

Theorem 4. *Consider a modern Hopfield network with dynamics governed by a transition function $T_i(\boldsymbol{\xi})$ given by equation (15), with an interaction function of the form $F(x) = e^x$. For a fixed $0 < \alpha < log(2)/2$ let $N = exp(\alpha d) + 1$ be the storage capacity of the network, and let $\{\mathbf{x}_k\}_{k=1}^N$ be N patterns sampled randomly and uniformally from $\{-1, 1\}^d$. Moreover, fix $\rho \in [0, 1/2)$ and let α depend on ρ such that*

$$\alpha < \frac{I(1-2\rho)}{2}$$

where $I(x) = \frac{1}{2}((1+x)log(1+x) + (1-x)log(1-x))$. Then, for any k and any $\tilde{\mathbf{x}}_k$ taken from the Hamming sphere[11] *with radius ρd (assumed to be an integer) and centered at $\mathbf{x}_k$, $\mathcal{S}(\mathbf{x}_k, \rho d)$, we have that*

$$\mathbb{P}\left(\exists k \, \exists i : T_i(\tilde{\mathbf{x}}_k) \neq x_i^k\right) \to 0$$

as $d \to \infty$.

Proof. A detailed proof of this theorem is given in *Appendix B*. □

Let us make a few observations about this theorem. Firstly, in words, theorem 4 states that, given certain numerical constraints, a modern Hopfield network with an exponential interaction function will be able to store exponentially many patterns in the size of the network. Furthermore the theorem assures us that not only will the stored patterns correspond to stable fixed points in the phase space, but that the network will be able to retrieve any distorted pattern by correcting individual bits in a single step, given that the % initial error is inversely related to the storage capacity in the manner described above. Secondly, note that the theorem may be proved analogously for a Hamming ball $\mathcal{B}(\mathbf{x}_k, \rho d)$, that is, the theorem holds for all Hamming spheres with a

[11]The *Hamming distance* between two binary strings is the number of bits which are different between them. A Hamming sphere with radius $\mathcal{S}(\mathbf{y}, r)$, is simply the set of all binary strings which are a hamming distance of r from $\mathbf{y}$.

radius smaller than ρd.

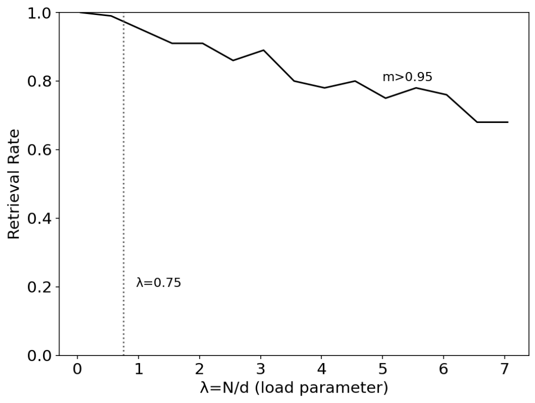

Figure 9: The above plot illustrates the results of a Monte-Carlo simulation where the retrieval rate for randomly generated patterns plotted against the load parameter $\lambda = N/d$ for a modern Hopfield network with an exponential interaction function. The load parameter corresponding to the critical storage capacity (according to theorem 4) is plotted as a dashed line ($\lambda = 0.75$)

We now repeat the same simulation as given in Figure 5, using a modern Hopfield network with an exponential interaction function. The results are illustrated in Figure 9: here we have simulated a network with $d = 20$ neurons, varying the number of stored patterns N, and therefore the load parameter, plotting the percentage of successful retrievals, that is where $m > m^* = 0.95$. As with the previous simulation, we set the initial error to be $(100 \cdot \rho)\%$, where $\rho = 0.15$, for all starting states. This gives, according to tehorem 4, a critical storage capacity of N_c =exp$(\alpha d) + 1 = 15$, where $\alpha \approx 0.135 < I(1-2\rho)/2$ - the corresponding critical load parameter $\lambda = N_c/d = 0.75$ has been indicated in the plot of the results, which again supports the theoretical result. By comparison, the classic Hopfield network under the same conditions had a retrieval rate of 70% by $\lambda \approx 0.15$, versus $\lambda \approx 7.0$ (7 times the size of the network) for the modern Hopfield network. The improvement in performance is hard to overstate.

As a further illustration and test of the Hopfield networks performance under the task of storing and retrieving highly correlated patterns, we expand the Simpson experiments performed above. Figure 10 illustrates the result of storing 24 binary patterns, corresponding to images of the faces of various characters from the Simpsons, and attempting to retrieve the image of Homer, starting in a state which is $\approx 50\%$ altered (i.e. masked Homer). As illustrated the network retrieves the correct image, and moreover it corrects all the bits in the state after a single update. The classic Hopfield network by comparison could not correctly retrieve the same pattern when we attempted to store 4 binary patterns from the same data set. The code for the modern Hopfield network with an exponential interaction function may be found in Appendix C. In the final section we go on to examine the most recent iteration of the Hopfield model - Hopfield networks able to store and retrieve real-valued patterns (i.e. continuous-state Hopfield networks).

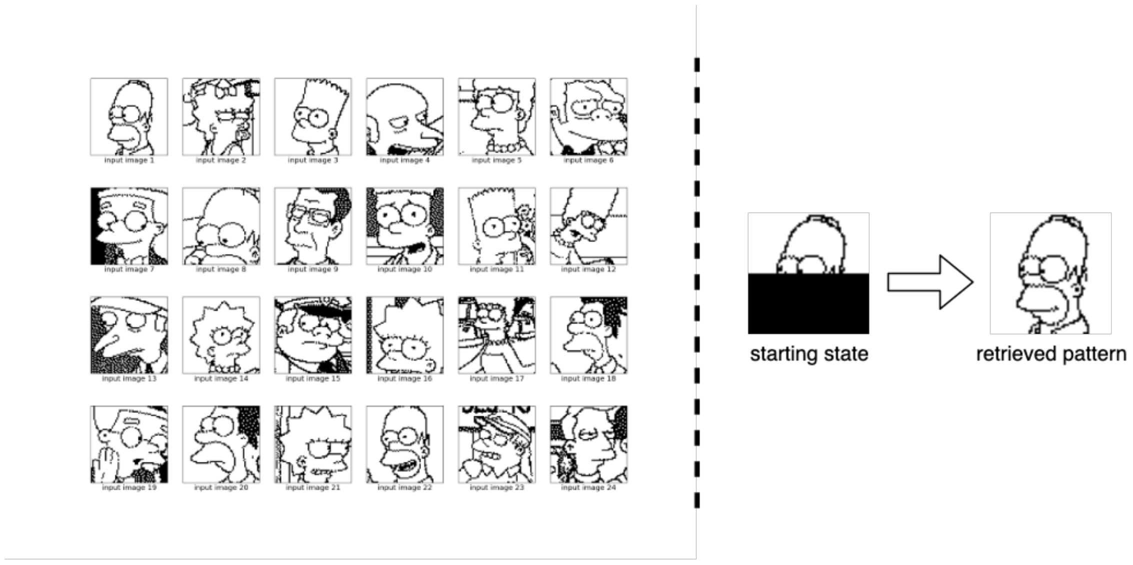

Figure 10: Successful pattern retrieval in a Modern Hopfield network with an exponential interaction function.

2.3 Continuous-State Hopfield Networks

A paper titled *"Hopfield Networks Is All You Need"* [16] was published earlier this year (2020) which extends the work in [13] (exponential storage capacity) to modern Hopfield Hopfield networks which are able to store and retrieve real-valued patterns. That is, the state vector of the newly proposed Hopfield model is $\boldsymbol{\xi} \in \mathbb{R}^d$. Additionally, the authors of [16] propose a new energy function, as well as an update rule, which give rise to a continuous state Hopfield model with the following properties:

1. Exponential storage capacity in the size of the network, d;
2. Global convergence to a local minimum, and;
3. Convergence after one update of the state of the network with high probability.

Energy function Note that the energy function given by equation (14) with $F(x) = \exp(x)$ can be written as $E = -\exp(\text{lse}(1, X^T\boldsymbol{\xi}))$, where $X = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ are the stored patterns, and lse stands for the *log-sum-expontial* function which has the form:

$$lse(\beta, \mathbf{y}) = \beta^{-1}\text{log}\left(\sum_{i=1}^{N} \exp(\beta y_i)\right)$$

for some $\beta > 0$, which is called the *inverse temperature* parameter. Then, the new energy proposed in [16] has the form:

$$\begin{aligned} E(\boldsymbol{\xi}) &= -\text{lse}(\beta, X^T\boldsymbol{\xi}) + \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} + \beta^{-1}\text{log}(N) + \frac{1}{2}M^2 && (20) \\ &= -\text{lse}(\beta, X^T\boldsymbol{\xi}) + \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} + C && (21) \end{aligned}$$

where $M = \max_i \|\boldsymbol{x}_i\|$. The first term in equation (20) is simply the (negative) log of the (negative) exponential energy proposed in [13]. The second (quadratic) term is used to ensure that the energy remains finite[12]. The

[12]Discrete Hopfield networks do not have this issue since the state is binary, and therefore bounded.

third and fourth terms are simple normalising constants. The $\beta^{-1}\log(N)$ term may be merged with the lse term, simply dividing the sum inside the log over all N patterns by N. Finally, the $\frac{1}{2}M^2$ term performs a function analogous to the second term; preventing the energy from growing too large. Importantly, it is simple to show (as in [16]) that the energy is bounded like

$$0 \leq E \leq 2M^2$$

Update rule The new energy function defined above has good properties which allow for the engineering of a new update rule which may be derived using the so-called *Concave-Convex Procedure* (CCCP) [19]. The CCCP is a method of deriving an update rule based on an energy function leading to a sequence of states which causes the energy to decrease monotonically. It is defined by the following theorem.

Theorem 5. *Consider a vector-valued energy function $E(\mathbf{v})$ which is bounded from below, and may be written in the form*

$$E(\mathbf{v}) = E_{vex}(\mathbf{v}) + E_{cave}(\mathbf{v})$$

where $E_{vex}(\mathbf{v})$ and $E_{cave}(\mathbf{v})$ are convex and concave functions of $\mathbf{v}$, respectively. Then, the discrete iterative update rule $\mathbf{v}^{(t)} \mapsto \mathbf{v}^{(t+1)}$ derived from

$$\nabla_{\mathbf{v}} E_{vex}(\mathbf{v}^{(t+1)}) = -\nabla_{\mathbf{v}} E_{cave}(\mathbf{v}^{(t)})$$

is guaranteed to produce a sequence of energy values which decreases monotonically and will hence converge to a minimum or saddle point of $E(\mathbf{v})$.

Now, using the CCCP the new update based on the new energy function may be derived in the following way. Note that we can write the energy function given by equation (20) as

$$E(\boldsymbol{\xi}) = E_1(\boldsymbol{\xi}) + E_2(\boldsymbol{\xi})$$

where $E_1(\boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} + C$ is convex, since it has a quadratic form, and $-\text{lse}(\beta, X^T\boldsymbol{\xi})$ is concave, since $\text{lse}(\beta, X^T\boldsymbol{\xi})$ is convex [13]. Therefore, since the energy is bounded from below, we can use the CCCP to derive a discrete, iterative update rule as follows

$$\begin{aligned}
&\nabla_{\boldsymbol{\xi}} E_1(\boldsymbol{\xi}^{(t+1)}) = -\nabla_{\boldsymbol{\xi}} E_2(\boldsymbol{\xi}^{(t)}) \\
\implies &\nabla_{\boldsymbol{\xi}} \left[\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} + C\right](\boldsymbol{\xi}^{(t+1)}) = \nabla_{\boldsymbol{\xi}} \left[\text{lse}(\beta, X^T\boldsymbol{\xi})\right](\boldsymbol{\xi}^{(t)}) \\
&\implies \boldsymbol{\xi}^{(t+1)} = \left[\frac{\beta^{-1}}{\sum_{k=1}^{N}\exp(\beta\mathbf{x}_k^T\boldsymbol{\xi})}\left(\sum_k \beta x_1^k \exp(\beta\mathbf{x}_k^T\boldsymbol{\xi}), \ldots, \sum_k \beta x_d^k \exp(\beta\mathbf{x}_k^T\boldsymbol{\xi})\right)^T\right](\boldsymbol{\xi}^{(t)}) \\
&\implies \boldsymbol{\xi}^{(t+1)} = \left[\frac{X}{\sum_{k=1}^{N}\exp(\beta\mathbf{x}_k^T\boldsymbol{\xi})}\left(\exp(\beta\mathbf{x}_1^T\boldsymbol{\xi}), \ldots, \exp(\beta\mathbf{x}_N^T\boldsymbol{\xi})\right)^T\right](\boldsymbol{\xi}^{(t)})
\end{aligned}$$

Altogether this yields an update rule with the following, simple form:

[13]Let X be a convex set in a real vector space. Then, a function $f : X \to \mathbb{R}$ is convex if $\forall x_1, x_2 \in X, \gamma \in [0,1]$ we have that $f(\gamma x_1 + (1-\gamma)x_2) \leq \gamma f(x_1) + (1-\gamma)f(x_2)$. The lse function is indeed convex, but the proof will not be included in this report.

$$\boldsymbol{\xi}^{(t+1)} = X\text{softmax}(\beta X^T \boldsymbol{\xi}^{(t)}) \tag{22}$$

Notice that in the update rule given by equation (22), the softmax$(\beta X^T \boldsymbol{\xi}^{(t)})$ function simply creates a distribution over all stored patterns weighted by the dot product of the state pattern, $\boldsymbol{\xi}$, which the authors of [16] call the *query*, with the stored patterns in X. An important assumption made by the authors about the pattern matrix X, upon which the update rule given by equation (22) depends, is that for each stored pattern $\mathbf{x}_i$ we have $||\mathbf{x}_i|| = K$, where $K \in \mathbb{R}$ is the radius of a sphere in $\mathbb{R}^d$ centered at the origin on which all stored patterns exist. This ensures that if the query (state pattern) corresponds to one of the stored patterns $\mathbf{x}_i$, the largest dot product when taken over all patterns $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ will be with the i^{th} pattern (itself) - this ensures pattern retrieval works as expected[14]. Returning to equation (22), after obtaining the weighted distribution over patterns from the softmax function, the dot product of the distribution is taken with the pattern matrix X which gives a weighted average over the stored patterns, whether the largest weighting is given to the pattern which has the largest dot product (similarity) with the query. Notice to that the weighted distribution depends directly on the parameter β, referred to by the authors of [16] as the *inverse temperature* parameter. Setting $\beta = \mathbf{0}$, the softmax function would return a uniform distribution, meaning that the update rule would return a uniformally weighted average over all patterns in X. On the other hand, as we make β larger, we increase the relative contributions of the dot products computed by $X^T\boldsymbol{\xi}^{(t)}$ inside the softmax function, and therefore increase the basin of attraction about the stored patterns in the phase space such that, in the limit $\beta \to \infty$, the update rule will return the pattern in X which most closely resembles the query. We illustrate the effect of varying β below, and the emergence of what the authors of [16] call *meta-stable states* (states which we previously referred to as corresponding to spurious minima of the energy function). We now establish he so-called global convergence theorem given in [16].

Theorem 6. *By applying $\boldsymbol{\xi}^{(t+1)} = X\mathit{softmax}(\beta X^T \boldsymbol{\xi}^{(t)})$ iteratively we have that $E(\boldsymbol{\xi}^{(t)}) \to E(\boldsymbol{\xi}^*)$ as $t \to \infty$, for a fixed point $\boldsymbol{\xi}^*$.*

Proof. The update rule given by equation (26) is derived using the CCCP [19] which guarantees us that the energy $E(\boldsymbol{\xi}^{(t)})$ will decrease monotonically as the update rule is recursively applied (as $t \to \infty$). Then since E is bounded from below the sequence of energy values is guaranteed to converge to a local minima (or saddle point[15]) of the energy function. $\square$

Theorem 6 assures us that we have convergence of the energy to a local minima or (a saddle point) if we apply the update rule given by equation (22). However, we still need to be sure that we have convergence of the network state, that is $\boldsymbol{\xi}^{(t)} \to \boldsymbol{\xi}^*$ as $t \to \infty$. The following theorem gives us such assurance.

Theorem 7. *Given a continuous state, modern Hopfield network in some state $\boldsymbol{\xi}^{(t)}$ at time t, if we recursively apply the update rule given by equation (22), we are guaranteed to get $E(\boldsymbol{\xi}^{(t)}) \to E(\boldsymbol{\xi}^*)$ as $t \to \infty$, for some fixed point $\boldsymbol{\xi}^*$. Furthermore we will have $||\boldsymbol{\xi}^{(t+1)} - \boldsymbol{\xi}^{(t)}|| \to 0$ and either:*

1. *$\{\boldsymbol{\xi}^{(t)}\}_{t=1}^{\infty}$ converges, or;*

[14]Practically, this involves normalising each of the stored patterns such that all stored patterns correspond to d-dimensional vectors of the same length.

[15]The authors of [16] note that they never once witnessed convergence to a saddle point in all their experiments.

2. *The set of limit points of $\{\boldsymbol{\xi}^{(t)}\}_{t=1}^{\infty}$ is a connected, compact subset of $\mathbb{L}(E^*)$ where $\ell(a) = \{\boldsymbol{\xi} \in \ell | E(\boldsymbol{\xi}) = a\}$. If $\ell(E^*)$ is finite, then any sequence $\{\boldsymbol{\xi}^{(t)}\}_{t=1}^{\infty}$ generate by applying the update rule given by equation (22) converges to some $\xi^* \in \ell(E^*)$.*

In words, theorem 7 simply states that either we are guaranteed to converge to a single fixed point, or the state will move between (continuous) states in a connected, compact set which all correspond to the same energy minima. We will now go on to examine some results regarding the storage capacity of continuous state, modern Hopfield networks.

Storage capacity Before we cover the theorem which, as with the modern binary state variant above, assures us that the continuous state variant proposed in [16] has an exponential storage capacity in the size of the network d, we need to define successful storage and retrieval for continuous states first. First, we consider the $d-$dimensional sphere S_i around some point in the state space corresponding to a pattern $\mathbf{x}_i$. We will say that $\mathbf{x}_i$ is *stored* if there is a single fixed point $\mathbf{x}_i^* \in S_i$ to which all points $\boldsymbol{\xi} \in S_i$ converge when applying the update rule given by equation (22). Furthermore we require $S_i \cap S_j$ for $i \neq j$. Then, we say $\mathbf{x}_i$ is *retrieved* if we have convergence of the the state of the network to the single fixed point $\mathbf{x}_i^* \in S_i$. Finally, we define the *retrieval error* as $||\mathbf{x}_i - \mathbf{x}_i^*||$, the vector norm between the stored pattern and the fixed point. We now go on to state the storage capacity theorem for continuous state, modern Hopfield networks.

Theorem 8. *We consider randomly generated, continuous state patterns in $\mathbb{R}^d$ which lie on the sphere centered at the origin with radius $M = K\sqrt{d-1}$, where $K \in \mathbb{R}$. Additionally, we assume (choose) so-called failure probability $0 < p \leq 1$. We then define the following constants:*

$$a := \frac{2}{d-1}\left[1 + ln(2K^2\beta p(1-d))\right] \tag{23}$$

$$b := \frac{2K^2\beta}{5} \tag{24}$$

$$c := \frac{b}{W_0\, exp(a + ln(b))} \tag{25}$$

where W_0 is the upper branch of the Lambert W function [16]*. We also ensure that $c \geq (\frac{2}{\sqrt{p}})^{4/(d-1)}$. Then, with probability $1-p$, the number of patterns that can be successfully stored and retrieved is*

$$N \geq \sqrt{p}c^{\frac{d-1}{4}}$$

We will omit the proof of this theorem as it it similar in nature to the proof of theorem 4. We will however make some observations about the theorem. The first and most important observation is that, in spite of the complex relationships between the variables a, b and c, the critical storage capacity N grows exponentially in d, which is what we want. Secondly, notice that the critical storage capacity is inversely related to the proportionality constant K which determines the radius of the sphere from which we take our random patterns, meaning that a larger variance across the set of patterns which we would like to store translates to a smaller storage capacity.

The final two important results given by the authors of [16] are (i) that the continuous state, modern Hopfield network with the dynamics defined above typically converges to a fixed point after one update (a specific upper

[16]The Lambert W function is a complex function used to solve transcendental equations. The "upper branch" is a reference to the branches which arise when rotating about a branch point singularity in the complex plane.

bound is given in [16] for the error after a single update between the resulting state and the closest fixed point), and (ii) that the retrieval error for all stored patterns is bounded from above, vanishing if the stored patterns are well separated (non-correlated).

Using the same test case as for the previous variants of the Hopfield network examined above, Figure 11 illustrates the successful storage and retrieval of the real-valued 'Homer' pattern by a continuous state, Modern Hopfield, from a set of 24 real-valued patterns. Additionally, 12 illustrates the effect of varying the inverse temperature parameter β, as seen in the update rule given by equation (22). Notice that for lower values of β, corresponding to a low temperature, we observe the emergence of *meta-stable* states, which are essentially weighted averages of

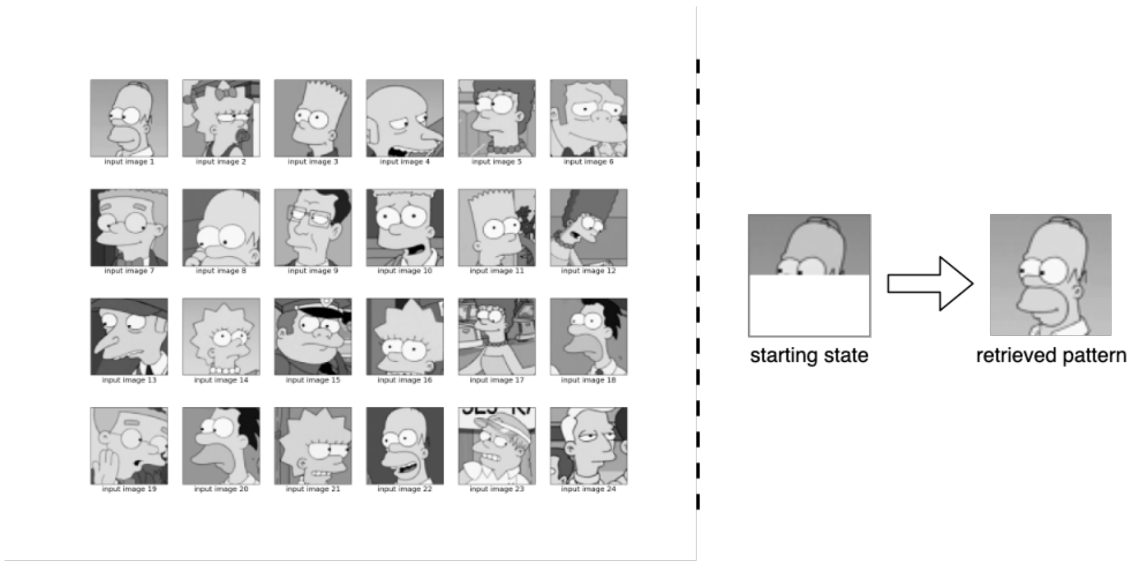

Figure 11: Successful pattern retrieval in a Modern, continuous state Hopfield network.

several patterns which correspond to points in the phase space which exist in close proximity to the fixed point corresponding to the 'Homer' vector. On the other hand, large values of β, corresponding to a low temperature, which causes the basins of attraction of the individual stored patterns to remain well separated, ensuring convergence to a fixed point corresponding to a stored pattern [16]. Note that, as we will explore below, in some practical applications the emergent meta-stable states may be thought of as prototypes [17] - generalised representations of similar, nearby patterns in the phase space - which may be used to improve class separation in classification tasks, as an example. We will now go on to explain the link between the update rule given by equation (22) and the attention mechanism used in modern Transformer models.

2.4 From Hopfield Networks to Transformer Attention

One of the principle contributions of the "Hopfield Networks is All You Need" paper [16] is showing that the update rule given by equation (22) is equivalent to the *attention* mechanism used in modern sequence to sequence models such as the *transformer* or *BERT* (Bidirectional Encoder Representations from Transformers) models. In its most basic form, *attention* in sequence to sequence mapping tasks is defined as the relative importance, or weighting, of different elements in the input and/or output sequence when trying to map a single element from one sequence to the other. The easiest way to understand this concept is consider the example of self-attention in

[17]We use the term in the same way it is used by Krotov and Hopfield in [11]

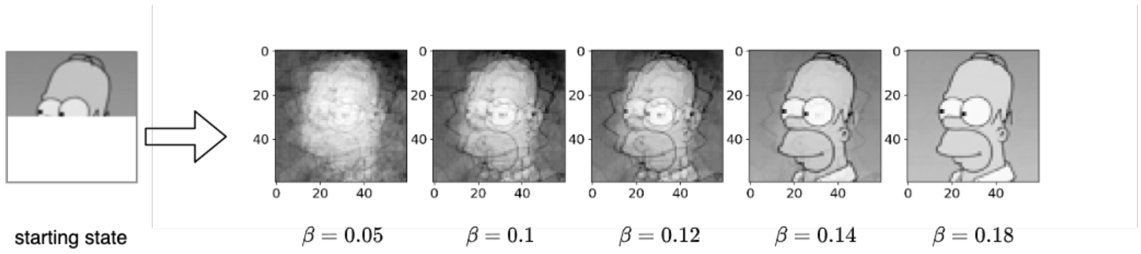

Figure 12: An illustration of the emergence of meta-stable states in a continuous state, modern Hopfield network when varying β.

a single sentence, as illustrated in Figure 11, where when asking the question "what other words in the sentence give us the most information about the word 'it'?". Clearly "it" is a reference to "the animal" and so we see that the self-attention (computed between the sentence and itself) weights "the" and "animal" as giving the most information about the word "it" in the context of the sentence. The attention mechanism is the central to the transformer model introduced in the paper *"Attention Is All You Need"* [18] which has lead to record breaking performance in the field of Natural Language Processing, in particular on translation tasks, which are sequence to sequence tasks by definition.

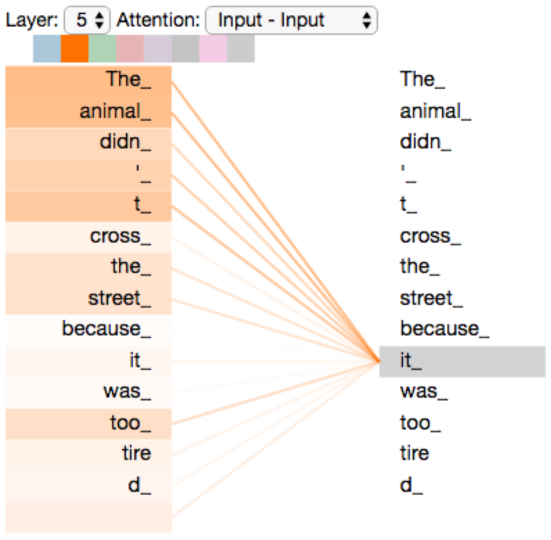

Figure 13: Self-attention over a single sentence, as computed by the self-attention mechanism of the transoformer. Image source: *www.jalammar.github.io/illustrated-transformer/*

To understand how the Hopfield update rule given by equation (22) is equivalent to the attention mechanism of the transformer, a good illustrative analogy is that of making a query in a relational database. Typically we would query a relational database by providing a query $\mathbf{Q}$ with the hopes of retrieving a value $\mathbf{V}$, which is simply some data stored in the database. To do so we compare the query with a set of keys, each of which is associated with a value, using a similarity function f, after which we would return the value corresponding to the key which is most "similar" to the query we submitted. In other words, the similarity function acts as a map $f(Q, K) \rightarrow V$, for some key K, some query Q and some value V. This is illustrated in Figure 12.

Notice that there is a clear similarity between the process of querying a relational database, and performing the

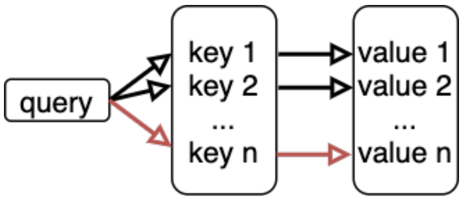

Figure 14: An illustration of how a query is performed in a relational database. The query is compared to each key by a "similarity function" and the value with the largest corresponding "similarity" is returned.

state update given by equation (22); for the sake of this comparison we may view the network state $\boldsymbol{\xi}$ as the query which is then compared which each pattern in X, which may be seen as the keys. The softmax function computes the similarity - a probability distribution or weighting over stored patterns - after which a pattern (or a weighted average over the stored patterns) is then selected from X via the dot product. To show this explicitly we consider the following adaptation of the update rule given by equation (22). Instead of a single state pattern, we consider a matrix of state patterns (corresponding to a sequence) $R = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_S)^T$ and, as usual, a matrix of stored patterns $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^T$ (which may also, for the sake of this example, be viewed as a sequence). Now, instead of comparing the two sets of patterns directly, we would first like to map them to an associative space of dimension d_k in the following way:

$$Q = RW_Q$$
$$K = YW_K$$

where $W_Q \in \mathbb{R}^{d \times d_k}$ and $W_K \mathbb{R}^{d \times d_k}$ are learnable weight matrices. We may view Q as the "query" matrix and K as the "key" matrix. We then set $\beta = 1/\sqrt{d_k}$ and compute the output of the adapted update rule as

$$(Q^{new})^T = K^T \text{softmax}\left(\frac{1}{\sqrt{d_k}} K Q^T\right)$$
$$\implies Q^{new} = \text{softmax}\left(\frac{1}{\sqrt{d_k}} Q K^T\right) K$$

where the softmax function acts on each column of the matrix $\frac{1}{\sqrt{d_k}} Q K^T$ to produce a multinomial probability distribution vector. Finally, we map the output of the update rule to an output space of dimensions d_v using another weight matrix $W_v \in \mathbb{R}^{d_k \times d_v}$, which altogether gives:

$$Z = Q^{new} W_v = \text{softmax}\left(\frac{1}{\sqrt{d_k}} Q K^T\right) K W_v = \text{softmax}\left(\frac{1}{\sqrt{d_k}} Q K^T\right) V \tag{26}$$

Equation (27) is the exact form of the attention mechanism given in [18] between the two sequences of vectors R and Y.

It is interesting to think of the modern Hopfield network as computing the attention of one pattern, the query or state ξ, over all the stored patterns X as opposed to simply performing pattern retrieval. Taking this view, the authors of [16] have created a so-called *Hopfield Layer*, which may be integrated into any deep neural network architecture to perform various useful functions, including attention, self-attention (if $R = Y$) and pooling. The architecture of the Hopfield Layer is shown below in Figure 13.

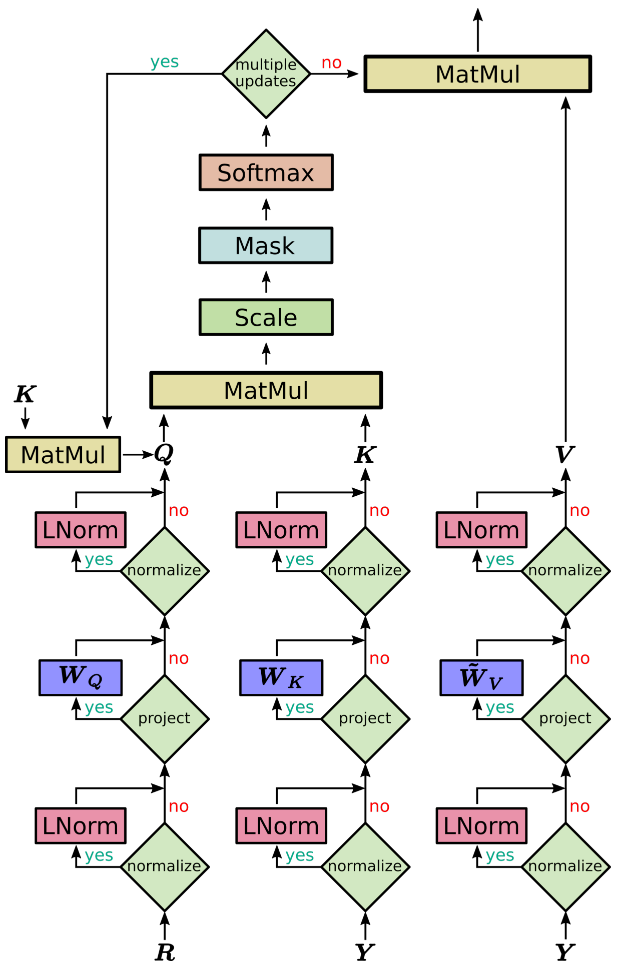

Figure 15: An illustration of the versatile Hopfield Layer proposed in [16].

3 Experiments: Pattern Recovery and Pattern Separation Using Hopfield Networks Applied to the MNIST Data Set

The MNIST data set is a collection of 70,000 images of handwritten digits, a selection of which are visualised in Figure 16, consisting of 10 distinct classes (one for each digit) of approximately 7000 instances each. Each digit may be represented as a real-valued pattern vector of 784 pixels; $\mathbf{x} \in [0,1]^{784}$. In this section we will perform two experiments to illustrate some of the properties and functionality of the continuous state, modern Hopfield network proposed in [16]. First, we will conduct a **pattern retrieval** experiment to determine how accurately the Hopfield model is able to perform accurate retrieval given a noisy/corrupted version of a stored digit pattern. Second, we will conduct a **pattern separation** experiment in order to determine whether or not the Hopfield model may be used to separate unlabelled patterns into their natural classes.

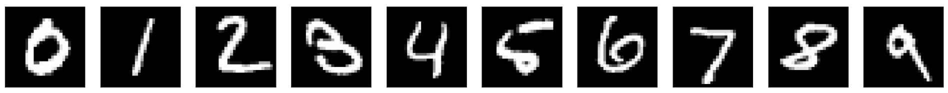

Figure 16: A sample of ten digits from the MNIST data set. Each digit is a real-valued vector of 784 pixels, represented above in a 60×60 grid.

3.1 Methodology

Pattern retrieval To test the effectiveness of the Hopfield model proposed in [16] when used for retrieval of corrupted patterns, we will conduct a Monte Carlo simulation similar in nature to the simulations used to test the storage capacity of the Hopfield model variants explored above. We will use the MNIST test set, which consists of 10,000 digit patterns, with approximately 1000 samples per digit class. As per usual, let X be the matrix of stored $N = 10000$ patterns, where each pattern vector has dimension $d = 784$. In order to ensure all pattern vectors in X exist on a sphere (having the same norm) we will normalise each pattern to unit length, and then divide each pattern by $\max_{k,i} x_i^k$ to ensure the elements of all pattern vectors are in the range [0,1]. Now, we perform our Monte Carlo simulation in the following way. Select, uniformally at random, a pattern $\mathbf{x}_i$ from the set of patterns in X. Next select, uniformally at random, ρd pixels from $\mathbf{x}_i$, where $\rho \in [0.0, 1.0]$, which we will set to 0 to create a new, altered/corrupted pattern[18] $\tilde{\mathbf{x}}_i$. Now, we will use a Hopfield network to attempt to perform pattern retrieval in the following way. Set the initial state of the Hopfield network to $\boldsymbol{\xi}^{(0)} = \tilde{\mathbf{x}}_i$, then perform iterative updates of the state, $\boldsymbol{\xi}^{(\boldsymbol{t}+\mathbf{1})} = f(\boldsymbol{\xi}^{(\boldsymbol{t})})$, according to the update rule f given by equation (22), until we arrive at a fixed point in the phase space, $\boldsymbol{\xi}^* = f(\boldsymbol{\xi}^*)$, recording the retrieval error $||\boldsymbol{\xi}^* - \mathbf{x}_i||$. We then repeat this process 100 times for a pair of fixed values of β and ρ, averaging over all retrieval errors to get an average. We repeat this for a range of β and ρ values, plotting the results for analysis - see below.

Pattern separation The MNIST data set is a benchmark data set in the statistical and machine learning communities for classification algorithms. For a classification algorithm to perform exceptionally well on MNIST, it needs to be able to determine a decision boundary in a 784-dimensional space of pixels which separates the clusters corresponding to the ten digit classes. What makes this task challenging is the fact that several of the classes are highly correlated (for example, the digit classes 4 and 9 contain several digit images which, even to the human eye, may be hard or impossible to identify correctly). The K-nearest neighbours algorithm is an example of a classification algorithm which relies on good class separation/clustering of classes. We would like to test whether we could use the emergent meta-stable states, which arise in phase space as the parameter β is decreased in the Hopfield model, in order to better separate the clusters of MNIST digit classes. In order to explore the effectiveness of the Hopfield model when applied to this task, we will select all patterns belonging to either of two classes from the MNIST test set, storing them in the pattern matrix X. Then, for each pattern $\mathbf{x}$ in X we set the initial state of our Hopfield model $\boldsymbol{\xi}^{(0)} = \tilde{\mathbf{x}}$, and again, apply the Hopfield update f until we obtain a fixed point, $\boldsymbol{\xi}^* = f(\boldsymbol{\xi}^*)$. We will repeat this process for all patterns in X for a range of β values, and, for each value of β, we will project the resulting fixed points onto a 2-dimensional plane using principle component analysis (2D) in order to analyse the separation of classes. Our hypothesis will be that as we drive β towards zero, the Hopfield updates will converge on fixed points corresponding to meta-stable states which are essentially weighted averages over nearby patterns. Clearly in the limit $\beta \to 0$ all fixed points in the phase space will eventually coalesce to form a single fixed point - a uniformly weighted average over all stored patterns - while, on the other hand, for $\beta >> 1$ we will have approximately the same number of fixed points as patterns. The ideal value of the β parameter will be one which allows for the coexistence of two fixed points, one corresponding to each class, where when Hopfield updates performed on starting state which corresponds to a pattern in one of

[18]The same experiment was conducted, except instead of setting the randomly chosen pixels to 0 each pixel value was sampled from a normal distribution (i.e. Gaussian noise), and the results were not found to be significantly different.

the two classes will yield convergence to the fixed point associated with that same class. We will perform this process for a select few pairs of digits and analyse the apparent effectiveness of the Hopfield model applied to the task of pattern separation.

3.2 Results and Discussion

Pattern retrieval Figure 17 illustrates two examples of pattern retrieval for a digit from the 0 class, using a continuous state, modern Hopfield network with $\beta = 4.0$ for two different percentages of the initial error. The results are as expected: the larger the initial percentage error (the more corrupted the pattern), the higher the retrieval error.

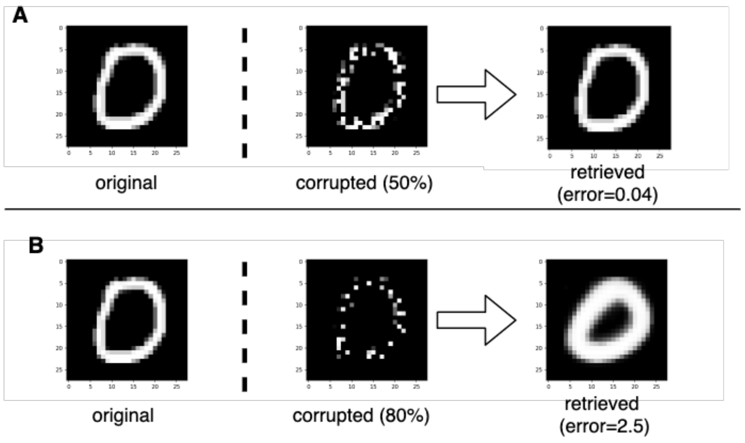

Figure 17: An example of pattern retrieval by a continuous state, modern Hopfield network. Here $\beta = 4.0$ for both examples. **A.** an initial error of 50%, resulting in a retrieval error of 0.04, and **B.** an initial error of 80%, resulting in a retrieval error of 2.5

The results of the full Monte-Carlo simulation, shown in Figure 18, exhibit some interesting trends. As expected, for large values of β (>2.0) we have that the retrieval error increases reliably with the $(100 \cdot \rho)\%$ initial error. This makes sense because if β is large the basins of attraction about each stored pattern in the phase space remain well separated and so we will have approximately as many fixed points as patterns, meaning that if the $(100 \cdot \rho)\%$ initial error is large then the initial state of the network, being the corrupted pattern, will likely settle at a fixed point which is far away from the original pattern in the phase space. As β decreases however, we see some interesting and unexpected results begin to emerge: if the $(100 \cdot \rho)\%$ initial error is large, smaller values of β produce lower retrieval errors on average. This may be due to the fact that as β is decreased, fixed points, which correspond to stored patterns when $\beta >> 1$, begin to coalesce as the basins of attraction begin to overlap. This ultimately causes a decrease in the number of fixed points, and so for a high $(100 \cdot \rho)\%$ initial error, although the initial state (the corrupted pattern) is far away from the state corresponding to the original pattern in phase space, there are fewer attractors (fixed points) in the phase space between the initial state and the goal state (corresponding to the original pattern) and so the state of the network will often converge on a fixed point corresponding to a meta-stable state which is not as far away from the goal state as would be the case for $\beta >> 1$.

Pattern separation The results of the pattern separation experiment can be seen in Figure 19. Results have only been plotted for a select set of pairs of digits classes: 1 and 7; 2 and 5; 3 and 8; 4 and 9 (from top to

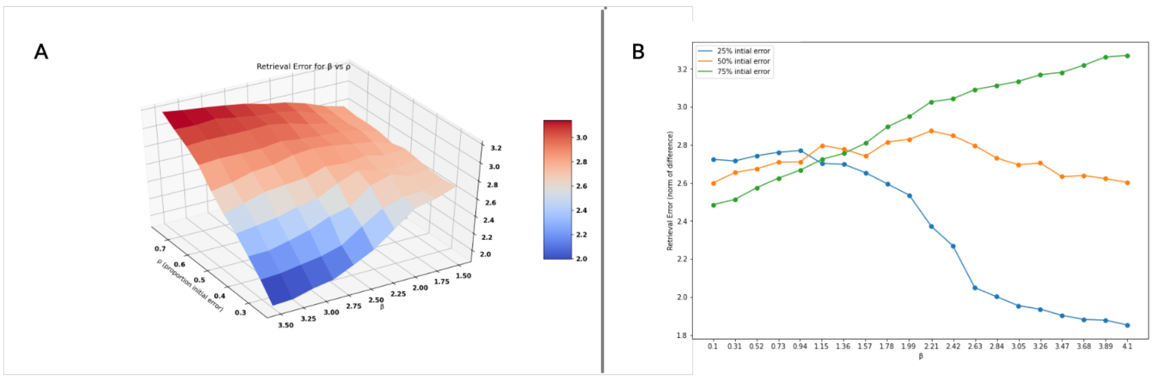

Figure 18: The results of the pattern retrieval experiment applied to the MNIST test set. Retrieval error is compared for values of β and ρ. **A**. A 3D surface plot of retrieval error over a range of β and ρ **B.** Cross sections of retrieval against β for various values of $(100 \cdot \rho)\%$ initial error.

bottom in Figure 19). After applying dimensionality reduction using PCA, the points corresponding to the samples from either class are plotted in the form of a scatter plot (top row) as well as a density plot (top row) since several patterns may converge on a single fixed point, making it difficult to interpret the density of the scatter plot for smaller values of β. We see that class separation works well provided that the classes don't have too large an overlap initially. This is the case with the pairs 1 and 7, as well as with 2 and 5; the Hopfield network is able to iteratively tease apart the patterns for lower values of β, which may improve the effectiveness of some classification methods, like k-nearest neighbours, for example. On the other hand if there is a large overlap between patterns, as is the case with the pairs of digit classes 3 and 8, as well as 4 and 9, we see that the Hopfield model fails to separate the classes, causing all patterns to converge on a single fixed point instead. One way to improve this method may be to use a variation of the update rule given by equation (22) where the dot product $X^T\boldsymbol{\xi}$ is replaced with the generalised dot product $X^TW\boldsymbol{\xi}$, where $W \in \mathbb{R}^{d\times d}$ is a learnable weight matrix (this is similar to the way that the attention mechanism is derived from the Hopfield update). If the objective measure were carefully designed, the weights in W may be learned through backpropogation such that, for a given multi-class pattern separation task, the association between the state and the stored patterns may be optimised towards improved separation of classes.

4 Conclusions and Future Work

In this research report we have explored the evolution of Hopfield networks over the past three decades. We have examined the mathematical properties of all the major variants of the Hopfield network, including the first discrete state associative memory model proposed by John Hopfield in 1982, the subsequent dense associative memory model, the improved dense associative memory model with exponential storage capacity, and finally, the continuous state, modern Hopfield network. Furthermore, we have confirmed the theoretical results established for each of these models via computational simulations, illustrating the relationship between storage capacity, network size and percentage initial error when performing retrieval of stored patterns. Finally, we have examined the relationship between modern Hopfield networks and modern transformers, illustrating the mathematical equivalence between the update rule of continuous state, modern Hopfield networks and the attention mechanism. The established relationship between modern Hopfield networks and transformers will likely open up new avenues for research into associative memory models as it provides and interesting link between energy-based models and deep learning models, which have historically been separate branches of research. Additionally, further research may involve the application of a generalised form of the modern Hopfield network (as illustrated by the 'Hopfield layer' in Figure 15 above, for example) to solving tasks which may require tools which are able to learn and exploit features and properties of associative relationships between sets or sequences of patterns. One example of such research may be seen in the paper "*Modern Hopfield Networks and Attention for Immune Repertoire Classification*" [17] which uses modern Hopfield networks to solve the challenging multiple-instance learning problem of immune repertoire classification.

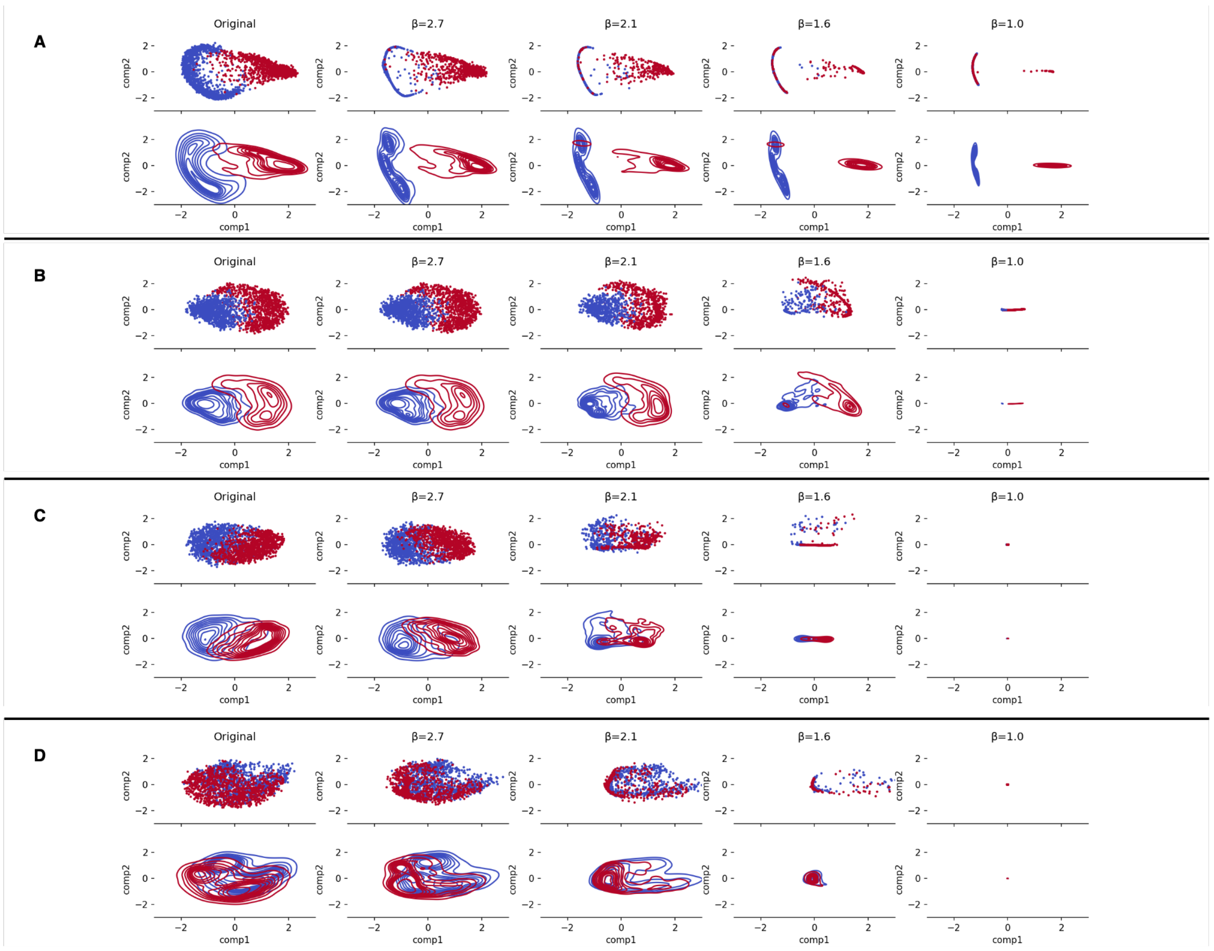

Figure 19: The results of the pattern separation experiment: each plot consists of a scatter plot (top row) and a density plot (bottom), both of which illustrate the results of attempted pattern separation using the Hopfield model for different values of β (higher values on the left, lower values on the right). The dimensionality reduction was performed using principle component analysis which iteratively performs orthogonal projection of data points in $d-$dimensional space onto a (d-1)-dimensional hyper plane, chosen in such a way that variance is maximised. **A.** The digit classes 1 and 7. **B.** The digit classes 2 and 5. **C.** The digit classes 3 and 8. **D.** The digit classes 4 and 9.

References

[1] Hopfield, John. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences Apr 1982, 79 (8) 2554-2558; DOI: 10.1073/pnas.79.8.2554.

[2] Hopfield, John. Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons. Proceedings of the National Academy of Sciences May 1984, 81 (10) 3088-3092; DOI: 10.1073/pnas.81.10.3088.

[3] Amit, Daniel, Gutfreund, Hanoch. Statistical Mechanics of Neural Networks near Saturation. Racah Institute of Physics, Hebrew University, 1986.

[4] Bruck, Jehoshua. On the Convergence Properties of the Hopfield Model . Proceedings Of The IEEE. Vol. 78, No. 10, OCTOBER 1990.

[5] Goles-Chacc, Eric, Françoise Fogelman-Soulié, and Didier Pellegrin. Decreasing energy functions as a tool for studying threshold networks. Discrete Applied Mathematics 12.3 (1985): 261-277.

[6] Takasaki, Kevin. Critical Capacity of Hopfield Networks. MIT Department of Physics, 2007.

[7] Folli, Viola, Marco Leonetti, and Giancarlo Ruocco. On the maximum storage capacity of the Hopfield model. Frontiers in computational neuroscience 10 (2017): 144.

[8] Rocchi, Jacopo, David Saad, and Daniele Tantari. High storage capacity in the Hopfield model with auto-interactions—stability analysis. Journal of Physics A: Mathematical and Theoretical 50.46 (2017): 465001.

[9] Gosti, Giorgio, Folli, Viola, Leonetti, Marco, Ruocco, Giancarlo. Beyond the Maximum Storage Capacity Limit in Hopfield Recurrent Neural Networks. Entropy, 21(8), 726.

[10] Torres, Joaquin, Mejias, Jorge. Maximum memory capacity on neural networks with short-term synaptic depression and facilitationn. Racah Institute of Physics, Hebrew University, 1986.

[11] Krotov, Dimitri, Hopfield, John. Dense Associative Memory for Pattern Recognition. Advances in neural information processing systems. 2016.

[12] Krotov, Dimitri, Hopfield, John. Dense Associative Memory is Robust to Adversarial Inputs. Neural computation 30.12 (2018): 3151-3167.

[13] Demircigil, Mete, Heusel, Judith, Lowe, Matthias, Upgang, Sven, Vermet, Franck. On a Model of Associative Memory With Huge Storage Capacity. Journal of Statistical Physics 168.2 (2017): 288-299.

[14] Dembo, Amir, Zeitouni, Ofer. Large deviations techniques and applications. Volume 38 of Stochastic Modelling and Applied Probability. Springer-Verlag, Berlin, 2010. Corrected reprint of the second (1998) edition.

[15] Krotov, Dimitri, Hopfield, John. Unsupervised learning by competing hidden units. Proceedings of the National Academy of Sciences 116.16 (2019): 7723-7731.

[16] Ramsauer et. al. Hopfield Networks is All You Need. Arxiv preprint, arXiv:2008.02217v1, 2020.

[17] Widrich et. al. Modern Hopfield Networks and Attention for Immune Repertoire Classification. Arxiv preprint, arXiv:2007.13505v1, 2020.

[18] Vaswani et. al. Attention Is All You Need. Arxiv preprint, arXiv:1706.03762v5, 2017.

[19] Yuille, Alan L., and Anand Rangarajan. The concave-convex procedure. Neural computation 15.4 (2003): 915-936

Appendix A: Glossary

- d The number of neurons in a Hopfield network.
- N The number of stored patterns in a Hopfield network.
- X The $d \times N$ matrix of pattern vectors stored in a Hopfield network.
- $\mathbf{x}_k$ A d-dimensional vector corresponding to the k^{th} pattern stored in a Hopfield network.
- x_i^k The i^{th} element of the k^{th} pattern vector stored in a Hopfield network.
- $\boldsymbol{\xi}^{(t)}$ A d-dimensional vector representing the state of a Hopfield network at time t.
- W An adjacency matrix which encodes the weights between neurons in a discrete-state Hopfield network.
- w_{ij} The weight of the edge incident with the i^{th} and j^{th} neurons in a discrete-state Hopfield network.
- $\mathbf{b}$ The vector of biases associated with each neuron in a discrete-state Hopfield network.
- $H_i(t)$ The stimulus received by the i^{th} neuron in a discrete-state Hopfield network at time t.
- $E(\boldsymbol{\xi})$ The energy function of a Hopfield network which depends directly on the state network state.
- $T_i(\boldsymbol{\xi})$ The transition function which determines the update to the activation of the i^{th} neuron in a modern Hopfield network.
- λ The load parameter of a Hopfield network, computed as N/d.
- ρ The proportion parameter determining the % initial error when simulating pattern retrieval in a Hopfield network.
- β The inverse temperature parameter which determines the number and nature of fixed points in the phase space of a continuous state Hopfield model.

Appendix B: Proofs

Proof of Theorem 4

Proof. Note: we will split this proof into six parts to make things as clear as possible.

Part 1 Suppose WLOG that our network is in some state $\tilde{\mathbf{x}}_1$ taken uniformally at random from the Hamming sphere $\mathcal{S}(\mathbf{x}_1, \rho d)$ for a fixed $\rho \in [0, \frac{1}{2})$. Then, given an interaction function $F(x) = e^x$, consider the change in energy that occurs as a consequence of changing the activation of the i^{th} neuron:

$$\Delta_i E(\tilde{\mathbf{x}_1}) = \sum_{k=1}^{N} \left(F\left(x_i^k \tilde{x}_i^1 + \sum_{j \neq i} x_j^k \tilde{x}_j^1 \right) - F\left(-x_i^k \tilde{x}_i^1 + \sum_{j \neq i} x_j^k \tilde{x}_j^1 \right) \right) \tag{27}$$

Now, updating the activation of the i^{th} neuron according to the transition function $T_i(\tilde{\mathbf{x}}_1)$ will result in a change if $\Delta_i E(\tilde{\mathbf{x}}_1) < 0$, but not if $\Delta_i E(\tilde{\mathbf{x}}_1) > 0$[19]. We can split the sum in the energy change into a signal portion, including the $k = 1$ term, and a noise portion, including all terms with $k \neq 1$:

$$\begin{aligned} E_{signal} &= F\left(\sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 \right) - F\left(-2 x_i^1 \tilde{x}_i^1 + \sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 \right) \\ E_{noise} &= \sum_{k=2}^{N} \left(F\left(\sum_{j=1}^{d} x_j^k \tilde{x}_j^1 \right) - F\left(-2 x_i^k \tilde{x}_i^1 + \sum_{j=1}^{d} x_j^k \tilde{x}_j^1 \right) \right) \end{aligned}$$

what we would like to show is that the activation of the i^{th} neuron will update incorrectly if $|E_{noise}| > |E_{signal}|$, and that this event vanishes in the limit $d \to \infty$. In order to see this we consider the two possible cases. Either we will have $\tilde{x}_i^1 = x_i^1$ or we will have $\tilde{x}_i^1 = -x_i^1$. In the first case, we would like the activation of i^{th} neuron to remain unchanged, that is, we want $\Delta_i E(\tilde{\mathbf{x}_1}) > 0$. In the latter case we would like the network to update the activation of the i^{th} neuron to have the correct value, that is, we want $\Delta_i E(\tilde{\mathbf{x}_1}) < 0$. In both cases E_{signal} supports the desired behaviour, indeed we have:

$$\begin{aligned} E_{signal} &= F\left(\sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 \right) - F\left(\sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 - 2 \right) > 0 \quad \text{if } \tilde{x}_i^1 = x_i^1 \\ E_{signal} &= F\left(\sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 \right) - F\left(\sum_{j=1}^{d} x_j^1 \tilde{x}_j^1 + 2 \right) < 0 \quad \text{if } \tilde{x}_i^1 = -x_i^1 \end{aligned}$$

since $F(x) = e^x$ is positive everywhere and monotonically increasing. This means that the activation of the i^{th} neuron will update correctly if $\mathbf{sgn}(E_{signal} + E_{noise}) = \mathbf{sgn}(E_{signal})$, which is fulfilled if $|E_{noise}| < |E_{signal}|$. Therefore a necessary condition for an incorrect update is $|E_{noise}| \geq |E_{signal}|$, which together with the fact that all patterns are sampled randomly and uniformally implies that:

$$\begin{aligned} \mathbb{P}\left(\exists k \, \exists i : T_i(\tilde{\mathbf{x}}_k) \neq x_i^k \right) &\leq d \cdot N \, \mathbb{P}\left(T_i(\tilde{\mathbf{x}}_1) \neq x_i^1 \right) \qquad \text{by definition of } \mathbb{P}(A \cap B) \\ &\leq d \cdot N \, \mathbb{P}\left(|E_{noise}| \geq |E_{signal}| \right) \end{aligned}$$

[19]Note: $\Delta_i E(\tilde{\mathbf{x}}_1) = 0$ is a negligible case in the limit $d \to \infty$

where the last inequality comes from the fact that having $|E_{noise}| \geq |E_{signal}|$ may result in a correct update by chance, but is a necessary condition for an incorrect update to occur. Then, we have that:

$$\begin{aligned} |E_{noise}| &\leq \sum_{k=2}^{N} \left|1 - \exp\left(-2x_i^k \tilde{x}_i^1\right)\right| \exp\left(\sum_{j=1}^{d} x_j^k \tilde{x}_j^1\right) \\ &\leq \left[1 - e^{-2}\right] e^2 \sum_{k=2}^{N} \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) \end{aligned}$$

where $\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle$ is the inner product on $\{\pm 1\}^d$. Additionally, we have:

$$|E_{signal}| > e^{d(1-2\rho)}[1 - e^{-2}]$$

which altogether gives

$$\mathbb{P}\left(\exists k\, \exists i : T_i(\tilde{\mathbf{x}}_k) \neq x_i^k\right) \leq d \cdot N\, \mathbb{P}\left(\sum_{k=2}^{N} e^2 \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)}\right) \tag{28}$$

Part 2 We would now like to show that the RHS of equation (21) is bounded above by an expression that goes to zero as $d \to \infty$. Before we do so, we are going to establish some further definitions and inequalities that we will use later. We first state three results from the theory of large deviations [14]. If $X_{m,p} \sim B(m,p)$ is a binomially distributed random variable, then for some $\epsilon > 0$ we have:

$$\mathbb{P}(X_{m,p} \geq m(p+\epsilon)) \leq \exp\left(-m\frac{\epsilon^2}{2(p+\epsilon)}\right) \tag{29}$$

and for a sum S_m of m i.i.d. Bernoulli random variables Y_i with $\mathbb{P}(Y_1 = 1) = \mathbb{P}(Y_1 = -1) = \frac{1}{2}$ and some $x \in (0,1)$ we have:

$$\mathbb{P}(S_m \geq mx) \leq \exp(-mI(x)) \tag{30}$$

as well as

$$\lim_{m\to\infty} \frac{1}{m} log(\mathbb{P}(S_m \geq mx)) = -I(x) \tag{31}$$

where $I(x)$ is as defined as per the statement of the theorem. Note that equation (24) is known as Cramers theorem for fair ± 1 Bernoulli random variables.

Next, we let $0 < \alpha < \frac{1}{2}I(1-2\rho)$, $N = \exp(\alpha d) + 1$ and $0 < \beta_0 < 1$ be such that $I(\beta_0) = \alpha$. By the continuity of I [20] there exists an $\epsilon > 0$ such that for all $x \in (1-2\rho-\epsilon, 1-2\rho]$ we have that $\alpha < \frac{1}{2}I(x) \leq \frac{1}{2}I(1-2\rho)$. Again, since I is continuous and strictly increasing on [0,1), we can choose $0 < \beta < \beta_0 < 1$ such that:

$$\alpha - \frac{\epsilon}{2} = I(\beta_0) - \frac{\epsilon}{2} < I(\beta) < I(\beta_0) = \alpha \tag{32}$$

[20]I(x) is a continuous, convex bowl defined on $x \in (-1,1)$ with a range of $[0, 2log(2))$, which is striclty increase over [0,1).

Finally, we will define:

$$\begin{aligned}
A &= \{k \in \{2, \dots, N\} | \langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle \geq \beta d\} \\
p &= \mathbb{P}\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle \geq \beta d\right)
\end{aligned}$$

That is, A is the set of the indices of all stored patterns which have an inner product with $\tilde{\boldsymbol{x}}_1$ greater than or equal to βd, which we will call "significant overlap". Also, p is the probability that one or the stored patterns $\boldsymbol{x}_k$, which is randomly, uniformally sampled, is such that $k \in A$. Now, we have that for $\eta = \frac{1}{2}(\alpha - I(\beta)) > 0$ (by 25) and d sufficiently large we have, by (23) and (24) respectively:

$$p < e^{-dI(\beta)} \tag{33}$$

$$p > e^{-d(I(\beta)+\eta)} \tag{34}$$

Part 3 We now continue by finding an upper bound for the probability in the RHS of equation (21). Let P_q be the set of all subsets of the set $\{2, ..., N\}$ of size q. Then, note if a randomly generated pattern $\mathbf{x}_k$ is such that $k \notin A$, then we will have $\langle \mathbf{x}_k | \tilde{\mathbf{x}}_1 \rangle < \beta d$. Then, we have

$$\begin{aligned}
&\mathbb{P}\left(\sum_{k=2}^{N} e^2 \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)}\right) \\
&= \sum_{X \subset \{2,\dots,N\}} \mathbb{P}\left(\left(\sum_{k=2}^{N} \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)-2}\right) \cap (A = X)\right) \quad \text{(law of total probability)} \\
&\leq \sum_{q=0}^{N-1} \sum_{X \in P_q} p^q (1-p)^{N-1-q} \mathbb{P}\left(\sum_{k \in X} \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) + (N-1-q) e^{\beta d} > e^{d(1-2\rho)-2} | A = X\right) \quad \text{(Bayes)}
\end{aligned}$$

Then notice that since all patterns are randomly generated, and thus identically distributed, the probability that A is an arbitrary subset of $\{2, ..., N\}$ of size q is the same as the probability that $A = \{2, \dots, q+1\}$, therefore by the binomial theorem, the above is equal to

$$= \sum_{q=0}^{N-1} \binom{N-1}{q} p^q (1-p)^{N-1-q} \mathbb{P}\left(\sum_{k=2}^{q+1} \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)-2} - (N-1-q) e^{\beta d} | A = \{2, \dots, q+1\}\right)$$

We now take the pattern which has maximum overlap with $\tilde{\boldsymbol{x}}_1$ as an upper bound. We note two things: first, taking the maximum over the set $\{2, \dots, q+1\}$ inside the probability is equivalent to a union (either $\mathbf{x_2}$ or $\mathbf{x_3}$ or ... or $\mathbf{x_{q+1}}$ produces the maximum inner product), and secondly, since all $q+1$ patterns are identically distributed, the probability that either on of them produces the maximum inner product is the same. Using these facts, we get

$$\begin{aligned}
&\leq \sum_{q=0}^{N-1} \binom{N-1}{q} p^q (1-p)^{N-1-q} \mathbb{P}\left(\max_{k \in \{2,\dots,q+1\}} \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > \frac{1}{q}(e^{d(1-2\rho)-2} - (N-1-q) e^{\beta d}) | A = \{2, \dots, q+1\}\right) \\
&\leq \sum_{q=0}^{N-1} q \binom{N-1}{q} p^q (1-p)^{N-1-q} \mathbb{P}\left(\exp\left(\langle \boldsymbol{x}_2 | \tilde{\boldsymbol{x}}_1 \rangle\right) > \frac{1}{q}(e^{d(1-2\rho)-2} - (N-1-q) e^{\beta d}) | A = \{2, \dots, q+1\}\right)
\end{aligned}$$

where the last line arises as an application of Boole's inequality. Notice that in the final expression above, the probability statement which is conditioned on $A = \{2, \dots, q+1\}$ only depends on the index 2 (i.e. the pattern $\boldsymbol{x}_2$), therefore by Bayes theorem the above simplifies to

$$= \sum_{q=0}^{N-1} q\binom{N-1}{q} p^q (1-p)^{N-1-q} \frac{\mathbb{P}\left(\exp\left(\langle \boldsymbol{x}_2 | \tilde{\boldsymbol{x}}_1 \rangle\right) > \frac{1}{q}(e^{d(1-2\rho)-2} - (N-1-q)e^{\beta d}) \cap 2 \in A\right)}{\mathbb{P}(2 \in A)}$$
$$\leq \sum_{q=0}^{N-1} q\binom{N-1}{q} p^q (1-p)^{N-1-q} \left(\frac{r(q)}{p}\right)$$

where $r(q) = \mathbb{P}\left(\exp\left(\langle \boldsymbol{x}_2 | \tilde{\boldsymbol{x}}_1 \rangle\right) > \frac{1}{q}(e^{d(1-2\rho)-2} - (N-1-q)e^{\beta d})\right)$ gives a larger probability, since the event is less restrictive. In summary, we have:

$$\mathbb{P}\left(\sum_{k=2}^{N} e^2 \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)}\right) \leq \sum_{q=0}^{N-1} q\binom{N-1}{q} r(q) p^{q-1} (1-p)^{N-1-q} \tag{35}$$

Part 4 We will now split the sum in equation (28) into two parts, $q \in \{0, \dots, \lfloor 2p(N-1) \rfloor\}$ and the remaining, and bound each one separately. We will begin by bounding the second half of the sum first, using the identity $q\binom{N-1}{q} = (N-1)\binom{N-2}{q-1}$ and the trivial fact that $r(q) \leq 1$ to obtain

$$\sum_{q=\lfloor 2p(N-1) \rfloor + 1}^{N-1} q\binom{N-1}{q} r(q) p^{q-1} (1-p)^{N-1-q} \tag{36}$$
$$\leq (N-1) \sum_{q=\lfloor 2p(N-1) \rfloor + 1}^{N-1} \binom{N-2}{q-1} p^{q-1} (1-p)^{N-2-(q-1)} \tag{37}$$
$$= (N-1)\mathbb{P}(X_{N-2,p} \geq \lfloor 2p(N-1) \rfloor) \tag{38}$$
$$\leq (N-1)\mathbb{P}(X_{N-2,p} \geq 3\frac{p}{2}(N-2)) \tag{39}$$

where $X_{N-2,p}$ is, as per part 2 of the proof, a binomially distributed random variable[21]. Additionally, we have used the bound $\lfloor 2p(N-1) \rfloor > \frac{3}{2}p(N-2)$, which is a consequence of the fact that $p(N-1)$ goes to infinity as $d \to \infty$, which we will show later. Finally, if we have that if we choose $\epsilon = \frac{p}{2}$ then, by the result given by equation (22), we have

$$\sum_{q=\lfloor 2p(N-1) \rfloor + 1}^{N-1} q\binom{N-1}{q} r(q) p^{q-1} (1-p)^{N-1-q} \leq (N-1)\exp\left(-\frac{p(N-2)}{12}\right)$$

Part 5 We will now bound the first part of the sum in the RHS of equation (28). We first notice that

$$\frac{1}{q}(e^{d(1-2\rho)-2} - (N-1-q)e^{\beta d}) = \frac{1}{q}(e^{d(1-2\rho)-2} - e^{(\alpha+\beta)d}) + e^{\beta d}$$

by definition of N. Consequentially, we must have that $r(q)$ is increasing in q if $\alpha + \beta < 1 - 2\rho$[22] - we will prove this inequality later on. If we assume this to be the case then

[21]The inequality from lines (30) to (31) is simply the definition of the cumulative distribution function.

[22]we can ignore the -2 in the exponent since we are considering $d \to \infty$

$$\begin{aligned}
&\sum_{q=0}^{\lfloor 2p(N-1)\rfloor} \binom{N-1}{q} p^{q-1}(1-p)^{N-1-q} q r(q) \\
&\leq \frac{1}{p} \max_{q\in\{0,\ldots,\lfloor 2p(N-1)\rfloor\}} q r(q) \quad \text{(CDF of binomial} \leq 1) \\
&\leq 2(N-1)\cdot r(2p(N-1))
\end{aligned}$$

Now, recall from part 2 (equation (26)) of the proof that $p < e^{-dI(\beta)}$, therefore together with all of the above we observe that

$$\begin{aligned}
r(2p(N-1)) &= \mathbb{P}\left(\exp(\langle \mathbf{x}_2|\tilde{\mathbf{x}}_1\rangle) > \frac{e^{-\alpha d}}{2p}(e^{d(1-2\rho)-2} - e^{(\alpha+\beta)d}) + e^{\beta d}\right) \\
&= \mathbb{P}\left(\exp(\langle \mathbf{x}_2|\tilde{\mathbf{x}}_1\rangle) > \frac{1}{2p}(e^{d(1-2\rho-\alpha)-2} - e^{\beta d}) + e^{\beta d}\right) \\
&\leq \mathbb{P}\left(\exp(\langle \mathbf{x}_2|\tilde{\mathbf{x}}_1\rangle) > \frac{1}{2}(e^{d(1-2\rho-\alpha+I(\beta))-2} - e^{(\beta+I(\beta))d}\right)
\end{aligned}$$

We would now like to show, regarding the powers in the RHS of the inequality of the probability statement above, that $1-2\rho-\alpha+I(\beta) > \beta+I(\beta) \iff 1-2\rho-\alpha > \beta$ such that the first term dominates the second in the final expression above for sufficiently large d. By concavity we have for $x \in (0,1)$:

$$I(x) \leq log\left(\frac{(1+x)^2}{2} + \frac{(1-x)^2}{2}\right) = log(1+x^2) \leq x^2 \leq x$$

From this, and the definitions and inequalities established in part 2 of the proof, we obtain

$$\alpha+\beta < \alpha+\beta_0 = \alpha + I(\alpha) \leq 2\alpha < I(1-2\rho) \leq 1-2\rho$$

This proves that $1-2\rho-\alpha+I(\beta) > \beta+I(\beta)$ and also proves that $r(q)$ is increasing in q, as required above. Now, take γ such that $1-2\rho-\epsilon < \gamma < 1-2\rho-\frac{\epsilon}{2}$ for some $\epsilon > 0$. Then, continuing from the inequality above and using the result from equation (25), we have

$$1-2\rho-\epsilon < \gamma < 1-2\rho-\alpha+I(\beta)$$

which in turn leads us to conclude that, for d sufficiently large and by the result given by equation (24), we get:

$$\begin{aligned}
r(2p(N-1)) &\leq \mathbb{P}\left(\exp(\langle \boldsymbol{x}_2|\tilde{\boldsymbol{x}}_1\rangle) > e^{\gamma d}\right) \\
&\leq \exp(-dI(\gamma))
\end{aligned}$$

Part 6 Bringing everything together we get:

$$
\begin{aligned}
&\mathbb{P}\left(\exists k\, \exists i : T_i(\tilde{\mathbf{x}}_k) \neq x_i^k\right) \\
&\leq d \cdot N \mathbb{P}\left(\sum_{k=2}^{N} e^2 \exp\left(\langle \boldsymbol{x}_k | \tilde{\boldsymbol{x}}_1 \rangle\right) > e^{d(1-2\rho)}\right) \\
&\leq d \cdot N \left[2(N-1)\exp(-dI(\gamma)) + (N-1)\exp\left(-\frac{p(N-2)}{12}\right)\right] \\
&= d \cdot N \left[\frac{2}{(N-1)}\exp\left(-d(I(\gamma) - 2\alpha)\right) + \frac{1}{(N-1)}\exp\left(2d\alpha - \frac{N-2}{N-1}\frac{p(N-1)}{12}\right)\right] \\
&\leq \frac{2d \cdot N}{(N-1)}\exp\left(-d(I(\gamma) - 2\alpha)\right) + \frac{d \cdot N}{(N-1)}\exp\left(2\alpha - \frac{N-2}{N-1}\frac{p(N-1)}{12}\right)
\end{aligned}
$$

Finally, to see that this upper bound vanishes in the limit, notice that by definition of γ we have $I(\gamma) > 2\alpha$, so the first term clearly tends to zero. For the second term, by using the lower bound on p - equation (27) - we get:

$$p(N-1) \geq \exp(d(\alpha - I(\beta) - \eta))$$

But we know, by definition, that $I(\beta) < I(\beta_0) = \alpha$ and $\eta = \frac{1}{2}(\alpha - I(\beta))$, so $\alpha - I(\beta) - \eta > 0$. Therefore, the second term also tends to zero. Also, a straightforward consequence of the bound above is that $p(N-1)$ goes to infinity as $d \to \infty$ and therefore confirms the assumption used above that $\lfloor 2p(N-1) \rfloor > \frac{3}{2}p(N-2)$. In conclusion we have that, so long as $\alpha < I(1-2\rho)/2$, we obtain

$$\mathbb{P}\left(\exists k\, \exists i : T_i(\tilde{\mathbf{x}}_k) \neq x_i^k\right) \to 0 \quad \text{as } d \to \infty$$

which concludes the proof. $\square$

Appendix C: Code

All code used to generate the results for this report may be found at:

https://github.com/JeremyDouglas91/Hopfield-Networks