Universal Hopfield Networks: A General Framework for Single-Shot Associative Memory Models

Beren Millidge [1] **Tommaso Salvatori** [2] **Yuhang Song** [1 2] **Thomas Lukasiewicz** [3 2] **Rafal Bogacz** [1]

Abstract

A large number of neural network models of associative memory have been proposed in the literature. These include the classical Hopfield networks (HNs), sparse distributed memories (SDMs), and more recently the modern continuous Hopfield networks (MCHNs), which possess close links with self-attention in machine learning. In this paper, we propose a general framework for understanding the operation of such memory networks as a sequence of three operations: *similarity*, *separation*, and *projection*. We derive all these memory models as instances of our general framework with differing similarity and separation functions. We extend the mathematical framework of Krotov & Hopfield (2020) to express general associative memory models using neural network dynamics with local computation, and derive a general energy function that is a Lyapunov function of the dynamics. Finally, using our framework, we empirically investigate the capacity of using different similarity functions for these associative memory models, beyond the dot product similarity measure, and demonstrate empirically that Euclidean or Manhattan distance similarity metrics perform substantially better in practice on many tasks, enabling a more robust retrieval and higher memory capacity than existing models.

1. Introduction

Associative, or 'semantic', memories are memory systems where data points are retrieved not by an explicit address, but by making a query to the system of approximately the same type as the data points that it stores. The system then returns the closest data point to the query according to some metric. For instance, an associative memory system, when given an image, can be used to return other 'similar' images. It is often argued that the brain similarly stores and retrieves its own memories (Hinton & Anderson, 2014; Rolls, 2013; Tsodyks & Sejnowski, 1995), as it is a common experience to be able to recall a memory given a partial cue, e.g., recalling a song given just a few notes (Bonetti et al., 2021). A large literature of neuroscience and computational theories has developed models of how such associative memory systems could be implemented in relatively biologically plausible neural network architectures (Kanerva, 1988; 1992; Hopfield, 1982; Hinton & Anderson, 2014).

Two classical and influential models are the Hopfield network (HN) (Hopfield, 1982; 1984) and the sparse distributed memory (SDM) (Kanerva, 1988; 1992; Jaeckel, 1989). More recently, they have been generalized to the modern-continuous Hopfield network (MCHN) (Ramsauer et al., 2020) and the modern-continuous sparse distributed memory (MCSDM) (Bricken & Pehlevan, 2021), which have substantially improved performance, close relationships with transformer attention, and can handle continuous inputs.

Here, we propose a unified framework that encompasses all these models as simple instantiations of a more general framework, which we call *universal Hopfield networks (UHNs)*. Mathematically, the UHN can be described as a function $\text{UHN}\colon \mathbb{R}^I \to \mathbb{R}^O$ mapping a vector in an input space of dimension I to a vector in an output space of dimension O, with two additional inputs of a memory matrix M of size $N \times I$, consisting of a set of N stored patterns, and a projection matrix P of size $O \times N$, consisting of a potentially different set of stored patterns with dimension O for heteroassociation. The dimensionality of the input and output patterns are allowed to differ to enable heteroassociative memories to be described in the same framework. For autoassociative memories $I = O$. The UHN function can be factorized into a sequence of three operations: *similarity*, *separation*, and *projection*, illustrated in Figure 1. First, *similarity* matching between the query and a stored set of memory vectors to produce a vector of similarity scores. Second, *separation* to numerically magnify small differences in original similarity scores into large differences in

[1]MRC Brain Network Dynamics Unit, University of Oxford, UK. [2]Department of Computer Science, University of Oxford, UK. [3]Institute of Logic and Computation, TU Wien, Austria. Corresponding author: Yuhang Song <yuhang.song@some.ox.ac.uk>.
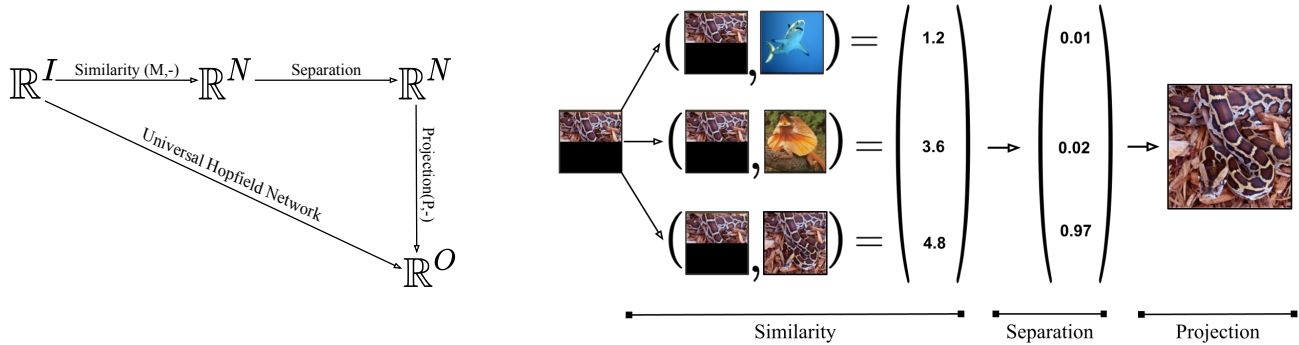
Figure 1. Left: Schematic of the key equations that make up the general theory of the abstract Hopfield network, which shows the factorization of a UHN into *similarity, separation*, and *projection*. Right: Visual representation of the factorization diagram when performing an associative memory task on three stored memories. The corrupted data point is scored against the three memories (*similarity*). The difference in scores are then exaggerated (*separation*), and used to retrieve a stored memory (*projection*).

the output scores so as to increase the relative separation of the scores, and finally *projection*, which multiplies the resulting set of output similarity scores with a projection matrix, and constructs an output based essentially on a list of stored data points in the memory[1] weighted by the output similarity scores, so that the network's output is most influenced by memories similar to the query vector. The main contributions of this paper are briefly as follows.

- We define a general framework of *universal Hopfield networks*, which clarifies the core computation underlying single-shot associative memory models.

- We demonstrate how existing models in the literature are special cases of this general framework, which can be expressed as an extension of the energy-based model proposed by Krotov & Hopfield (2020).

- We demonstrate that our framework allows straightforward generalizations to define novel associative memory networks with superior capacity and robustness to MCHNs by using different similarity functions.

It is also important to draw a distinction between feedforward and iterative associative memory models. In feedforward models, memory retrieval is performed through a fixed computation mapping a query to its retrieved output. Examples of feedforward associative memory models include the DAM (Krotov & Hopfield, 2016), the MCHN (Ramsauer et al., 2020), and the MCSDM (Bricken & Pehlevan, 2021), which effectively instantiate 2-layer MLPs and perform retrieval as a feedforward pass through the network. Conversely, iterative associative memory models retrieve memories by either iterating over neurons (Hopfield, 1982; Demircigil et al., 2017) or else iterative over multiple forward passes of the network by feeding back the output into the network as a new input. It has been shown empirically

that standard autoencoder networks (Radhakrishnan et al., 2018; 2020; Jiang & Pehlevan, 2020) and predictive coding networks (Salvatori et al., 2021) can store memories as fixed points of these dynamics. In Section 3 and our experiments, we primarily investigate feedforward associative memories, while in Section 4 we derive a general framework and energy function that can support both feedforward and iterative associative memory models.

The rest of this paper is organized as follows. In Section 2, we define the mathematical framework of *universal Hopfield networks*. In Section 3, we show how existing models can be derived as special cases of our framework. In Section 4, we extend the neural model of Krotov & Hopfield (2020) to define an energy function and associated neural dynamics for the UHN. In Section 5, we show that our framework enables generalization to novel similarity and separation functions, which result in higher capacity and more robust networks, while experiments on the separation functions empirically confirm theoretical results regarding the capacities of associative memory models.

2. Universal Hopfield Networks (UHNs)

A single-shot associative memory can be interpreted as a function that takes an input vector q (ideally, a corrupted version of a data point already in memory) and outputs a vector corresponding to the closest stored data point. Mathematically, our framework argues that every feedforward associative memory in the literature admits the following factorization, which defines an abstract and general *universal Hopfield network (UHN)*:

$$z = \underbrace{P}_{\text{Projection}} \cdot \underbrace{\text{sep}}_{\text{Separation}} \underbrace{(\text{sim}(M, q))}_{\text{Similarity}}, \qquad (1)$$

where z is the $O \times 1$ output vector of the memory system, P is a projection matrix of dimension $O \times N$, *sep* is the

---

[1]For heteroassociative memory models, a separate projection memory is used containing the outputs associated with each input.

separation function, sim is the similarity function, M is an $N \times I$ matrix of stored memories or data points, and q is the query vector of dimension $I \times 1$.

The intuition behind this computation is that, given an input query, we first want to rank how similar this query is to all the other stored memories. This is achieved by the *similarity* function, which outputs a vector of similarity scores between each data point held in the memory and the query. Given these similarity scores, since we will ultimately be retrieving based on a linear combination of the patterns stored in the projection matrix, weighted by their similarity scores, and we ultimately only want to produce one clear output pattern without interference from the other patterns, then we need a way to emphasize the top score and de-emphasize the rest. This is achieved by the *separation* function. It is well known that separation functions of higher polynomial degrees lead to capacity increases of the order of $C \propto N^{n-1}$, where N is the number of visible (input) neurons, and n is the order of the polynomial (Chen et al., 1986; Horn & Usher, 1988; Baldi & Venkatesh, 1987; Abbott & Arian, 1987; Caputo & Niemann, 2002; Krotov & Hopfield, 2016), while exponential separation functions (such as the softmax) lead to exponential memory capacity (Demircigil et al., 2017; Ramsauer et al., 2020). Taking this further, it is clear to see that simply using a *max* separation function leads to a theoretically unbounded capacity in terms of the dimension of the query vector, since then presenting an already stored pattern as a query will always return itself as a memory. However, the 'attractors' in such a network grow increasingly small so that, in practice, the real bound on performance is not the capacity but rather the ability of the similarity function to distinguish between the query and various possible stored patterns — a pattern that is clear already with the 'exponential' capacity MCHN network, which despite its theoretical exponential capacity often performs relatively poorly at retrieval in practice with corrupted or noisy queries. Finally, the *projection* matrix takes the vector of separated similarity scores and maps it to the correct output expected of the network.

Importantly, Equation 1 can be interpreted as a feedforward pass through an artificial neural network with a single hidden layer, where the activation function of the first layer is the separation function, and the activation function of the output is linear or else is some post-processing function such as binarization (as in the classical HN). Interpretations of memory networks in this way have been independently proposed by Kanerva (1988) for SDM and recently by Krotov (2021) for the MCHN (Ramsauer et al., 2020). Furthermore, coming from the other direction, recent work has also begun to suggest that standard 2-layer multi-layer perceptrons (MLPs) may naturally tend to function as associative memory models in practice. For instance, Geva et al. (2020) show that the feedforward layers of the transformer appear

to serve as key-value memories (Geva et al., 2020), and it has been suggested that these feedforward layers can be replaced with simply persistent memory vectors (Sukhbaatar et al., 2019).

3. Instances of Universal Hopfield Networks

Now that we have defined our universal Hopfield network (UHN), we shall show how the currently existing main associative memory models can be derived as specific instances of the UHN. The equivalences are summarized in Table 1.

3.1. Hopfield Networks

Hopfield networks (HNs) (Hopfield, 1982; 1984) consist of a single neural network layer that stores an array of binary memories $M = [m_1, m_2, \ldots, m_N]$, where M is an $N \times I$ matrix, and I is the dimension of each memory vector, and N is the number of memories stored. The memory arrays are then stored in a synaptic weight matrix $W = MM^T$. Memories are retrieved by fixing the input neurons to a query pattern q, which is a binary vector of length I. While the original HN of Hopfield (1984) iteratively minimized the energy function over individual neurons, we here describe the 'feedforward' Hopfield networks described in (Krotov & Hopfield, 2016; Little, 1974), which retrieve memories by performing a forward pass through the network to compute an output $z = \text{sign}(W \cdot q)$, where $sign$ is the sign function, and z is the retrieved pattern and is also a binary vector of length I (since the HN is autoassociative). This process can be repeated if necessary to further minimize the energy by feeding in the reconstructed output again to the network as its input. This network can be interpreted as minimizing a 'Hopfield energy function', which is equivalent to the energy function of an Ising spin-glass model (Kirkpatrick & Sherrington, 1978; Keeler, 1988). To show that the HN is an example of a UHN, first recall that the synaptic weight matrix in the HN is defined not as the stored pattern matrix but as the outer product $W = MM^T$. By substituting this into the HN update rule, we obtain $z = \text{sign}((MM^T) \cdot q) = \text{sign}(M\mathcal{I}(M^T \cdot q))$, where we use $\mathcal{I}$ to denote the identity function. Thus, we can understand the HN within our framework as using a dot-product similarity function and an identity separation function (which is the cause of the HN's relatively poor storage capacity). The sign function plays no part in memory retrieval and simply binarizes the network's output.

3.2. Sparse Distributed Memories

Sparse distributed memories (Kanerva, 1988; 1992) (SDM) are designed to heteroassociate long binary vectors. The network consists of two matrices — an 'Address' matrix and a 'Pattern' matrix. Memories are thought of as being stored in a data-type with both an 'Address' and a 'Pattern' pointer.

Memory Network	Similarity Function	Separation Function
(Classical) Hopfield Network (HN)	Dot Product	Identity
Sparse Distributed Memory (SDM)	Hamming Distance	Threshold
Dense Associative Memory (DAM)	Dot Product	Polynomial
Modern Continuous Hopfield Network (MCHN)	Dot Product	Softmax

Table 1. Associative memory models.

To retrieve a memory, a query vector is compared against all stored addresses in the Address matrix, and the binary Hamming distance between the query and all addresses is computed. Then, a certain number of addresses are activated that are below a threshold Hamming distance from the query. The memory is retrieved by summing the pattern pointers for all the addresses activated by the query. The 'read' phase of the SDM (Kanerva, 1988) can be written mathematically as $P \cdot \text{thresh}(d(M, q))$, where d is the Hamming distance function, and $thresh$ is a threshold function that returns 1 if the Hamming distance is greater than some threshold, and 0 otherwise. Here, it is clear that the SDM can be naturally understood using our framework with similarity function d (Hamming distance) and separation function $thresh$, which implements a top-k operation to cut out poor matches.

3.3. Dense Associative Memories and Modern Continuous Hopfield Networks

In recent years, the capabilities of both of these classical memory models have been substantially improved, and a number of new Hopfield architectures have been developed based on the dense associative memory (DAM) initially proposed by Krotov & Hopfield (2016) and extended by Demircigil et al. (2017). Specifically, Krotov & Hopfield (2016) argued for generalizing the standard Hopfield energy function ($E = q^T W q + q^T b$) (where b is an $I \times 1$ bias vector to convert between binary and bipolar representations) to an arbitrary function of q and W: $E = F(W \cdot q)$ and showed that as F becomes a polynomial of increasing order, the memory storage capacity of the network increases as $C \propto I^{n-1}$, where I is the number of visible neurons, and n is the order of the polynomial. Demircigil et al. (2017) extended this argument to exponential energy functions of the form $E = \sigma(W \cdot q)$, where $\sigma(x)$ is the softmax function, and showed that the resulting networks have *exponential* storage capacity. Then, Ramsauer et al. (2020) generalized these networks to continuous (instead of binary) inputs to derive the modern continuous Hopfield network (MCHN). The MCHN uses the energy function $E = q^T q + \text{logsumexp}(Wq)$, which can be minimized with the convex-concave procedure (Yuille & Rangarajan, 2003), giving the update rule $z = W^T \sigma(Wq)$, which enables exponential capacity, memory retrieval in a single step, and is extremely similar to the feedforward pass of a self-attention unit $z = V\sigma(KQ)$ with 'Query Matrix' Q, 'Key Matrix'

K, and 'Value Matrix' V, where we can associate $Q = q$, $K = W$, and $V = W$ (Bahdanau et al., 2014; Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020; Radford et al., 2019). Lastly, Krotov & Hopfield (2020) presented a unified set of neural dynamics that can reproduce the original HN, the polynomial interaction functions of (Krotov & Hopfield, 2016), and the exponential Hopfield network of (Demircigil et al., 2017; Ramsauer et al., 2020), using only local computations, and which Tang & Kopp (2021) have shown also to be related to the spherical normalization dynamics in the recent MLP-mixer (Tolstikhin et al., 2021).

3.4. Continuous Sparse Distributed Memories

Recent work has also uncovered a close link between SDMs and transformer attention (Bricken & Pehlevan, 2021). Recall that the SDM read rule can be expressed as $P \cdot thresh(d(A, q))$, where $thresh$ is a threshold function, A is an $M \times N$ matrix of addresses, P is a $K \times O$ matrix mapping each stored data point to its associated pattern, and d is the Hamming distance between each of the stored addresses in A and the query pattern q. Bricken & Pehlevan (2021) first generalized SDM from binary vectors to the 'continuous SDM', where P, A, and q contain real values instead of bits. Then, they replaced the Hamming distance (which only applies to binary vectors) with the dot product, using the argument that the Hamming distance is the dot product (mod 2) of binary vectors, and thus that the dot product is the natural generalization of the Hamming distance to continuous variables. Finally, they noted that the decay of the number of addresses that are not cutoff by the threshold function decreases approximately exponentially as the Hamming distance threshold decreases. The mathematical reason for this is that the distribution of addresses within a given Hamming distance of a query is a binomial distribution, which can be well approximated with a Gaussian at large N, and the tails of a Gaussian distribution decay exponentially. This approximately exponential decay in the number of addresses passing the threshold allows us to heuristically replace the threshold function with an exponential function, resulting in the following approximate update rule for the 'continuous SDM' model $z = P\sigma(Aq)$, which is closely related to the self-attention update rule and is identical to the rule for the MCHN.

3.5. Auto- and Heteroassociative Memories

Our framework also provides a simple explanation of the difference between autoassociative memories (which map a corrupted version of a memory to itself) and heteroassociative memories (which map some input memory to some other memory type, potentially allowing for memory chains and sequence retrieval): namely, that autoassociative memories set the projection matrix P equal to the memory matrix M, i.e., one recalls the memories used for similarity matching, while heteroassociative memory networks set the projection matrix equal to the associated heteroassociated memory. It is thus clear to see why the HN and MCHN networks are autoassociative, and how to convert them to heteroassociative memory networks. Namely, for the MCHN, set the update rule to $z = P\sigma(M.q)$, and for the HN set the weight matrix $W = PM^T$. Demonstrations of these novel heteroassociative HNs and MCHNs are given in Appendix B. Interestingly, the heteroassociative MCHN update rule is equivalent to the self-attention update rule found in transformer networks (Vaswani et al., 2017), and thus suggests that the fundamental operation performed by transformer networks is heteroassociation of inputs (the queries) and memories (the keys) with other memories (the values).

4. Neural Dynamics

In this section, extending the work of Krotov & Hopfield (2020), we present an abstract energy function for the UHN and set of neural dynamics that minimize it, which can be specialized to reproduce any of the associative memory models in the literature. By framing associative memory models in terms of an energy function, we can describe the operation of both iterative and feedforward associative memory models, as well as mathematically investigate the properties of the fixed points that they use as memories. We define a general neural implementation and energy function for our abstract associative memory model that uses only local interactions. In this model, there are two types of 'neurons': 'value neurons' v and 'memory neurons' h. We can think of the 'value neurons' v being initialized to the query pattern q such that $v^{t=0} = q$, and then updated to produce the output pattern z. This is because the UHN effectively implements a two-layer artificial neural network where the value neurons are the input layer and the memory neurons are the hidden layer. The memory and value neurons are interconnected by the memory matrix M. The neural activities v and h are also passed through two activation functions g and f such that $f = f(h)$ and $g = g(v)$. The network has the following recurrent neural dynamics:

$$\tau_v \frac{dv_i}{dt} = -\frac{\partial E}{\partial v_i} = \sum_j \frac{\partial \text{sim}(M_{i,j}, v_i)}{\partial v_i} f_i - v_i g'(v_i) \quad (2)$$

$$\tau_h \frac{dh_i}{dt} = -\frac{\partial E}{\partial h_i} = f'(h_i)\left[\sum_j \text{sim}(M_{i,j}, v_i) - h_i\right],$$

where τ_v and τ_h are time-constants of the dynamics. These dynamics can be derived from the energy function:

$$E(M, v, h) = \left[\sum_i (v_i - I_i)g_i - \mathcal{L}_v\right] +$$
$$\left[\sum_i f_i h_i - \mathcal{L}_h\right] - \sum_i \sum_j f_i \text{sim}(M_{i,j}, v_i), \quad (3)$$

where we define the 'Lagrangian' functions $\mathcal{L}_v$ and $\mathcal{L}_h$ such that their derivatives are equal to the activation functions $g = \frac{\partial \mathcal{L}_v}{\partial v}$ and $f = \frac{\partial \mathcal{L}_h}{\partial h}$. The energy function is defined such that it only includes second-order interactions between the value and memory neurons in the third term, while the first two terms in square brackets only involve single sets of neurons. In Appendix A, we show that the energy function is a Lyapunov function of the dynamics, i.e., it always decreases over time, as long as the Hessian of the activation functions f and g are positive definite. To obtain the dynamics equations (2) and (3), we first define function f to be the separation function $f(h) = \text{sep}(h)$ such that $\mathcal{L}_h = \int dh \text{sep}(h)$ and denote $f'(h) = \frac{\partial f(h)}{\partial h}$ as the derivative of f. We also set $\mathcal{L}_v = \frac{1}{2}\sum_i v_i^2$, which implies $g_i = \frac{\mathcal{L}_v}{\partial v_i} = v_i$. A further simplification of the energy and dynamics occurs if we assume that τ_h is small, and thus the dynamics of the hidden neurons are fast compared to the value neurons such that we can safely assume that these dynamics have converged. This allows us to write $h^* = \sum_j \text{sim}(M_{i,j}, v_i)$, since when setting $\frac{dh_i}{dt} = 0$, we can cancel the $f'(h_i)$ terms as long as $f'(h) \neq 0$, which is true for all separation functions we consider in this paper except the max function, which is therefore heuristic. This gives us the simpler and intuitive energy function:

$$E = \sum_i v_i^2 - \frac{1}{2}\sum_i v_i^2 + \sum_i f_i \sum_j \text{sim}(M_{i,j}v_i)$$
$$- \mathcal{L}_h - \sum_i \sum_j f_i \text{sim}(M_{i,j}, v_i)$$
$$= \sum_i \frac{1}{2}v_i^2 - \int \text{sep}(\sum_j \text{sim}(M_{i,j}, v_i)), \quad (4)$$

where the integral is over the input to the separation function. It is now straightforward to derive the HN and MCHN. To do so, we set $\text{sim}(M, v) = Mv$ and $\text{sep}(x) = x$ for the HN, and $\text{sep}(x) = \frac{e^x}{\sum e^x}$ for the MCHN. Following Krotov & Hopfield (2020), for the MCHN, we can derive a single step of the dynamics by taking the gradient of the energy:

$$E = \sum_i \frac{1}{2}v_i^2 - \log\sum_j e^{(\text{sim}(M_{i,j}, v_i))} \quad (5)$$

$$\tau_v \frac{dv_i}{dt} = -\frac{\partial E}{\partial v_i} = -v_i + \frac{e^{\sum_j \text{sim}(M_{i,j}, v_i)}}{\sum_i e^{\sum_j \text{sim}(M_{i,j}, v_i)}}\frac{\partial \text{sim}(M_{i,j}v_i)}{\partial v_i}.$$

If we then perform an Euler discretization of these dynamics and set $\Delta t = \tau_v = 1$, then we obtain the following update step:

$$v_i^{t+1} = M\sigma(Mv_i^t), \quad (6)$$

where $\sigma(x) = \frac{e^x}{\sum e^x}$ is the softmax function by using the fact that the MCHN uses the dot product similarity function $\text{sim}(M, v) = Mv$. It was proven in (Ramsauer et al., 2020) that this update converges to a local minimum in a single step. This thus derives the MCHN update

$$v^* = M\sigma(Mq), \qquad (7)$$

since $v^{t=0} = q$ as the visible neurons are initialized to the input query and equilibrium occurs after a single step. Similarly, to derive the HN, we set the separation function to the identity ($\text{sep}(x) = x$) and similarity function to the dot product and using the fact that $\int d(Mv)\text{sep}(\text{sim}(Mv)) = \int d(Mv)Mv = \frac{1}{2}(Mv)^2$, thus resulting in the energy function and equilibrium update rule:

$$E = \sum_i \frac{1}{2}v_i^2 - \frac{1}{2}\sum_i\sum_j v_i M_{i,j} M_{i,j}^T v_i$$

$$\tau_v \frac{dv_i}{dt} = -\frac{\partial E}{\partial v_i} = -v_i + \sum_i\sum_j M_{i,j} M_{i,j}^T v_i, \quad (8)$$

where, again, if we perform the Euler discretization, we obtain the following update step:

$$v^{t+1} = MM^T v^t, \qquad (9)$$

which, with a final normalizing $sign$ function to binarize the output reconstruction, is identical to the HN update rule. We thus see that using this abstract energy function, we can derive a Lyapunov energy function and associated local neural dynamics for any associative memory model that fits within our framework. Moreover, our framework also describes iterative associative memory models if these inference dynamics are integrated over multiple steps instead of converging in a single step.

5. Experiments

Our general framework allows us to define an abstract associative memory model with arbitrary similarity and separation functions, as well as a set of neural dynamics and associated energy function for that model. A natural question is whether we can use this abstract framework to derive more performant associative memory models by using different similarity and separation functions. In this section, we empirically test a wide range of potential separation and similarity functions on associative memory retrieval tasks. We find similarity functions such as the Manhattan (or absolute or $l1$ norm) distance metric perform substantially better than the dot product distance used in the MCHN across a datasets and is more robust to input distortion. We define novel associative memory models with state-of-the-art performance, which can scale beyond that considered previously in the literature, especially on the Tiny ImageNet dataset. In Appendix E, we discuss the detailed numerical normalizations and other implementation details that are necessary to achieve a good performance in practice.

5.1. Capacity under Different Similarity Functions

We investigate the capacity of the associative memory models to increasing numbers of stored memories on a suite of potential similarity functions. The similarity and separation functions tested are defined in Appendix D. We tested the retrieval capacity on three image datasets: MNIST, CIFAR10, and Tiny ImageNet. All images were normalized such that all pixel values lay between 0 and 1. Before presenting the images to the network as queries, they were flattened into a single vector. When masking the images, the masked out pixels were set to 0. When adding Gaussian noise to the images, we clipped the pixel values after noise was added to maintain all values between 0 and 1.

From Figure 2, we see that the similarity function has a large effect on the memory capacity of the associative memory models. Empirically, we see very robustly that the highest performing and highest capacity similarity function is the Manhattan distance $\text{sim}(M, q) = \text{abs}(M - q)$, where the subtraction is taken over rows of the memory matrix. Moreover, the superiority of the Manhattan distance as a similarity function appears to grow with the complexity of the dataset. It is roughly equivalent to the Euclidean and dot product on MNIST, slightly better on CIFAR10 and substantially better on Tiny ImageNet. The Euclidean distance also performs very well across image datasets. Other potential measures such as the KL divergence, Jensen-Shannon distance, and reverse KL perform substantially worse than simple Euclidean, dot-product, and Manhattan distance measures. The dot product metric used in the MHCN also performs very well, although it must be carefully normalized (see Appendix E). Interestingly, we see stable levels of performance for increasing capacity for a given similarity function across a wide range of memory capacities.

The similarity functions are so important, because they are the fundamental method by which the abstract associative memory model can perform ranking and matching of the query to memory vectors. An ideal similarity function would preserve a high similarity across semantically non-meaningful transformations of the query vectors (i.e., insensitive to random noise, perturbations, and masking of parts of the image), while returning a low similarity for transformed queries originating from other memory vectors. An interesting idea is that, while thus far we have used simple similarity functions such as the dot product and the Euclidean distance, it is possible to define smarter distance metrics native to certain data types, which should be expected to give an improved performance. Moreover, it may be possible to directly *learn* useful similarity functions by defining the similarity function itself as a neural network trained on a contrastive loss function to minimize differences between variants of the same memory and maximize differences between variants of different ones.
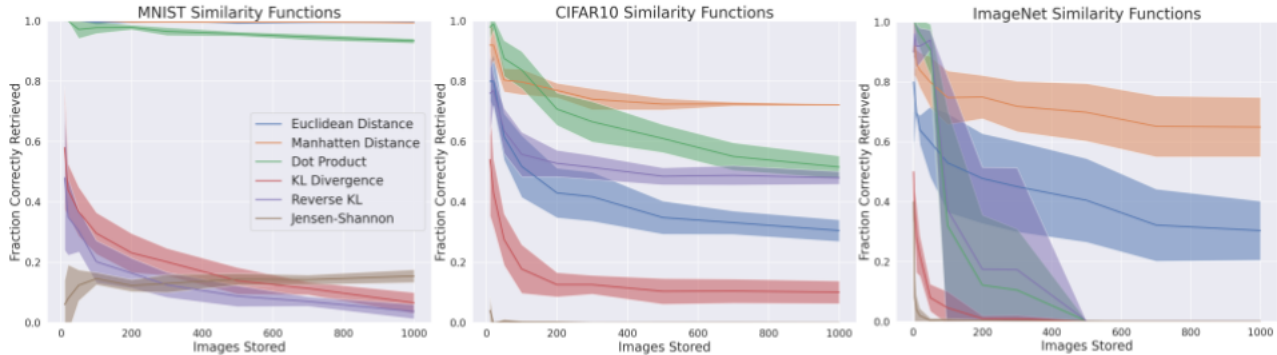
Figure 2. Capacity of the associative memory networks with different similarity functions, as measured by increasing the number of stored images. The capacity is measured as the fraction of correct retrievals. To test retrieval, the top-half of the image was masked with all zeros (this is equivalent to a fraction masked of 0.5 in Figure 4) and was then presented as the query vector for the network. A retrieval was determined to be correct if the summed squared difference between all pixels in the retrieved image and the true reconstruction was less than a threshold T, which was set at 50. The queries were presented as the stored images corrupted with independent Gaussian noise with a variance of 0.5. Mean retrievals over 10 runs with different sets of memories images. Error bars are computed as the standard deviations of the correct retrievals of the 10 runs. A softmax separation function was used with a β parameter of 100.
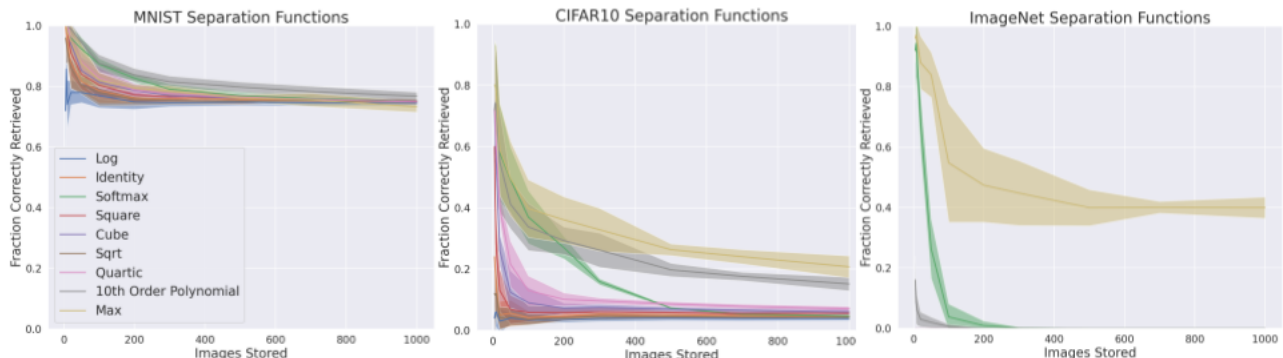

Figure 3. The retrieval capacity of the network on retrieving half-masked images using the dot-product similarity function. Plotted are the means and standard deviations of 10 runs. A query was classed as correctly retrieved if the sum of squared pixel differences was less than a threshold of 50.

5.2. Capacity under Different Separation Functions

In Figure 3, we considered the effect of the separation function on retrieval capacity by measuring the retrieval performance using a fixed similarity function (dot product) for different separation functions (defined in Appendix D). The empirical effect of the separation function on capacity appear to align closely with known theoretical results (Demircigil et al., 2017; Keeler, 1988; Abu-Mostafa & Jacques, 1985; Ma, 1999; Wu et al., 2012). Namely, that the exponential and max functions have substantially higher capacity than that of other separation functions and that low-order polynomial or lesser separation functions suffer a very rapid decline in retrieval performance as capacity increases. High-order polynomials perform very well, as predicted by the mathematical capacity results in (Krotov & Hopfield, 2016; Demircigil et al., 2017). Here, the softmax performs relatively poorly compared to the 10th order polynomial due to the β parameter in the softmax being set to 1, which was

done for a fair comparison to other methods. However, as $\beta \to \infty$, the softmax function tends to the max, so the relative performance of the softmax can be increased by simply increasing β. The importance of the separation functions, and especially using 'high-powered' separation functions such as softmax, max, and a 10th order polynomial increases with the complexity of the data. This is due to the greater level of interference caused by more complex and larger images, which requires a more powerful separation function to numerically push apart the similarity scores.

5.3. Retrieval under Different Similarity Functions

We also tested (Figure 4) the effect of the similarity function on the retrieval capacity of the network for different levels of noise or masking of the query vector, a proxy for the robustness of the memory network. We tested the retrieval capacity on two types of query perturbation: Gaussian noise and masking. In the first case, independent zero-mean Gaus-

sian noise with a specific noise variance σ was added elementwise to the query image. As the image pixel values were restricted to lie in the range $[0, 1]$, a σ of 1 results in a huge distortion of the original image. With masking, the top k fraction of pixels were set to 0. A fraction of 0.9 results in only the bottom 10% of the image being visible in the query vector. Example visualizations different noise levels and masking fractions are given in Appendix C.

We observe in Figure 4 that the used similarity functions have strong effects on the robustness of the retrieval under different kinds of perturbations. For independent Gaussian noise, it largely appears that the dot product similarity measures allow for relatively robust reconstructions even up to very high levels of noise, which would make the queries uninterpretable to humans (see Appendix C). The Manhattan distance similarity metric, however, performs better under masking of the image, definitely for relatively small fractions masked, although for Tiny ImageNet, the dot-product similarity function appears to be more robust to extremely high masking fractions. Overall, it appears that the similarity function plays a large role in the degree of robustness of the memory to corrupted queries, but that the same few similarity functions, such as dot product and Manhattan distance, consistently perform well across a range of circumstances.

6. Discussion

In this paper, we have proposed a simple and intuitive general framework that unifies existing single-shot associative memory models in the literature. Moreover, we have shown that this scheme comes equipped with a set of local neural dynamics and that leads immediately to useful generalizations in terms of varying the similarity function, which led to the discovery of the superior performance of Manhattan distance, which outperforms the state-of-the-art MCHN at various retrieval tasks with complex images. Finally, our general framework lets us see the natural and clear relationship between auto- and heteroassociative memory models, which amounts entirely to the selection of the projection matrix P, a fact that has often been unclear in the literature.

Our framework gives a clear insight into the two key steps and bottlenecks of current associative memory models. The major bottleneck is the similarity function, which is fundamental to the retrieval capacity and performance of the model, and it is the similarity metric that, assuming a sufficiently powerful separation function, is the main determinant of retrieval performance, especially of noisy or corrupted queries. Here, we only considered single-layer networks, which apply the similarity function directly to raw image data. However, performance may be increased by first feeding the raw queries through a set of preprocessing steps or, alternatively, an encoder network trained to produce a useful latent representation of the input, and then performing associative memory on the latent representations.

This naturally leads to a hierarchical scheme for associative memories models, which will be explored in future work. This scheme also has close associations with the field of metric learning (Kulis et al., 2013; Yang, 2007), where we consider the similarity function as defining a metric on the underlying data geometry and then the associative memory simply performs nearest-neighbour matching with this metric. Using predefined similarity functions corresponds to directly defining a metric on the space, however, using a deep neural network to map into a latent space and then performing a similarity scoring on that latent space is equivalent to a learnable metric which is implicitly parametrized by the deep neural network encoder (Kaya & Bilge, 2019).

A conceptual benefit of our framework is that it makes clear that single-shot associative memory models are simply two-layer MLPs with an unusual activation function (i.e., the separation function), which works best as a softmax or max function, and where the weight matrices directly encode explicit memory vectors instead of being learnt with backpropagation. This leads immediately to the question of whether standard MLPs in machine learning can be interpreted as associative memories instead of hierarchical feature extractors. A crucial requirement for the MLP to function as an associative memory appears to be a high degree of sparsity of intermediate representations (ideally one-hot output) so that an exact memory can be reconstructed instead of a linear combination of multiple memories. With a dense representation at the intermediate layers, no exact memory can be reconstructed, and the network will instead function as a feature detector. This continuum between associative memories and standard MLPs, which depends on the sparsity of the intermediate representations, has resonances in neuroscience, where neural representations are typically highly sparse, as well as to helps contextualize results showing that associative memory capabilities naturally exist in standard machine learning architectures (Radhakrishnan et al., 2018).

In terms of the separation function, it is clear that for exact retrieval, the max function is simply the best option, as it removes any interference between different stored memories. The improvement of the separation function is the fundamental cause behind the vast gulf of theoretical capacity and practical performance between the classical HN and the MCHN. It is straightforward to show that with the max separation function, as long as queries are simply uncorrupted copies of the memory images, and that the similarity function of a memory and query has its minimum at the memory (i.e., $\text{sim}(x, x) < \text{sim}(x, y)$ for any y), then the max separation function will achieve a theoretically *infinite* capacity for any fixed size of input query (although, of course, requiring an infinite dimensional memory matrix M). However, this theoretical capacity is irrelevant in practice where, for corrupted queries, it is the propensity of the similarity function to detect the right match between query and memory that is
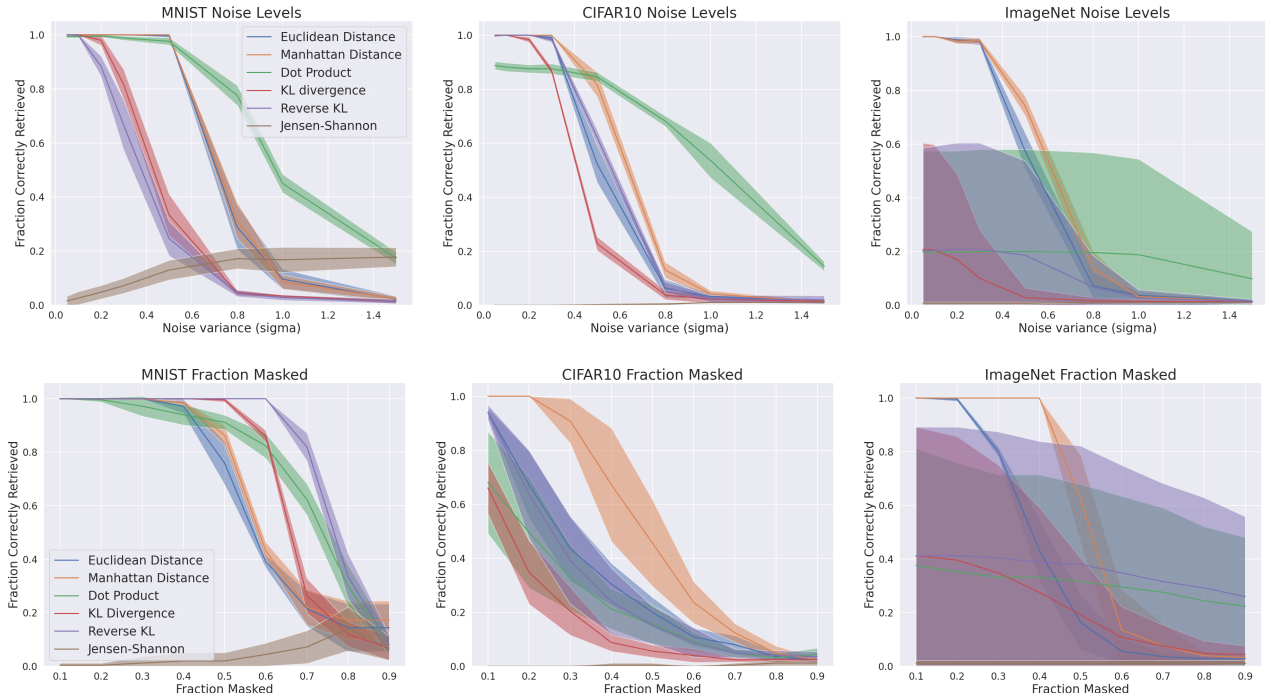
Figure 4. **Top Row**: Retrieval capability against increasing levels of $i.i.d$ added to the query images for different similarity functions. **Bottom Row**: Retrieval capability against increasing fractions of zero-masking of the query image. The networks used a memory of 100 images with the softmax separation function. Error bars are across 10 separate runs with different sets of memories stored. Datasets used left to right: MNIST, CIFAR, and Tiny ImageNet.

the main determinant of retrieval quality.

Our framework also makes a straightforward prediction that the retrieval capacity of hetero- and autoassociative memories are identical for powerful separation functions. This is because the key 'work' performed by the memory model is in the first two stages of computing the similarity scores and then separating them, while whether the result is a hetero- or autoassociative memory depends entirely on the projection matrix used to project the resulting separated similarity scores. As such, if the separation scores are nearly a one-hot vector at the correct memory index, the correct image will be 'retrieved' by the projection matrix regardless of whether it is a hetero- or autoassociated memory. We verify this prediction by studying the retrieval capacities of hetero- vs. autoassociative MCHNs and HNs in Appendix B.

Finally, while the capabilities and performance of these associative memory models may seem remote to state-of-the-art machine learning, recent work has begun to link the MHCN with self-attention in transformers (Ramsauer et al., 2020), which has also more recently been linked to SDM (Bricken & Pehlevan, 2021). These close links between associative memory models and transformer attention may therefore indicate that improvements in understanding and increasing the effective capacity of such models may also lead to improvements in transformer performance for large-scale machine learning tasks. Perhaps the most interesting avenue

lies in testing different similarity functions in transformer models, which, up to now, have almost entirely utilized the dot-product similarity function. This paper, however, has suggested that other similarity functions such as Euclidean and Manhattan distance are also highly competitive with the dot-product similarity and may lead to comparable or superior results when used in transformer self-attention. Preliminary results (Appendix F) suggest that the Manhattan and Euclidean distance similarity functions are competitive with dot product attention in small-scale transformer networks, despite transformer architectures being optimized for the dot product, and suggest that investigating transformer performance more thoroughly with different similarity functions may be an important avenue for future work.

7. Acknowledgements

References

Abbott, L. F. and Arian, Y. Storage capacity of generalized networks. *Physical Review A*, 36(10):5091, 1987.

Abu-Mostafa, Y. and Jacques, J. S. Information capacity of the Hopfield model. *IEEE Transactions on Information Theory*, 31(4):461–464, 1985.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Baldi, P. and Venkatesh, S. S. Number of stable points for spin-glasses and neural networks of higher orders. *Physical Review Letters*, 58(9):913, 1987.

Bonetti, L., Brattico, E., Carlomagno, F., Donati, G., Cabral, J., Haumann, N., Deco, G., Vuust, P., and Kringelbach, M. Rapid encoding of musical tones discovered in whole-brain connectivity. *NeuroImage*, 245:118735, 2021.

Bricken, T. and Pehlevan, C. Attention approximates sparse distributed memory. *arXiv preprint arXiv:2111.05498*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Caputo, B. and Niemann, H. Storage capacity of kernel associative memories. In *International Conference on Artificial Neural Networks*, pp. 51–56. Springer, 2002.

Chen, H., Lee, Y., Sun, G., Lee, H., Maxwell, T., and Giles, C. L. High order correlation model for associative memory. In *AIP Conference Proceedings*, volume 151, pp. 86–99. American Institute of Physics, 1986.

Demircigil, M., Heusel, J., Löwe, M., Upgang, S., and Vermet, F. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2): 288–299, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Gupta, A., Dar, G., Goodman, S., Ciprut, D., and Berant, J. Memory-efficient transformers via top-k attention. *arXiv preprint arXiv:2106.06899*, 2021.

Hinton, G. E. and Anderson, J. A. *Parallel Models of Associative Memory: Updated Edition*. Psychology Press, 2014.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10): 3088–3092, 1984.

Horn, D. and Usher, M. Capacities of multiconnected memory models. *Journal de Physique*, 49(3):389–395, 1988.

Jaeckel, L. A. *An alternative design for a sparse distributed memory*. Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.

Jayakumar, S. M., Czarnecki, W. M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., and Pascanu, R. Multiplicative interactions and where to find them. In *8th International Conference on Learning Representations*, 2020.

Jiang, Y. and Pehlevan, C. Associative memory in iterated overparameterized sigmoid autoencoders. In *International Conference on Machine Learning*, pp. 4828–4838. PMLR, 2020.

Kanerva, P. *Sparse Distributed Memory*. MIT Press, 1988.

Kanerva, P. *Sparse distributed memory and related models*, volume 92. NASA Ames Research Center, Research Institute for Advanced Computer Science, 1992.

Kaya, M. and Bilge, H. Ş. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

Keeler, J. D. Comparison between Kanerva's SDM and Hopfield-type neural networks. *Cognitive Science*, 12(3): 299–329, 1988.

Kirkpatrick, S. and Sherrington, D. Infinite-ranged models of spin-glasses. *Physical Review B*, 17(11):4384, 1978.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Krotov, D. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Krotov, D. and Hopfield, J. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.

Krotov, D. and Hopfield, J. J. Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems*, 29:1172–1180, 2016.

Kulis, B. et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

Little, W. A. The existence of persistent states in the brain. *Mathematical Biosciences*, 19(1-2):101–120, 1974.

Ma, J. The asymptotic memory capacity of the generalized Hopfield network. *Neural Networks*, 12(9):1207–1212, 1999.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Radhakrishnan, A., Yang, K., Belkin, M., and Uhler, C. Memorization in overparameterized autoencoders. *arXiv preprint arXiv:1810.10333*, 2018.

Radhakrishnan, A., Belkin, M., and Uhler, C. Overparameterized neural networks implement associative memory. *Proceedings of the National Academy of Sciences*, 117 (44):27162–27170, 2020.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Rolls, E. The mechanisms for pattern completion and pattern separation in the hippocampus. *Frontiers in Systems Neuroscience*, 7:74, 2013.

Salvatori, T., Song, Y., Hong, Y., Frieder, S., Sha, L., Xu, Z., Bogacz, R., and Lukasiewicz, T. Associative memories via predictive coding. *arXiv preprint arXiv:2109.08063*, 2021.

Sukhbaatar, S., Grave, E., Lample, G., Jegou, H., and Joulin, A. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Tang, F. and Kopp, M. A remark on a paper of Krotov and Hopfield [arxiv: 2008.06996]. *arXiv preprint arXiv:2105.15034*, 2021.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A. P., Keysers, D., Uszkoreit, J., et al. MLP-mixer: An all-MLP architecture for vision. In *35th Conference on Neural Information Processing Systems*, 2021.

Tsodyks, M. and Sejnowski, T. Associative memory and hippocampal place cells. *International Journal of Neural Systems*, 6:81–86, 1995.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wu, Y., Hu, J., Wu, W., Zhou, Y., and Du, K. Storage capacity of the Hopfield network associative memory. In *2012 5th International Conference on Intelligent Computation Technology and Automation*, pp. 330–336, 2012.

Yang, L. An overview of distance metric learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2007.

Yuille, A. L. and Rangarajan, A. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

Appendix A: Proof of Energy Function Being a Lyapunov Function of the Dynamics

In this appendix, we demonstrate that the energy function is a Lyapunov function of the dynamics. This means that by running the dynamics forward in time, the value of energy function is guaranteed to decrease. To do so, we simply compute the time derivative of the energy function and show that it must be negative:

$$
\begin{aligned}
\frac{dE}{dt} &= \sum_i \sum_j v_i \frac{\partial \mathcal{L}^2}{\partial v_i \partial v_j} \frac{dv_i}{dt} + \frac{dv_i}{dt} \frac{\partial \mathcal{L}_v}{\partial v_i} - \frac{\partial \mathcal{L}_v}{\partial v_i} \frac{dv_i}{dt} + \sum_i \sum_j h_i \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \frac{dh_i}{dt} + \\
&\quad \sum_i \frac{\partial \mathcal{L}_h}{\partial h_i} \frac{dh_i}{dt} - \sum_i \frac{dh_i}{dt} \frac{\partial \mathcal{L}_h}{\partial h_i} - \sum_i \sum_j \frac{\partial \mathcal{L}_v^2}{\partial v_i \partial v_j} \frac{\partial \mathrm{sim}(M_{i,j}, v_i)}{\partial v_i} \frac{dv_i}{dt} - \sum_i \sum_j \mathrm{sim}(M_{i,j} v_i) \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \frac{dh_i}{dt} \\
&= \sum_i \sum_j v_i \frac{\partial \mathcal{L}_v^2}{\partial v_i \partial v_j} \frac{dv_i}{dt} + h_i \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \frac{dh_i}{dt} + \frac{\partial \mathcal{L}_v^2}{\partial v_i \partial v_j} \frac{\partial \mathrm{sim}(M_{i,j} v_i)}{\partial v_i} \frac{dv_i}{dt} - \mathrm{sim}(M_{i,j}, v_i) \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \frac{dh_i}{dt} \\
&= \sum_i \sum_j \frac{dv_i}{dt} \frac{\partial \mathcal{L}_v^2}{\partial v_i \partial v_j} \Big[v_i - \frac{\partial \mathrm{sim}(M_{i,j}, v_i)}{\partial v_i} \Big] + \frac{dh_i}{dt} \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \Big[h_i - \mathrm{sim}(M_{i,j}, v_i) \Big] \\
&= - \sum_i \sum_j \frac{dv_i}{dt} \frac{\partial \mathcal{L}_v^2}{\partial v_i \partial v_j} \frac{dv_i}{dt} + \frac{dh_i}{dt} \frac{\partial \mathcal{L}_h^2}{\partial h_i \partial h_j} \frac{dh_i}{dt} ,
\end{aligned}
\tag{10}
$$

which is clearly always negative as long as the Hessians of the activation functions are positive definite. In the usual case of elementwise activation functions, this requires that the functions be monotonically increasing. Note that in this derivation, we have assumed that the input currents are constant $\frac{dI}{dt} = 0$, the fact that the derivative of the Langrangians can be defined by the chain rule as $\frac{d\mathcal{L}_\sqsubseteq}{dt} = \frac{\partial \mathcal{L}_v}{\partial v} \frac{dv}{dt}$, and the definition of the dynamics of the visible and hidden neurons.

Appendix B: Heteroassociative Hopfield Networks

In this appendix, we follow up on the discussion in Section 1.1 and demonstrate that both the MCHN and the HN can be straightforwardly extended to perform heteroassociative memory retrieval with no impact on performance compared to autoassociativity. This is done simply by replacing the memory matrix in the projection step by a different memory matrix which represents the other memories that must be heteroassociated with the main memories. If we define the memory matrix as M and the heteroassociative projection matrix as P, this results in the following update rule for the MCHN:

$$z^* = P\sigma(M \cdot q) \tag{11}$$

and the following update rule for the HN:

$$z^* = \text{sign}(\tilde{T} \cdot q), \tag{12}$$

where the heteroassociative memory matrix $\tilde{T}$ can be written as PM^T. The reason for the negligible performance difference between auto- and heteroassociation is that all the 'difficult' computation that can lead to mis-reconstructions occurs during the computation of the similarity scores and the application of the separation function. Once the set of similarity scores is computed, these scores simply select the linear combination of rows of the projection matrix that is to be reconstructed. Whether this projection matrix is the same as the memory matrix M, or some other autoassociation matrix P is immaterial.
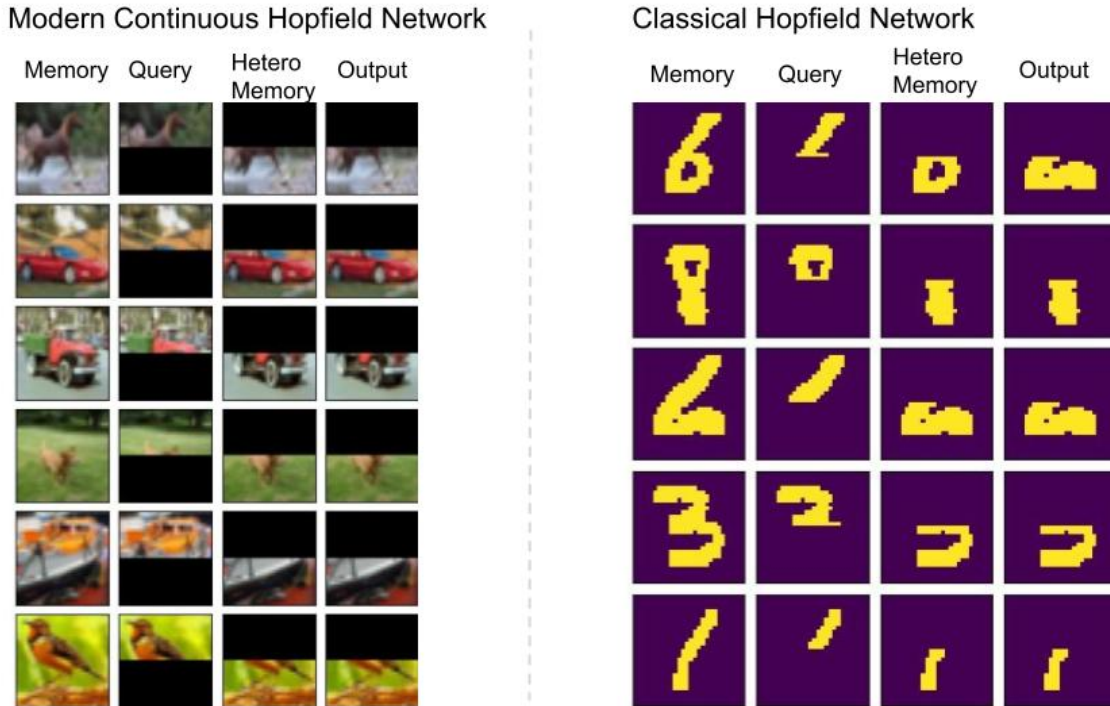

Figure 5. Examples of heteroassociative memory retrieval for both the MCHN and the HN. The networks were queried with one half of either a CIFAR10 image for the MCHN or a binarized MNIST digit for the HN. The autoassociated memory was the other half of the original memory image. On the left, the MCHN achieves perfect heteroassociation, even of challenging CIFAR10 images, due to its superior capacity to the HN. On the right, the HN achieves substantially correct heteroassociations but occasionally misreconstructs an image.

An additional direct consequence of our theoretical framework is that there should be effectively no difference in hetero- vs. autoassociative memory retrieval performance for any feedforward memory model, since all the 'work' is already done in computing the similarity and separation functions, while the difference between auto- and heteroassociative memories occurs only at the projection matrix. We verify this prediction here by comparing the retrieval capacity of auto- and heteroassociative MCHNs and HNs as the memory size increases and find them to be statistically identical.
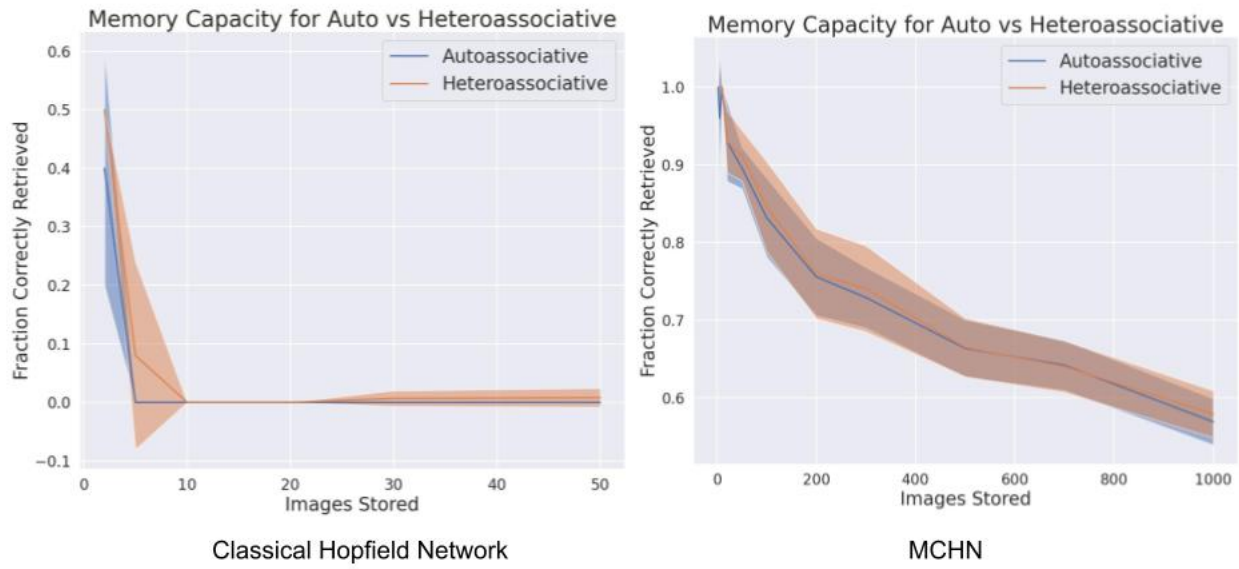
Figure 6. Comparison of auto- vs. heteroassociative MCHN and HNs on retrieval task. For both, given a corrupted image, the heteroassociative task was to retrieve only the bottom half. The MCHN was queried with CIFAR10 images corrupted with Gaussian noise of variance 0.5. The HN was tested on binarized MNIST images where the query was the top half of the image. Error bars are the standard deviations of the retrieval capacity over 10 runs. The performance of the HN is extremely poor due to interference between memories caused by its identity separation function. In both cases, the differences between auto- and heteroassociative capacity are negligible

Appendix C: Reconstructions under Varying Levels of Image Perturbation

In this appendix, we include example reconstructions under varying levels of query degradation as an example of the two kinds of query degradation examined in the main text. These are addition of random Gaussian noise with varying variance, and masking (with zeros) of a given fraction of the query image. We present sample reconstructions of an associative memory network (with Manhattan distance similarity and softmax separation functions) under both different levels of noise variances and fractions of the image masked out. The values shown here are the same as in the capacity robustness plots (Fig. 2), so that an intuitive picture of the difficulty of the network's tasks can be gauged.
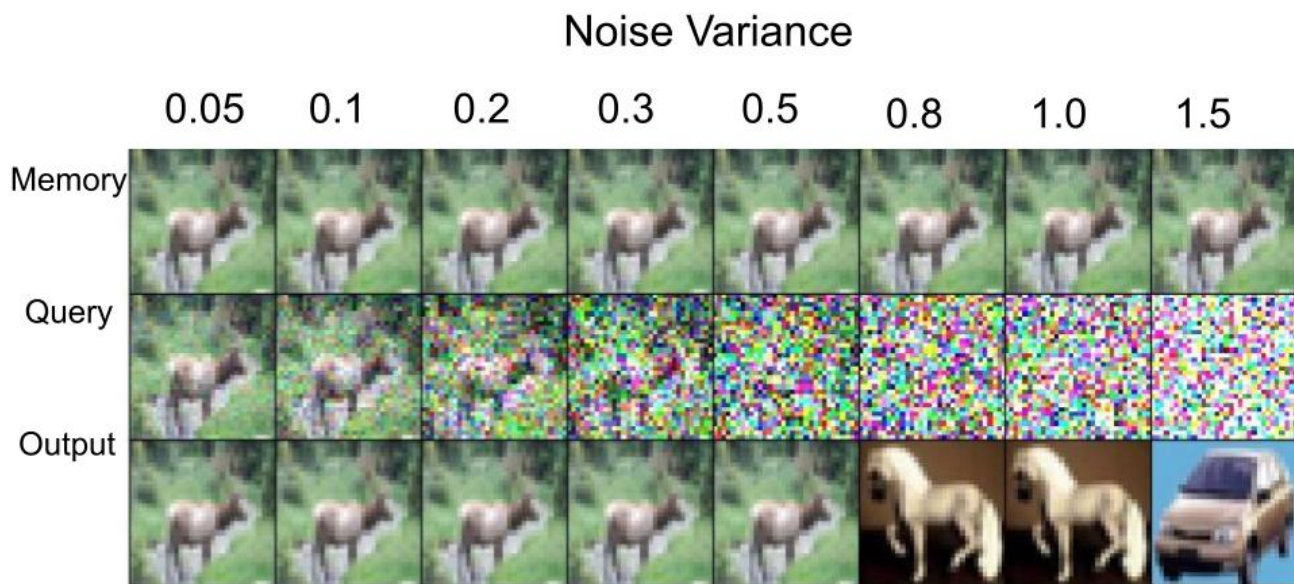
Figure 7. Examples of reconstruction of an associative memory network using the Manhattan distance similarity function and the softmax separation function. The network achieves a perfect performance up to a noise variance of about 0.5, which visually is an extremely distorted version of the original image. For high variances, the reconstructions are incorrect, however, a feature of the MCHN is that the attractors span the space such that any input pattern, even a meaningless one, will eventually be classed as a given pattern.
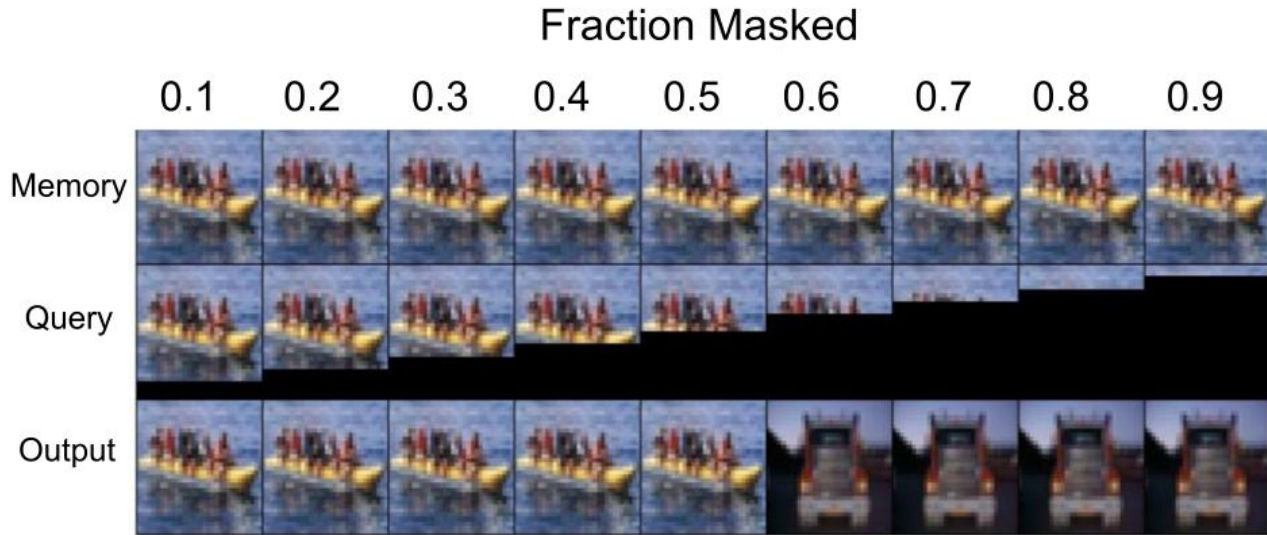
Figure 8. Examples of reconstruction of an associative memory network using the Manhattan distance similarity function and softmax separation function. The network achieves a perfect performance up to a fraction masked of 0.5, while reconstructions afterwards are incorrect. Interestingly, visually to a human this task is much easier than the Gaussian noise distortion, but the network finds denoising the Gaussian noise significantly easier. This may be due to the design of the similarity functions for which the noisy images are 'closer' in space to the memory than images with half or more of the image as zeros, which of course generates large errors for all the zero pixels.

Appendix D: Suite of Similarity and Separation Functions

The list of similarity functions tested is presented in Table 2 below.

Table 2. Similarity Functions

Similarity Function	Definition
Euclidean Distance	$\text{sim}(M, q) = \sum(M - q)^2$
Manhattan Distance	$\text{sim}(M, q) = \sum \text{abs}(M - q)$
Dot Product	$\text{sim}(M, q) = M.q$
KL Divergence	$\text{sim}(M, q) = \sum q \ln \frac{M}{q}$
Reverse KL Divergence	$\text{sim}(M, q) = \sum M \ln \frac{M}{q}$
Jensen-Shannon Divergence	$\text{sim}(M, q) = \frac{1}{2}KL[M||q] + \frac{1}{2}KL[q||M]$

Similarly, the list of separation functions tested is given in the Table 3 below.

Table 3. Separation Functions

Separation Function	Definition
Identity	$\text{sep}(x) = x$
Square	$\text{sep}(x) = x^2$
N-th Order Polynomial	$\text{sep}(x, n) = x^n$
Log	$\text{sep}(x) = \ln x$
Softmax	$\text{sep}(x, \beta) = \frac{e^{-\beta * x}}{\sum e^{-\beta * x}}$
Max	$\text{sep}(x) = \max(x)$

Appendix E: Numerical and Simulation Notes

Two key issues for making a fair comparison between similarity functions is the numerical effects of scaling and the direction of similarity. Firstly, implemented naively, the similarity metrics often have different characteristic scales, i.e., one measure might naturally return values that are much larger or smaller than another. This would then change the effect of the separation function and thus the reconstruction accuracy. For instance, a method that returned high similarity values would often be easier to separate than one which returned small ones. To address this problem, we normalized the similarity scores of each similarity function so that they would sum to 1. A second problem is the direction of similarity, namely, whether the similarity function returned high or low values for similar inputs. Similarity measures such as the dot product give large dot products for similar inputs, while distance measures such as the Euclidean distance give small distances for similar inputs. To address this, for distance metrics, we instead returned the normalized reciprocal of the distances, so that large inverse distances correspond to a high similarity. Thus, all similarity functions returned their scores in a standardized normalized format whereby larger scores represented larger degrees of similarity, and the sum of all scores was normalized to one. The outputs of the separation function are also normalized such that their sum is 1.

To compute the number of correct retrievals, given a memory matrix and a query vector, we first implemented the reconstruction through the main equation $z = M^T \text{sep}(\text{sim}(M, q))$, where sim is defined to output scores in the normalized format. The input vector q is derived by simply flattening the input image. The memory matrix M consists of a series of flattened vectors for the stored images $M = [m_1, m_2, \dots]$, where m_1 is the flattened vector of a memory image. Once the reconstruction z was computed, we compared it to the original image and computed a reconstruction score based on the sum of the squares of the elementwise differences between the reconstruction and the true image $L = \sum_i (z_i - z_i^*)^2$. If the sum of squares was less than a threshold (here, we used 50), then the image was classed as being correctly retrieved and otherwise not. The threshold value was chosen empirically to allow reconstructions that are almost indistinguishable by eye from the true input, while ruling out incorrect and poor reconstructions.

E.1. Dot-Product Similarity

A key weakness of the dot-product similarity metric is that it is not invariant to the norm of the vectors. This means that the similarity computed depends heavily on the vector norms often more so than the similarities. Mathematically, this results in the fact that it is not the case that $x^T x \geq x^T z$, where z is any other vector. What this means is that two completely different inputs can have a higher dot product similarity than the input dotted with itself. This does not happen with other similarity metrics such as the Euclidean distance where the minimum distance of 0 is achieved when the query vector and a memory vector are identical. This occurs because the z vector may have a larger norm than the x vector. This problem in practice leads to catastrophically poor performance of the dot-product similarity metric, especially on dense color images like the CIFAR and Tiny ImageNet datasets, where the un-normalized dot product simply tends to simply measure the degree of high pixel values in an image. To alleviate this issue, results in the paper are instead reported using a normalized dot-product similarity function defined as

$$\text{dot}(X, z) = \text{norm}(\text{norm}(X) \cdot \text{norm}(z)), \tag{13}$$

where $\text{norm}(x) = \frac{x}{\sum x}$ simply normalizes the entries in the vector to sum to 1, and where the norm on the memory matrix X is taken for each row (i.e., stored vector) independently. The dot product of the two normalized vectors is then normalized again for numerical reasons, since otherwise the similarity scores computed were often extremely small leading to numerical issues and poor performance with the separation function.

A similar normalization was also performed for the similarity functions that involved a KL divergence, which possesses a probabilistic interpretation, and thus all the input vectors were normalized, so as to sum to one, and thus preserve an interpretation in terms of probability distributions.

Appendix F: Transformer Experiments

To test whether the insights gained from this framework might apply to large-scale machine learning in the form of improving transformer attention, we implemented transformer layers using various similarity functions. Mathematically, we modified the transformer update rule to

$$z = V\sigma(\beta \cdot \text{sim}(K, Q)), \tag{14}$$

where V, K, and Q are the Value, Key, and Query matrices of transformer attention, β is the softmax temperature, σ is the softmax function, and sim is the similarity function. All other aspects of the transformer architecture remained the same.

We utilized an encoder-decoder transformer architecture with 2 residual blocks. Each residual block included a modified attention layer, an MLP layer, and batch normalization. The transformers were trained on the Wikitext dataset using the ADAM optimizer. The MLP layer had a hidden dimension of 200, and the embedding dimension was also 200. Two attention heads were used. A batch size of 20 was used.
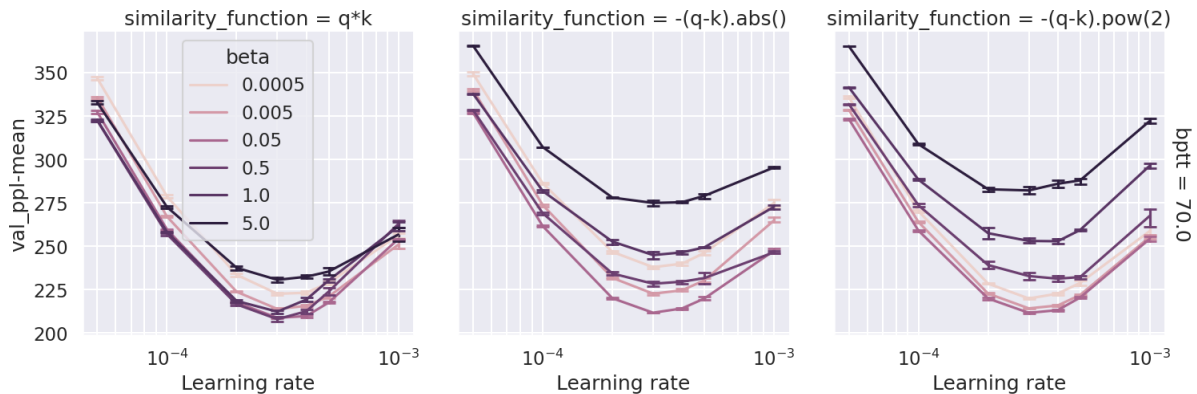

Figure 9. Achieved perplexity on the WikiText dataset using transformer models with varying similarity functions across a range of learning rates. All successful similarity functions achieved similar results although the absolute value and Euclidean distance similarity functions appeared more sensitive to choices of the β hyperparameter.

Although small-scale, from the preliminary results, it appears that the similarity function used actually makes relatively little difference for the final learning performance of the transformer. This may suggest that despite recent works interpreting attention through the lens of heteroassociative memories (Ramsauer et al., 2020; Bricken & Pehlevan, 2021), transformers are not particularly functioning as associative memories in that the learnt K and Q matrices do not directly correspond to learnt 'memories' but rather that the success of the transformer architecture is simply due to the mathematical structure of the update rule — for instance, the multiplicative interactions, which have also been shown to be crucial in non-transformer contexts, such as in the MLP-mixer architecture (Tolstikhin et al., 2021; Jayakumar et al., 2020). Supporting evidence for this comes from the fact that the post-softmax attention outputs are typically dense and the softmax temperature is often relatively high, meaning that the system is not reconstructing exact memories as the MCHN does. Sparsifying self-attention outputs (for instance, with a large inverse temperature or with an explicit *max* or *top-k* function (Gupta et al., 2021) replacing the softmax), however, will have the effect of making the self-attention function much more similar to an associative memory that retrieves specific memories given a cue.

Additionally, it could simply be that given that the K, Q, and V matrices are all learnable, that backpropagation can simply route around the different similarity functions, and specialize representations for a given similarity function. If this is the case, then it may indicate that substantial computational savings may be obtained by devising similarity functions that are less expensive than the dot product currently used. Much work in this area is already underway and many computationally cheaper simplifications of the transformer have been proposed (Kitaev et al., 2020; Wang et al., 2020; Tay et al., 2020).

Appendix G: Effect of Error Threshold

For the simulations in this paper, the error threshold was set somewhat arbitrarily as the sum of squared pixel values between the reconstruction and the original image, needing to be less than 50 for the reconstruction to be classified correctly. Although heuristic, this threshold roughly corresponds to retrieving the correct image and tolerating some fuzziness in reconstruction, but not too much. As an example of reconstructions with different squared differences, see Figure 10.

Figure 10. Examples of reconstructed images by a Manhatten distance associative memory model according to various threshold values. Note that the correct retrieval is typically achieved with significantly higher threshold values, but the relatively low threshold enforces *exact* as opposed to blurry reconstructions.

To validate that the exact value of the threshold had relatively little affect on our results, we demonstrate the retrieval accuracy on CIFAR10 of the three similarity functions over a range of potential thresholds, noting that the curve is relatively flat at the threshold of 50 chosen and that all similarity functions are affected approximately equally by the change in the threshold — unsurprisingly, with more images being correctly retrieved with a larger threshold value.
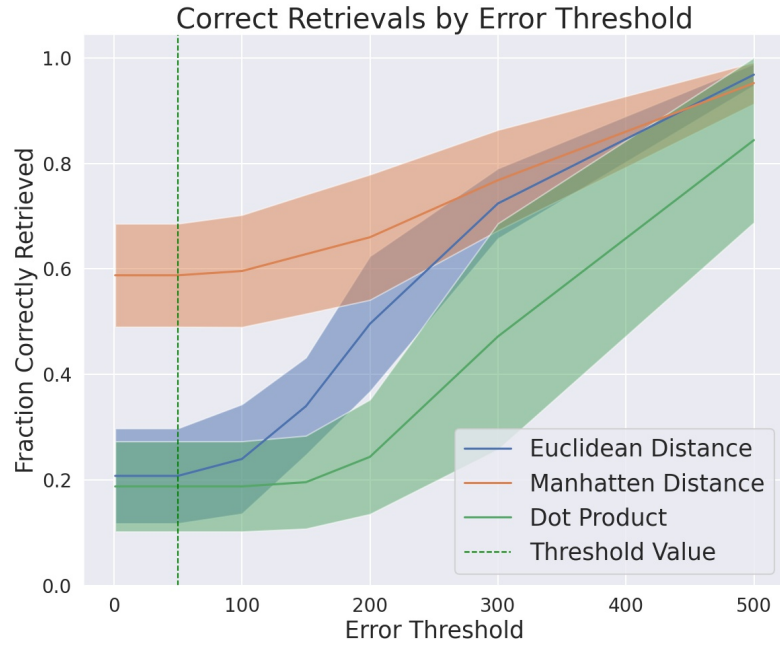
Figure 11. Percentage of correct retrieval against error threshold. Note that the threshold chosen lies at the flat part of the graph, although unsurprisingly, correct retrievals increase rapidly with large thresholds.

A. Appendix H: Additional Image Perturbations

To test whether the same effects of similarity function are maintained on a wider set of image perturbations, in this appendix, we additionally present an analysis of two further perturbations — *random masking*, where randomly chosen pixels on the image are masked out (set to zero) with a certain probability, and inpainting, where the edges of the image are masked out, and the edges must be reconstructed from the center. Examples of both perturbations for differing levels of pixel masking are presented in Figures 12 and 13, respectively.
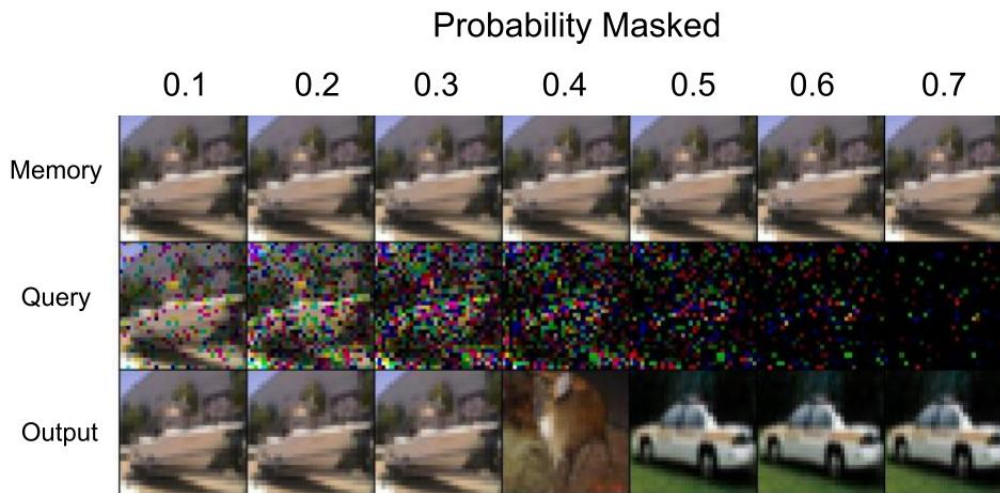

Figure 12. Example queries and reconstructions for the random masking perturbation.
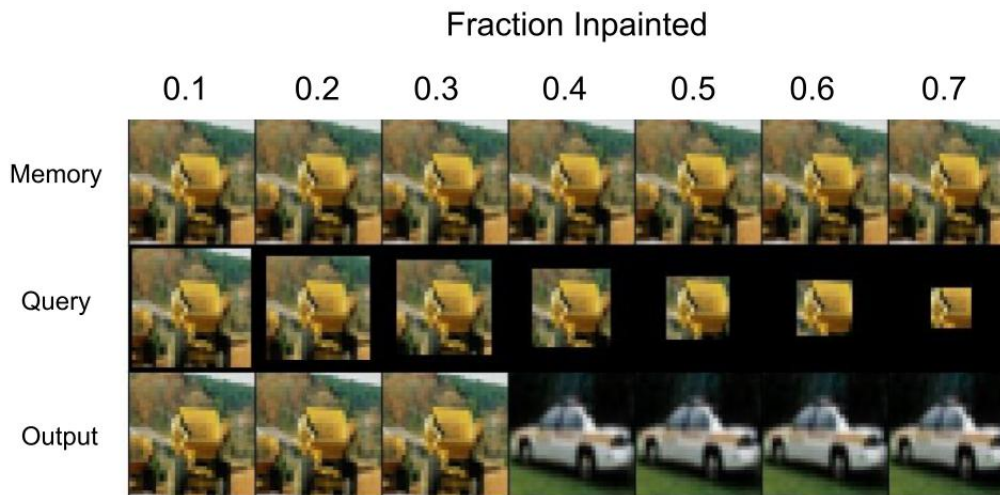

Figure 13. Example queries and reconstructions for the image inpainting perturbation.

We performed equivalent capacity tests of the three main similarity functions proposed (Manhattan and Euclidean distance and dot-product similarity) and plotted the results below,
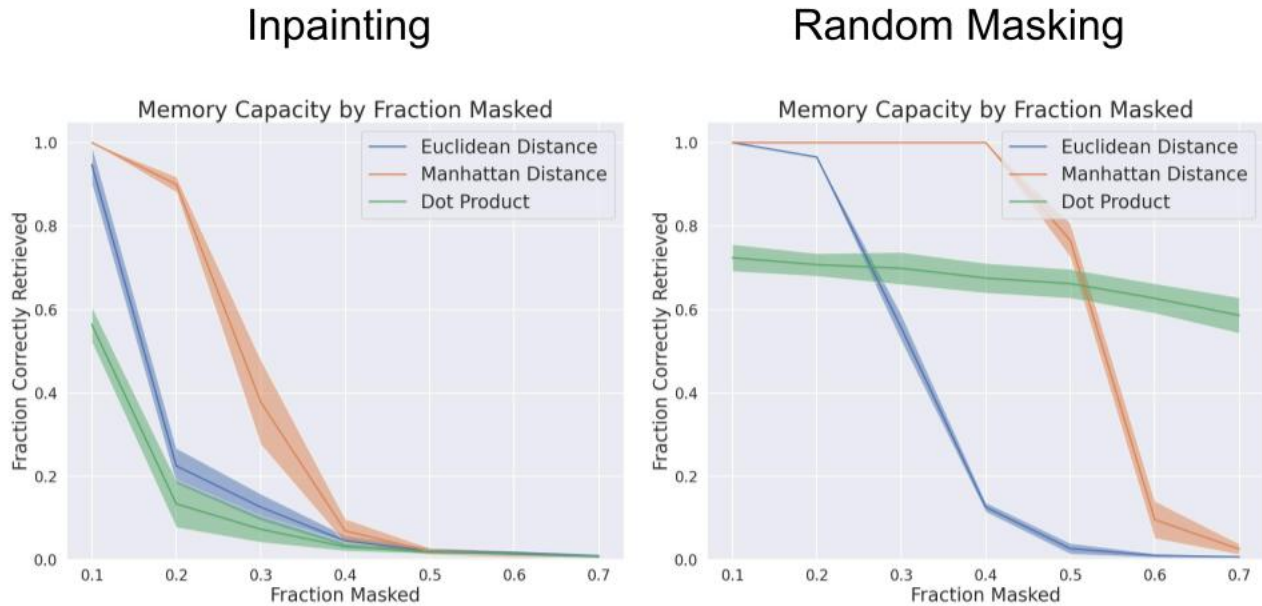
Figure 14. Capacity of networks by similarity function on the two additional perturbations, inpainting and random masking. Error bars are standard deviations across 10 runs.

We see that inpainting followed the standard pattern of superiority of the Manhattan distance metric. However, random masking showed a significantly more robust performance at high capacity for the dot product similarity. This highlights how the optimal similarity function is highly situational depending significantly on the likely set of perturbations that need to be corrected.

B. Code Availability

Code to reproduce all the experiments and figures reported in this paper is freely available at: https://github.com/BerenMillidge/Theory_Associative_Memory.