Attention Approximates Sparse Distributed Memory

Trenton Bricken
Systems, Synthetic and Quantitative Biology
Harvard University
trentonbricken@g.harvard.edu

Cengiz Pehlevan
Applied Mathematics
Harvard University
cpehlevan@seas.harvard.edu

Abstract

While Attention has come to be an important mechanism in deep learning, it emerged out of a heuristic process of trial and error, providing limited intuition for why it works so well. Here, we show that Transformer Attention can be closely related under certain data conditions to Kanerva's Sparse Distributed Memory (SDM), a biologically plausible associative memory model. We confirm that these conditions are satisfied in pre-trained GPT2 Transformer models. We discuss the implications of the Attention-SDM map and provide new computational and biological interpretations of Attention.

Introduction

Used most notably in the Transformer, Attention has helped deep learning to arguably approach human level performance in various tasks with larger models continuing to boost performance [1, 2, 3, 4, 5, 6, 7, 8, 9]. However, the heuristic motivations that produced Attention leave open the question of why it performs so well on many tasks that humans do [1, 10]. Insights into why Attention is so effective would not only make it more interpretable but also guide future improvements.

Much has been done to try and explain Attention's success, including work showing that Transformer representations map more closely to human brain recordings and inductive biases than other models [11, 12]. Our work goes further in showing the potential relationship between Attention and the brain at the level of its neuronal wiring, giving a new and exciting perspective behind Attention's success. This potential relationship to the brain is created by showing mathematically that Attention closely approximates Sparse Distributed Memory (SDM).

SDM is an associative memory model developed in 1988 to solve the "Best Match Problem", where we have a set of memories and want to quickly find the "best" match to any given query [13, 14]. In the development of its solution, SDM respected fundamental biological constraints, such as Dale's law, that synapses are fixed to be either excitatory or inhibitory and cannot dynamically switch (see Section 1 for an SDM overview and [13] or [15] for a deeper review). Despite being developed independently of neuroanatomy, SDM's biologically plausible solution maps strikingly well onto the cerebellum[1] [13, 21].

Abstractly, the relationship between SDM and Attention exists because SDM's read operation that uses intersections between high dimensional hyperspheres is approximately an exponential over a sum of exponentials that is the Attention softmax rule (Section 2). Establishing that Attention approximates SDM mathematically, we then test it in pre-trained GPT2 Transformer models (Section 3). We use the Query-Key Normalized Transformer variant to directly show that the relationship to SDM holds well [22]. We then use original GPT2 models to help confirm this result and make it

[1] This cerebellar relationship is additionally compelling by the fact that cerebellum-like neuroanatomy exists in many other organisms including numerous insects (eg. the Drosophila Mushroom Body) and potentially cephalopods [16, 17, 18, 19, 20].

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

more general. Given these results, we put forward the hypothesis that for Attention to work well it may need to exist in the same regime as that used by SDM.

Using the SDM framework, we are able to go beyond Attention and interpret the Transformer architecture as a whole, providing its heuristic structure with deeper intuition (Section 4). Motivated by this mapping between SDM and the Transformer, we introduce the hypothesis that Attention is performing similar associative memory operations to the brain by summarizing SDM's relationship to the cerebellum and other potential brain regions (Section 5). In related work (Section 6), we link SDM to other memory models [23, 24], including how SDM is a generalization of Hopfield Networks and, in turn, how our results explain and extend work relating Hopfield Networks to Attention [25]. Finally, we discuss limitations, and future research directions that could leverage our work (Section 7).

To what extent will our most successful deep learning systems converge with the brain? Some convergence seems inevitable, given that we want our models to excel at the same tasks that brains do. However, the brain and machine learning models have different optimizers, objective functions, and constraints. In this light, it is surprising that the connectome of the cerebellum, if it implements Sparse Distributed Memory, closely approximates Attention and many aspects of the whole Transformer architecture. We believe this work provides a small piece of evidence that the brain and the best deep learning models may be more convergent than we might otherwise think.

1 Review of Kanerva's SDM

Here, we present a short overview of SDM with a deeper review on the motivations behind SDM and the features that make it biologically plausible in [13, 15]. SDM provides an algorithm for how memories are stored in, and retrieved from, neurons in the brain. There are three primitives that all exist in an n dimensional binary vector space:

Patterns ($\mathbf{p}$) - have two components: the pattern address, $\mathbf{p}_a^\mu \in \{0,1\}^n$, is the vector representation of a memory; the pattern "pointer", $\mathbf{p}_p^\mu \in \{0,1\}^n$, is bound to the address and points to itself when autoassociative or to a different pattern address when heteroassociative. A heteroassociative example is memorizing the alphabet where the pattern address for the letter a points to pattern address b, b points to c etc. For tractability in analyzing SDM, we assume our pattern addresses and pointers are random. There are m patterns and they are indexed by the superscript $\mu \in \{1, \ldots, m\}$.

Neurons ($\mathbf{x}$) - it is sufficient in showing SDM's relationship to Attention to know there are r neurons with fixed addresses $\mathbf{x}_a^\tau \in \{0,1\}^n$ that store a set of all patterns written to them. Each neuron will sum over its set of patterns to create a superposition. This creates minimal noise interference between patterns because of the high dimensional nature of the vector space and enables all patterns to be stored in an n dimensional storage vector. Their biologically plausible features are outlined in [13, 15]. When we assume our patterns are random, we also assume our neurons are randomly distributed so that they can best cover the space of patterns. Of the 2^n possible vectors in our binary vector space, SDM is "sparse" because it assumes that $r \ll 2^n$ neurons exist in the space.

Query ($\boldsymbol{\xi}$) - is the input to SDM, denoted $\boldsymbol{\xi} \in \{0,1\}^n$. The goal in the Best Match Problem is to return the pattern pointer stored at the closest pattern address to the query. We will often care about the maximum noise corruption that can be applied to our query, while still having it read out the correct pattern. An autoassociative example is wanting to recognize familiar faces in poor lighting. Images of faces we have seen before are patterns stored in memory and our query is a noisy representation of one of the faces. We want SDM to return the noise-free version of the queried face, assuming it is stored in memory.

SDM uses the Hamming distance metric between any two vectors defined: $d(\mathbf{a}, \mathbf{b}) := \mathbb{1}_n^T |\mathbf{a} - \mathbf{b}|$. The all ones vector $\mathbb{1}_n$ is of n dimensions and $|\mathbf{a} - \mathbf{b}|$ takes the absolute value of the element-wise difference between the binary vectors. When it is clear what two vectors the Hamming distance is between, we will sometimes use the shorthand $d_v := d(\mathbf{a}, \mathbf{b})$.

The Hamming distance is crucial for determining how many neurons read and write operations are distributed across. The optimal Hamming distance for the read and write circles denoted d^* depends upon the distribution of patterns in the vector space and what the memories are being used for (e.g. maximizing the number of memories that can be stored versus the memory system's robustness to query noise). We provide three useful reference d^* values, using equations outlined in Appendix B.5.

The Signal-to-Noise Ratio (SNR) optimal d^*_{SNR} maximizes the probability a noise-free query will return its target pattern [15]. The memory capacity optimal d^*_{Mem} maximizes the number of memories that can be stored with a certain retrieval probability and also assumes a noise-free query. The critical distance d^*_{CD} maximizes, for a given number of patterns, the amount of noise that can be applied to a query such that it will converge to its correct pattern [15].

These d^*s are only reference points for later comparisons to Transformer Attention, first and foremost because they assume random patterns to make their derivations tractable. In addition, Transformer Attention will not be optimizing for any one of these objectives, and likely interpolate between these optimal d^*s as it wants to both have a good critical distance to handle noisy queries and have a reasonable memory capacity. These optimal d^* are a function of n, r and m. For the Transformer Attention setting where $n = 64$, $r = 2^n$ and $m \leq 1024$, $d^*_{\text{SNR}} = 11$, $d^*_{\text{Mem}} = 5$, $d^*_{\text{CD}} = 15$ as shown in Appendix B.5 [1].

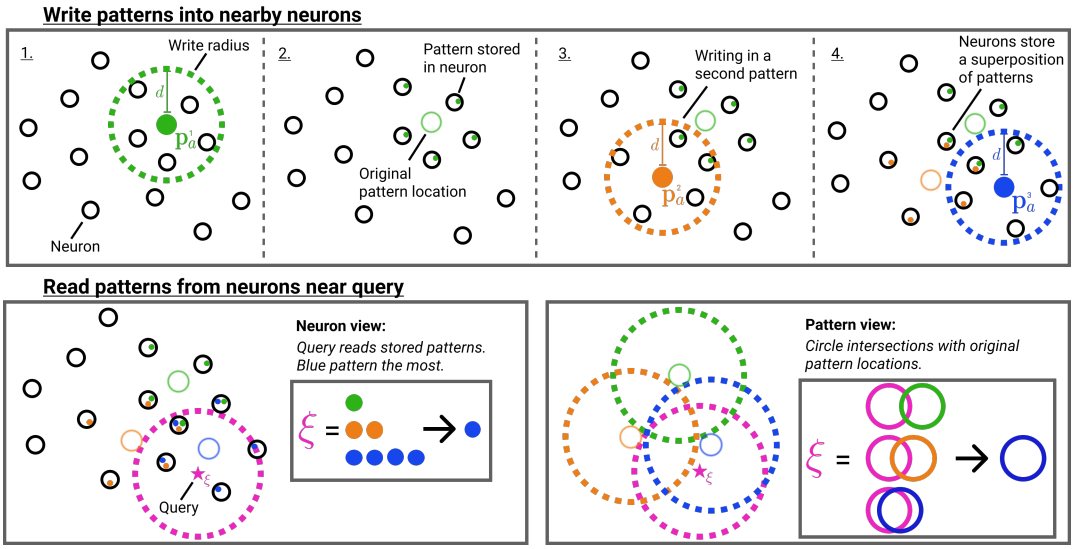

Figure 1: **Top Row** three patterns being written into nearby neurons. *1.* The first write operation *2.* Patterns stored inside neurons and the ephemeral pattern location *3.* Writing a second pattern *4.* Writing a third pattern and neurons storing a superposition of multiple patterns. **Bottom Row** shows two isomorphic perspectives of the read operation. Neuron view (left) shows the query reading from nearby neurons with the inset showing that four blue patterns a majority (Eq. 1) this would result in a one step convergence. Pattern view (right) is crucial to relating SDM to Attention. We abstract away the neurons by assuming they are uniformly distributed and just consider the circle intersection between the query and the original locations of each pattern where blue has the largest circle intersection.

1.1 SDM Read Operation

For the connection to Attention we focus on the read operation and briefly summarize the SDM write operation: all patterns write their pointers $\mathbf{p}_p$ in a distributed fashion to all neuron addresses located within Hamming distance d. This means that each neuron will store a superposition of pattern pointers from those pattern addresses within d: $\mathbf{x}_v^\tau = \sum_{\{\mathbf{p}:d(\mathbf{p}_a^\mu, \mathbf{x}_a^\tau)<d, \forall\mu\}} \mathbf{p}_p$. Having stored patterns in a distributed fashion across nearby neurons, SDM's read operation retrieves stored pattern pointers from all neurons within distance d of the query and averages them. This average is weighted because the same patterns have distributed storage across multiple neurons being read from. The pattern weighting will be higher for those patterns with addresses nearer the query because they have written their pointers to more of the neurons the query reads from. Geometrically, this weighting of each pattern can be interpreted as the intersection of radius d circles[2] that are centered on the query and

[2]In this binary space, the Hamming distance around a vector is in fact a hypercube but the vertices of an n dimensional unit cube lie on the surface of an n dimensional sphere with radius $\sqrt{n}/2$ and we refer to this as a

each pattern address $\mathbf{p}_a^\mu$ for all μ. A high level overview of the SDM read and write operations is shown in Fig. 1.

The number of 2^n possible neurons that have both stored this pattern's pointer $\mathbf{p}_p$ *and* been read by $\boldsymbol{\xi}$ is: $|O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)|$, where $|\cdot|$ is the cardinality operator and $O_n(\boldsymbol{\xi}, d) = \{\mathbf{x}_a \in \{0,1\}^n : d(\boldsymbol{\xi}, \mathbf{x}_a) < d\}$ is the set of all possible neuronal addresses $\mathbf{x}_a$ within radius d of $\boldsymbol{\xi}$. Mathematically, SDM's read operation sums over each pattern's pointer, weighted by its query circle intersection:

$$\boldsymbol{\xi}^{new} = g\left(\frac{\sum_{\mathbf{p}\in P} |O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)| \mathbf{p}_p}{\sum_{\mathbf{p}\in P} |O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)|}\right), \qquad g(e) = \begin{cases} 1, & \text{if } e > \frac{1}{2} \\ 0, & \text{else} \end{cases}, \tag{1}$$

and g acts elementwise on vectors. The denominator normalizes all of the weights so they sum to 1 in the numerator and enable computing if the element-wise majority value is a 0 or 1, using the function $g(\cdot)$. Intuitively, the query will converge to the nearest "best" pattern because it will have the largest circle intersection weighting. The output of the SDM read operation is written as updating the query $\boldsymbol{\xi} \to \boldsymbol{\xi}^{new}$ so that it can (but is not required to) apply the read operation iteratively if full convergence to its "best match" pattern is desired and was not achieved with one update.

The circle intersection (derived in Appendix B.1) is calculated as a function of the Hamming distance threshold for the read and write operations d, the dimensionality n, and the vector distance between the query and pattern $d_v = d(\mathbf{p}_a, \boldsymbol{\xi})$:

$$|O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)| = \sum_{a=n-d-\lfloor \frac{d_v}{2} \rfloor}^{n-d_v} \sum_{c=\max(0,n-d-a)}^{d_v-(n-d-a)} \left(\binom{n-d_v}{a} \cdot \binom{d_v}{c} \right). \tag{2}$$

Eq. 2 is approximately exponential for the closest, most important patterns where $d(\mathbf{p}_a, \boldsymbol{\xi}) \leq 2d$, which is crucial to how SDM approximates Attention. This is shown for a representative instance of SDM in Fig. 2. The details of this approximation are provided in Appendix B.2, but at a high level the binomial coefficients can be represented as binomial distributions and then approximated by normal distributions that contain exponentials. With the correctly chosen constants, c_1 and c_2, that are independent of the vector distance $d(\mathbf{p}_a, \boldsymbol{\xi})$, we can make the following approximation:

$$|O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)| \approx c_1 \exp\big(-c_2 \cdot d(\mathbf{p}_a, \boldsymbol{\xi})\big) \tag{3}$$

2 Attention Approximates SDM

To be able to handle a large number of patterns, we let the pattern address matrix with each pattern as a column be: $P_a = [\mathbf{p}_a^1, \mathbf{p}_a^2, ..., \mathbf{p}_a^m]$ with pointers $P_p = [\mathbf{p}_p^1, \mathbf{p}_p^2, ..., \mathbf{p}_p^m]$.

The Attention update rule [1] using its original notation is:

$$\boldsymbol{\xi}^{new} = k\big(V \text{softmax}(\beta K^T Q)\big) = k\Big((W_v Y)\text{softmax}\big(\beta (W_k Y)^T (W_q \mathbf{q})\big)\Big).$$

We can re-write this using our notation as:

$$\boldsymbol{\xi}^{new} = k\big(P_p \text{softmax}(\beta P_a^T \boldsymbol{\xi})\big), \tag{4}$$

where $k(\cdot)$ represents post processing to the Attention update that applies a linear projection then LayerNorm [26] and will be related to the majority bit operation $g(\cdot)$ used in SDM. K, V, and Q symbolize the "key", "value", and "query" matrices, respectively. $\mathbf{q}$ is a single query vector and Y represents the patterns that are stored in memory. We write $K = W_k Y = P_a$ as the patterns Y are projected into the SDM vector space to become the addresses P_a by the learnt weight matrix W_k. Similarly, $V = W_v Y = P_p$ and $Q = W_q \mathbf{q} = \boldsymbol{\xi}$. The $\text{softmax}(\beta \mathbf{x}) = \exp(\beta \mathbf{x}) / \sum_{i=1}^n \exp(\beta x_i)$, where the exponential is element-wise and Attention sets $\beta = 1/\sqrt{n}$. Softmax normalizes a vector of values to sum to 1 and gives the largest values the most weight, to what extent depending on β.

To illustrate the relation between SDM Eq. 1 and Attention Eq. 4, we must relate the discrete setting of SDM to continuous vectors, and choose the β coefficient such that it closely approximates the almost

circle because of our two dimensional diagrams. We adopt this useful analogy, taken from Kanerva's book on SDM [13], throughout the paper.

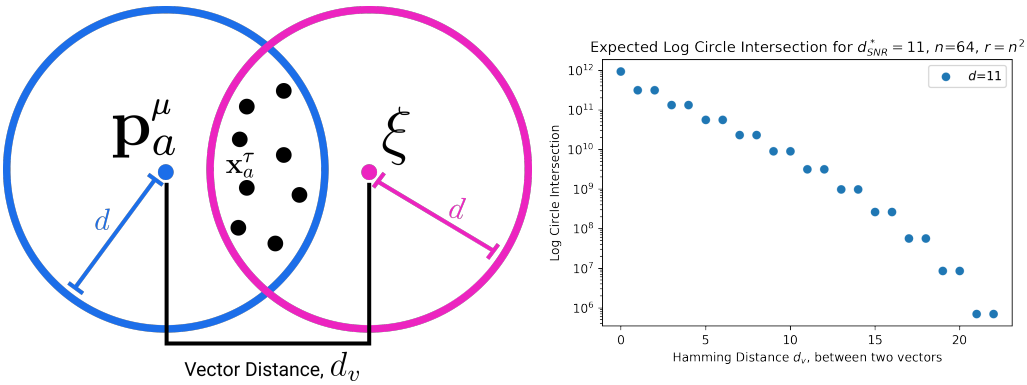

Figure 2: **(Left)** SDM's read operation using the Hamming distance d for reading and writing and the vector distance $d_v = d(\boldsymbol{\xi}, \mathbf{p}_a^\mu)$. Recall that during the write operation, pattern addresses $\mathbf{p}_a^\mu$ write their pattern pointer $\mathbf{p}_p^\mu$ to neurons located at the addresses $\mathbf{x}_a^\tau$ (denoted here as black dots) within radius d. During the read operation, the query $\boldsymbol{\xi}$ reads from each neuron within radius d, thus creating an intersection. **(Right)** As d_v between the query and pattern increases (x-axis), the size of their circle intersection declines approximately exponentially (y-axis). We use Eq. 2 with $n = 64$ and $d^*_{\text{SNR}} = 11$, while varying d_v up to the distance of $d_v = 2d$ at which point there is no circle intersection. We plot the y-axis on a log scale to show how, because the curve is approximately linear, the circle intersection is approximately exponential. See Appendix B.2 for a formal analysis of the exponential approximation the circle intersection creates that is robust across parameters n, d, and r.

exponential decay of SDM's circle intersection calculation. SDM can operate with continuous vectors by letting its addresses and pointers exist in the space $\mathbb{R}^n$ but be L^2 normalized. This continuous version of SDM has its own circle intersection equation described in Appendix B.3 that closely matches that of binary SDM in being approximately exponential. As an alternative way we will use cosine similarity, which takes values in the interval $[-1, 1]$ and denoted by $\mathcal{C}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$, as a metric of similarity in the continuous space. To relate distances and patterns in binary and continuous spaces we define a map from binary patterns to L^2 normalized continuous vectors $h(\mathbf{a}) = \hat{\mathbf{a}}$ such that for any pair of pattern addresses the following holds:

$$d(\mathbf{a}, \mathbf{b}) = \frac{n\big(1 - \mathcal{C}(\hat{\mathbf{a}}, \hat{\mathbf{b}})\big)}{2}, \tag{5}$$

where $\mathbf{a}$, $\mathbf{b}$ are binary vectors and $\hat{\mathbf{a}}$, $\hat{\mathbf{b}}$ are L^2 normalized continuous vectors. We assume that such a map exists, at least approximately.

We can plug this relationship into the exponential approximation of Eq. 3 to relate binary SDM to a distance preserving continuous version that allows us to use the dot product between the pattern and query (assuming L^2 normalization) instead of Hamming distance:

$$\begin{aligned} |O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)| &\approx c_1 \exp\big(-c_2 \cdot d(\mathbf{p}_a, \boldsymbol{\xi})\big) = c_1 \exp\Big(\frac{-c_2 n}{2}\big(1 - \mathcal{C}(\hat{\mathbf{p}}_a, \hat{\boldsymbol{\xi}})\big)\Big) \\ &= c_3 \exp\big(\beta \hat{\mathbf{p}}_a^T \hat{\boldsymbol{\xi}}\big), \end{aligned} \tag{6}$$

where c_3 encompasses the constants outside of the exponential and we replaced the remaining constants in the exponential with β that is a function of n and d.[3] Finally, this result allows us to show the relationship between SDM and Attention:

$$\begin{aligned} \boldsymbol{\xi}^{new} &= g\left(\frac{\sum_{\mathbf{p} \in P} |O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)| \mathbf{p}_p}{\sum_{\mathbf{p} \in P} |O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)|}\right) \\ &\approx k\left(\frac{\sum_{\mathbf{p} \in P} \exp(\beta \hat{\mathbf{p}}_a^T \hat{\boldsymbol{\xi}}) \mathbf{p}_p}{\sum_{\mathbf{p} \in P} \exp(\beta \hat{\mathbf{p}}_a^T \hat{\boldsymbol{\xi}})}\right) = k\Big(P_p \text{softmax}(\beta \hat{P}_a^T \hat{\boldsymbol{\xi}})\Big). \end{aligned} \tag{7}$$

[3] Instead of converting cosine similarity to Hamming distance and using the circle intersection Eq. 2 in binary vector space, we can also use a continuous version of the circle intersection outlined in Appendix B.3 that closely agrees with the binary version.

Recall that $g(\cdot)$ is the elementwise majority function that ensures our output is a valid binary vector. In the continuous setting, this is analogous to L^2 normalizing the output of the Attention update. This L^2 normalization is later related to the LayerNorm [26] operation represented by $k(\cdot)$ used in the Transformer.

One way to interpret this mapping from discrete SDM to continuous Attention is as a continuous relaxation because a given instantiation of SDM, using a particular n and d, corresponds to Attention with a specific β coefficient. To fit β we convert the Hamming distances into cosine similarity using Eq. 5 and use a univariate log linear regression that has a closed form solution:

$$\log\Big(|O_n(\mathbf{p}_a, d) \cap O_n(\boldsymbol{\xi}, d)|\Big) \approx \log(c_3) + \beta(\hat{\mathbf{p}}_a^T \hat{\boldsymbol{\xi}}). \tag{8}$$

We expect the exponential behavior to break at some point, if nothing for the reason that if $d(\mathbf{p}_a, \boldsymbol{\xi}) \geq 2d$ the circle intersection becomes zero. However, closer patterns are those that receive the largest weights and "attention" such that they dominate in the update rule and are the most important.

In Fig. 3, we plot the softmax approximation for our smallest optimal $d^*_{\text{Mem}} = 5$ and largest $d^*_{\text{CD}} = 15$ to show not only the quality of the approximations but also how many orders of magnitude smaller the normalized weights are when $d(\mathbf{p}_a, \boldsymbol{\xi}) > d$. For these plots, we generate a query vector and position 65 vectors at each distance between 0 (the points are the same) and 64, when $n = 64$. We use these vectors to compute the true circle intersection before fitting our β to create the exponential approximation. To focus our exponential approximation on the most important, closest patterns, we fit our regression to those patterns $d(\mathbf{p}_a, \boldsymbol{\xi}) < d$ and allow it to extrapolate to the remaining values. We then normalize the values and plot it along with an smaller inset plot in log space to better show the exponential relationship. In both plots, looking at the log inset plot first, the point at which the circle intersection in blue ceases to exist or be exponential corresponds to a point in the main normalized plot where the weights are ≈ 0.

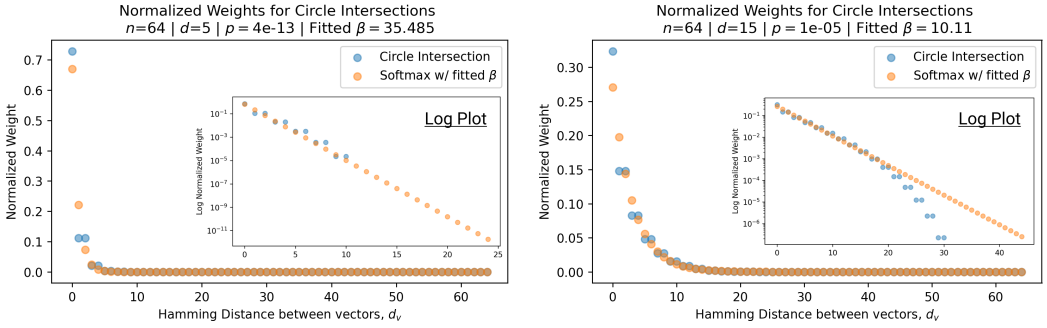

Figure 3: Representative plots showing the relationship in Eq. 7 between the softmax approximation and the normalized circle intersection using the Transformer Attention parameters. We use Eq. 5 to relate Hamming distance and cosine similarity. The insets are the same plots with a log y-axis to show how the circle intersections are approximately exponential when $d(\mathbf{p}_a, \boldsymbol{\xi}) \leq 2d$. The blue dots are the true circle intersection, the orange dots are the softmax approximation with regression fitted β. **(Left)** Uses $d^*_{\text{Mem}} = 5$, which corresponds to the Hamming distance reading and writing to $p = 4e - 13$ of the vector space. **(Right)** The same plots but with $d^*_{\text{CD}} = 15$. These results hold well for other values of n and d but we focus on the Attention parameter setting.

The number of neurons r and how well they cover the pattern manifold are important considerations that will determine SDM's performance and degree of approximation to Attention. Increasing the number of neurons in the circle intersection can be accomplished by simply increasing the number of neurons in existence,ensuring they cover the pattern manifold, and reducing the dimensionality of the manifold, increasing neuron density.[4] In SDM's original formulation it was assumed that neuronal addresses were randomly distributed and fixed in location, however, extensions to SDM [27]

[4]This can be done by learning weight projection matrices like in Attention to make the manifold lower dimensional and also increase separability between patterns.

have proposed biologically plausible competitive learning algorithms to learn the pattern manifold [28]. To ensure the approximations to SDM are tight, we test random and correlated patterns in an autoassociative retrieval task across different numbers of neurons and SDM variants (see Appendix B.7). These variants include SDM implemented using simulated neurons and the Attention approximation with a fitted β.[5]

To summarize, Attention closely approximates the continuous version of our SDM update rule when using a correctly chosen β and acknowledging the use of weight matrices W_k, W_q, and W_v to learn useful projections of the vectors.

3 Trained Attention Converges with SDM

For many instantiations of SDM, there exists a β that can be found via the log linear regression Eq. 8 that makes Attention approximate it well. However, depending on the task at hand, there are instantiations of SDM that are better than others as highlighted by the different d^* optimal values. If Attention in the Transformer model is implementing SDM, we should expect for trained Attention to use βs that correspond to reasonable instances of SDM. We use as reference points these optimal d^*s. Attention learns useful pattern representations that are far from random so this SDM β that fits the optimal d^*s are only a weak reference for what β values might be reasonable. However, because these d^* definitions of optimality span from maximizing convergence under query noise to maximizing memory capacity under noise free queries, we should expect the Transformer dealing with noisy queries and wanting reliable retrieval to interpolate between these d^* values. Here, we provide empirical evidence that this is indeed the case. We analyze the β coefficients learnt by the "Query-Key Normalization" Transformer Attention variant [22]. Query-Key Normalization, makes it straightforward to find β because it is learnt via backpropagation and easy to interpret because it uses cosine similarity between the query and key vectors. To further evaluate the convergence between Attention and SDM β coefficients and make it more general, we also investigate the GPT2 architecture [3]. However, in this case we need to infer "effective" β values from the size of query key dot products in the softmax. This makes these results, outlined in Appendix A.2, more approximate but they remain largely in agreement with the learnt βs of Query-Key Norm.

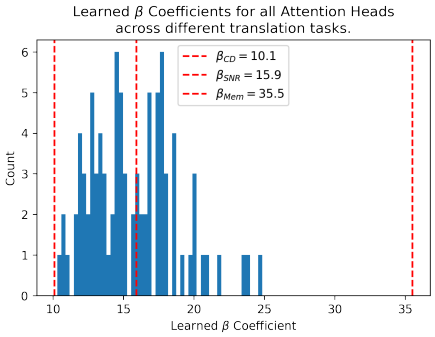

Figure 4: Histogram showing the learned β coefficients for all Attention heads across layers for the 5 translation tasks used in [22]. We plot the β values for Attention that approximate the different d^* definitions showing how the βs are interpolating between them. β_{CD} is optimal for critical distance (maximum noise for each query); β_{SNR} is optimal for Signal-to-Noise ratio. This assumes there is no query noise and SDM wants to minimize noise from other queries. β_{Mem} maximizes memory capacity, also assumes no query noise.

The Query-Key Norm Attention heads learn $\beta \in [10, 25]$ as shown in Fig. 4.[6] Note that the whole range of $\beta \in [10, 25]$ interpolate between the d^* values well, in particular with the critical distance that realistically assumes noisy queries and the SNR optimal where having a high signal to noise ratio is desirable for both memory capacity and critical distance (see Appendix B.5).

In requiring that Attention vectors be L^2 normalized and β fitted, SDM predicted Query-Key Norm. This is interesting because Query-Key Norm has initial evidence of improving Transformer performance and training speed [22, 29]. We say more about its advantages and caveats in Appendix A.1.

[5]The code for running these experiments, other analyses, and reproducing all figures is available at https://github.com/trentbrick/attention-approximates-sdm.

[6]These values were graciously provided by the authors of [22] in private correspondence for their trained Transformer models.

4 Transformer Components Interpreted with SDM

We can leverage how Attention approximates SDM to interpret many components of the Transformer architecture.[7] This exercise demonstrates the explanatory power of SDM and, in relating it to additional unique Transformer components, widens the bridge upon which ideas related to SDM and neuroscience can cross into deep learning and vice versa.

A crucial component of the Transformer is its Feed Forward (FF) layer that is interleaved with the Attention layers and uses approximately 2/3rds of the Transformer's parameter budget [1]. Attention's Transformer implementation and SDM deviate importantly in Attention's use of ephemeral patterns from the current receptive field. In order to model temporal dependencies beyond the receptive field, we want Attention to be able to store persistent memories. Work has compellingly shown that the FF layers store these persistent memories [30, 31, 32]. SDM can be interpreted as this FF layer because in [30] the FF layer was substituted with additional, persistent key and value vectors that Attention learnt independently rather than projecting from its current inputs. This substitution performed on par with the FF layer which, combined with deeper analysis in [31], shows the FF layers are performing Attention with long term memories and thus can be directly interpreted as SDM.

Another crucial component of the Transformer is its use of LayerNorm [26, 33, 1]. LayerNorm has a natural interpretation by SDM as implementing a similar constraint to L^2 normalization. However, while it ensures all of the vectors are on the same scale such that their dot products are comparable to each other, it does not constrain these dot products to the interval $[-1, 1]$ like the cosine similarity. In addition to providing an interpretation for LayerNorm, this discrepancy has led to two insights: First, as previously mentioned, it predicted that Query-Key Normalization could be a useful inductive bias (more details in Appendix A.1). Second, it has provided caution to interpretations of Attention weights. Fig. 8 of Appendix A.3 shows that there are many value vectors that receive very small amounts of attention but have large L^2 norms that dominate the weighted summation of value vectors. This makes it misleading to directly interpret Attention weights without L^2 normalization of the value vectors and this has not been done in work including [1, 34, 35, 36]. Beyond helping future interpretations of Attention, we tested L^2 normalized value vectors as a potentially useful inductive bias by training a GPT2 model with it. Our results showed that this did not change performance but L^2 normalization should still be performed in cases where the Attention weights will be interpreted. See Appendix A.3 for a full discussion.

Finally, multi-headed Attention is a Transformer component where multiple instantiations of Attention operate at the same hierarchical level and have their outputs combined. Multi-heading allows SDM to model probabilistic outputs, providing an interpretation for why it benefits Attention. For example, if we are learning a sequence that can go "$A \rightarrow C$" and "$A \rightarrow Z$" with equal probability, we can have one SDM module learn each transition. By combining their predictions we correctly assign equal probability to each. This probabilistic interpretation could explain evidence showing that Attention heads pay attention to different inputs and why some are redundant post training [37, 38, 10].

An important difference between SDM and the Transformer that remains to be reconciled is in the Transformer's hierarchical stacking of Attention. This is because, unlike in the traditional SDM setting where the pattern addresses (keys) and pattern pointers (values) are known in advance and written into memory, this cannot be done for layers of SDM beyond the first that will need to learn latent representations for its pattern addresses and pointers (keys and values). Transformer Attention learns its higher level keys and values by treating each input token as its own query to generate a new latent representation of the token that is then projected into keys and values [1]. This does not mean SDM would fail to benefit from hierarchy. As a concrete example, the operations of SDM are related to the hierarchical operations of [39]. More broadly, we believe thinking about how to learn the latent keys and values for the higher levels of SDM could present new Transformer improvements. A fundamental breakthrough with the recent Performer architecture is that it uses a reduced set of latent keys and values [40]. This highlights the arbitrariness of the original Transformer solution using queries, keys and values for every token at every layer and presents opportunities for new ways to learn the keys and values of higher layers.

[7] For a review of the components that make up the full Transformer architecture see Fig. 9 of Appendix A.

5 A Biologically Plausible Implementation of Attention

Here, we provide an overview of SDM's biological plausibility to introduce as a hypothesis that Attention is performing the same associative memory operations as the cerebellum. SDM's read and write operations have non trivial connectivity requirements described in [13, 15]. Every neuron must: (i) know to fire if it is within d Hamming distance of an input; (ii) uniquely update each element of its storage vector when writing in a new pattern; (iii) output its storage vector during reading using shared output lines so that all neuron outputs can be summed over.

Unique architectural features of the cerebellar cortex can implement all of these requirements, specifically via the three way convergence between granule cells, climbing fibers and Purkinje cells. (i) all granule cells receive inputs from the same mossy fibers to check if they are within d of the incoming query or pattern; (ii) each granule cell has a very long parallel fiber that forms synapses with thousands of Purkinje cells [41]. These synaptic connections store memories and are updated by the joint firing of granule cells and climbing fibers that wrap around individual Purkinje cells that induce LTP/LTD (Long Term Potentiation/Depression); (iii) all granule cells output their stored memories via their synapses to the Purkinje cells that perform the summation operation and use their firing threshold to determine if the majority bit was a 1 or 0, outputting the new query [13, 15].

This mapping to specific cell types and their connectivity goes beyond the low bar of simply showing how SDM can be written as a single hidden layer feedforward network and assuming that this is biologically plausible. Granule cells can compute Hamming distances and store patterns, climbing fibers can write in patterns via LTP/LTD, and Purkinje cells can perform the summation and binarizing operations. Moreover, the Drosophila mushroom body is highly similar to the cerebellum and the previous cell labels for each function can be replaced with Kenyon cells, dopaminergic neurons, and mushroom body output neurons, respectively [16].

While SDM fits the unique features of the cerebellum well, this connection and its evidence relies on neuroscientific evidence available in 1988. Explanations for some of the original model's limitations have been put forward to account for sparse dendritic connections of Granule cells and the functions of the three inhibitory interneurons: Golgi, Stellate and Basket cells [42, 27, 43]. However, there are futher challenges that remain including better explanations of the inputs to the mossy and climbing fibers and outputs from the Purkinje cells. In particular, how the mossy and climbing fiber inputs synchronize for the correct spike time dependent plasticity [44]. Another phenomena that SDM does not account for is the ability for Purkinje cells to store the time intervals associated with memories [45]. Future work will be necessary to update the state of SDM's biological plausibility with modern neuroscientific findings.

6 Related Work

Previous work showed that the modern Hopfield Network, when made continuous and optimized differently, becomes Attention [25]. This result was one motivation for the current work because Hopfield Networks are another associative memory model. In fact, it has been shown that SDM is a generalization of the original Hopfield Network (Appendix B.6) [27]. While SDM is a generalization of Hopfield Networks, their specific discrepancies provide different perspectives on Attention. Most notably, Hopfield Networks assume symmetric weights that create an energy landscape, which can be powerfully used in convergence proofs including showing that one step convergence is possible for the modern Hopfield Network, and by proxy, Attention and SDM when it is a close approximation [25]. However, these symmetric weights come at the cost of biological plausibility that SDM compellingly provides in addition to its unique geometric framework and relation to Vector Symbolic Architectures [27].

Other works have tried to reinterpret or remove the softmax operation from Attention because the normalizing constant can be expensive to compute [46, 47]. However, while reducing computational cost, these papers show that removing the softmax operation harms performance. Meanwhile, SDM not only shows how Attention can be written as a Feedforward model [15] but also reveals that through simple binary read and write operations, (the neuron is either within Hamming/cosine distance or it's not) the softmax function emerges and need not be removed.

Since the publication of SDM, there have been a number of advancements not only to SDM specifically, but also through the creation of related associative memory algorithms under the name of "Vector Symbolic Architectures" [48]. Advancements to SDM include using integer rather than binary

vectors [49], handling correlated patterns [27], and hierarchical data storage [50]. Vector Symbolic Architectures, most notably Holographic Reduced Representations, have ideas that can be related back to SDM and the Transformer in ways that may be fruitful [51, 52, 53, 54, 55, 56].

The use of external memory modules in neural networks has been explored most notably with the Neural Turing Machine (NTM) and its followup, the Differentiable Neural Computer (DNC) [23, 57]. In order to have differentiable read and write operations to the external memory, they use the softmax function. This, combined with their use of cosine similarity between the query and memory locations, makes both models closely related to SDM. A more recent improvement to the NTM and DNC directly inspired by SDM is the Kanerva Machine [24, 58, 59, 60]. However, the Kanerva Machine remains distinct from SDM and Attention because it does not apply the a Hamming distance threshold on the cosine similarity between its query and patterns. Independent of these discrepancies, we believe relating these alternative external memory modules to SDM presents a number of interesting ideas that will be explored in future work.

7 Discussion

The result that Attention approximates SDM should enable more cross pollination of ideas between neuroscience, theoretical models of associative learning, and deep learning. Considering avenues for future deep learning research, SDM's relationship to Vector Symbolic Architectures is particularly compelling because they can apply logical and symbolic operations on memories, making SDM more powerful [52, 61, 62, 63]. SDM and its relation to the brain can inspire new research in not only deep learning but also neuroscience, because of the empirical success of the Transformer and its relation to the cerebellum, via SDM.

Limitations One limitation of SDM is that Attention remains an approximation rather than the equality that Hopfield Networks established [25]. However, what SDM loses in its precise mapping to Attention it gains in stronger biological plausibility. The best efforts to date to make the modern Hopfield Network biologically plausible still fail to address the weight transport problem that is a core model component SDM is free of [64].

Beyond motivating specific research agendas in deep learning and neuroscience, we hope that our results serve as an existence proof for how powerful deep learning equations can be approximated by, and mapped onto, the functional attributes and connectivity patterns of neuronal populations. At a time when many new neuroscientific tools are mapping out uncharted neural territories, we hope that more discoveries along the lines of this work connecting deep learning to the brain will be made [65, 66, 67].

8 Conclusion

We have shown that the Attention update rule closely approximates SDM when made continuous, L^2 norms its vectors, and has an appropriate β coefficient. This result has been shown to hold true in both theory and empirical evaluation of GPT2 Transformer models. SDM predicts that Transformers should normalize their key, query and value vectors, preempting the development of Query-Key Normalization and adding nuance to the interpretation of Attention weights. In mapping SDM onto the Transformer architecture as a whole, relating it to other external memory implementations, and highlighting extensions to SDM, we lay the groundwork for future deep learning research directions. In SDM being a generalization of Hopfield Networks, we outline how it encompasses previous results while contributing new connectome level biological plausibility. More broadly, our work highlights a convergence between deep learning and neuroscience, providing support for how the brain may implement, and deep learning may approximate, associative memory.

Acknowledgements

Thanks to Alex Cuozzo, Pentti Kanerva, Joe Choo-Choy, Beren Millidge, Jacob Zavatone-Veth, Blake Bordelon, Nathan Rollins, Alan Amin, Max Farrens, David Rein, Sam Eure, Grace Bricken, and Davis Brown for providing invaluable inspiration, discussions and feedback. Special thanks to Miles Turpin for help working with the Transformer model experiments. We would also like to thank the open source software contributors that helped make this research possible, including but not limited to: Numpy, Pandas, Scipy, Matplotlib, PyTorch, HuggingFace, and Anaconda.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[3] Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications. *OpenAI Blog https://openai.com/blog/better-language-models*, 2019.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[5] Mark Chen, A. Radford, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML 2020*, 2020.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[7] Anonymous. Videogen: Generative modeling of videos using {vq}-{vae} and transformers. In *Submitted to International Conference on Learning Representations*, 2021. under review.

[8] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, and Scott Gray. Dall·e: Creating images from text, Jan 2021.

[9] Gwern. May 2020 news 'on gpt-3', Dec 2019.

[10] Anna Rogers, O. Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[11] Martin Schrimpf, I. Blank, Greta Tuckute, Carina Kauf, Eghbal A. Hosseini, N. Kanwisher, J. Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative reverse-engineering converges on a model for predictive processing. *bioRxiv*, 2020.

[12] Shikhar Tuli, Ishita Dasgupta, Erin Grant, and T. Griffiths. Are convolutional neural networks or transformers more like human vision? *ArXiv*, abs/2105.07197, 2021.

[13] Pentti Kanerva. *Sparse distributed memory*. MIT Pr., 1988.

[14] Marvin Minsky and Papert Seymour. *Time vs. memory for best matching - an open problem*, page 222 – 225. 1969.

[15] P. Kanerva. Sparse distributed memory and related models. 1993.

[16] M. Modi, Yichun Shuai, and G. Turner. The drosophila mushroom body: From architecture to algorithm in a learning circuit. *Annual review of neuroscience*, 2020.

[17] G. Wolff and N. Strausfeld. Genealogical correspondence of a forebrain centre implies an executive brain in the protostome–deuterostome bilaterian ancestor. *Philosophical Transactions of the Royal Society: Biological Sciences*, 371, 2016.

[18] Ashok Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, and L. Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93:1153–1164.e7, 2017.

[19] T. Shomrat, Nicolas Graindorge, C. Bellanger, G. Fiorito, Y. Loewenstein, and B. Hochner. Alternative sites of synaptic plasticity in two homologous "fan-out fan-in" learning and memory networks. *Current Biology*, 21:1773–1782, 2011.

[20] S. Shigeno and C. W. Ragsdale. The gyri of the octopus vertical lobe have distinct neurochemical identities. *Journal of Comparative Neurology*, 523, 2015.

[21] M. Kawato, Shogo Ohmae, and Terry Sanger. 50 years since the marr, ito, and albus models of the cerebellum. *Neuroscience*, 462:151–174, 2021.

[22] A. Henry, Prudhvi Raj Dachapally, S. Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *EMNLP*, 2020.

[23] A. Graves, G. Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014.

[24] Yan Wu, Greg Wayne, Alex Graves, and Timothy Lillicrap. The kanerva machine: A generative distributed memory, 2018.

[25] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need, 2020.

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[27] James D. Keeler. Comparison between kanerva's sdm and hopfield-type neural networks. *Cognitive Science*, 12(3):299 – 329, 1988.

[28] David E. Rumelhart and David Zipser. Feature discovery by competitive learning. 1985.

[29] Guang Yang, Xiang Chen, K. Liu, and C. Yu. Deeppseudo: Deep pseudo-code generation via transformer and code feature extraction. *ArXiv*, abs/2102.06360, 2021.

[30] Sainbayar Sukhbaatar, E. Grave, Guillaume Lample, H. Jégou, and Armand Joulin. Augmenting self-attention with persistent memory. *ArXiv*, abs/1907.01470, 2019.

[31] Mor Geva, R. Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *ArXiv*, abs/2012.14913, 2020.

[32] N. Carlini, Florian Tramèr, Eric Wallace, M. Jagielski, Ariel Herbert-Voss, K. Lee, A. Roberts, Tom Brown, D. Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. *ArXiv*, abs/2012.07805, 2020.

[33] Kevin Lu, Aditya Grover, P. Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *ArXiv*, abs/2103.05247, 2021.

[34] Jesse Vig. Visualizing attention in transformer-based language representation models. *ArXiv*, abs/1904.02679, 2019.

[35] Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. The language interpretability tool: Extensible, interactive visualizations and analysis for nlp models. In *EMNLP*, 2020.

[36] Joris Baan, Maartje ter Hoeve, M. V. D. Wees, Anne Schuth, and M. Rijke. Do transformer attention heads provide transparency in abstractive summarization? *ArXiv*, abs/1907.00570, 2019.

[37] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *ArXiv*, abs/1905.10650, 2019.

[38] J Alammar. Interfaces for explaining transformer language models, 2020.

[39] Yubei Chen, Dylan M. Paiton, and Bruno A. Olshausen. The sparse manifold transform. In *NeurIPS*, 2018.

[40] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. *ArXiv*, abs/2009.14794, 2021.

[41] Eriola Hoxha, F. Tempia, P. Lippiello, and M. C. Miniaci. Modulation, plasticity and pathophysiology of the parallel fiber-purkinje cell synapse. *Frontiers in Synaptic Neuroscience*, 8, 2016.

[42] Louis A. Jaeckel. A class of designs for a sparse distributed memory. 1989.

[43] Eren Sezener, Agnieszka Grabska-Barwinska, Dimitar Kostadinov, Maxime Beau, Sanjukta Krishnagopal, David Budden, Marcus Hutter, Joel Veness, Matthew M. Botvinick, Claudia Clopath, Michael Häusser, and Peter E. Latham. A rapid and efficient learning rule for biological neural circuits. *bioRxiv*, 2021.

[44] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275:213 – 215, 1997.

[45] Charles R. Gallistel. The coding question. *Trends in Cognitive Sciences*, 21:498–508, 2017.

[46] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Franccois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *ArXiv*, abs/2006.16236, 2020.

[47] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021.

[48] P. Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1:139–159, 2009.

[49] R. Prager and F. Fallside. The modified kanerva model for automatic speech recognition. *Computer Speech Language*, 3:61–81, 1989.

[50] L. Manevitz and Yigal Zemach. Assigning meaning to data: Using sparse distributed memory for multilevel cognitive tasks. *Neurocomputing*, 14:15–39, 1997.

[51] Ivo Danihelka, Greg Wayne, B. Uria, Nal Kalchbrenner, and A. Graves. Associative long short-term memory. In *ICML*, 2016.

[52] T. Plate. Holographic reduced representations: Convolution algebra for compositional distributed representations. In *IJCAI*, 1991.

[53] P. Kanerva. Binary spatter-coding of ordered k-tuples. In *ICANN*, 1996.

[54] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46:159–216, 1990.

[55] R. Gayler. Multiplicative binding, representation operators analogy. 1998.

[56] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10:23, 2016.

[57] A. Graves, Greg Wayne, M. Reynolds, T. Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, J. Agapiou, Adrià Puigdomènech Badia, K. Hermann, Yori Zwols, Georg Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

[58] Y. Wu, Gregory Wayne, K. Gregor, and T. Lillicrap. Learning attractor dynamics for generative memory. In *NeurIPS*, 2018.

[59] K. Gregor, Danilo Jimenez Rezende, Frédéric Besse, Y. Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. *ArXiv*, abs/1906.09237, 2019.

[60] Adam Marblestone, Yan Wu, and Greg Wayne. Product kanerva machines: Factorized bayesian memory, 2020.

[61] C. Eliasmith. How to build a brain: A neural architecture for biological cognition. 2013.

[62] B. Lake, Tomer D. Ullman, J. Tenenbaum, and S. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2016.

[63] S. Piantadosi. The computational origin of representation. *Minds and Machines*, 31:1–58, 2021.

[64] D. Krotov and J. Hopfield. Large associative memory problem in neurobiology and machine learning. *ArXiv*, abs/2008.06996, 2020.

[65] S. Alon, Daniel R. Goodwin, Anubhav Sinha, A. Wassie, Fei Chen, Evan R. Daugharthy, Yosuke Bando, Atsushi Kajita, Andrew G. Xue, Karl Marrett, Robert Prior, Yi Cui, A. Payne, Chun-Chen Yao, Ho-Jun Suk, Ru Wang, Chih chieh Yu, Paul W. Tillberg, P. Reginato, N. Pak, S. Liu, Sukanya Punthambaker, Eswar P. R. Iyer, Richie E. Kohman, J. Miller, E. Lein, Ana Lako, N. Cullen, S. Rodig, K. Helvie, Daniel L. Abravanel, N. Wagle, B. Johnson, J. Klughammer, Michal Slyper, Julia Waldman, J. Jané-Valbuena, O. Rozenblatt-Rosen, A. Regev, G. Church, Adam H. Marblestone, and E. Boyden. Expansion sequencing: Spatially precise in situ transcriptomics in intact biological systems. *bioRxiv*, 2020.

[66] C. Xu, Michał Januszewski, Z. Lu, S. Takemura, K. Hayworth, G. Huang, K. Shinomiya, Jeremy B. Maitin-Shepard, David Ackerman, Stuart E. Berg, T. Blakely, J. Bogovic, Jody Clements, Tom Dolafi, Philip M. Hubbard, Dagmar Kainmueller, W. Katz, Takashi Kawase, K. Khairy, Laramie Leavitt, Peter H. Li, Larry Lindsey, Nicole L. Neubarth, D. J. Olbris, H. Otsuna, Eric T. Troutman, L. Umayam, Ting Zhao, M. Ito, Jens Goldammer, T. Wolff,

R. Svirskas, P. Schlegel, E. Neace, Christopher J Knecht, Chelsea X. Alvarado, Dennis A Bailey, Samantha Ballinger, J. Borycz, Brandon S Canino, Natasha Cheatham, Michael Cook, M. Dreher, Octave Duclos, Bryon Eubanks, K. Fairbanks, S. Finley, N. Forknall, Audrey Francis, Gary Hopkins, Emily M Joyce, Sungjin Kim, Nicole A. Kirk, Julie Kovalyak, Shirley Lauchie, Alanna Lohff, Charli Maldonado, Emily A. Manley, Sari McLin, C. Mooney, Miatta Ndama, Omotara Ogundeyi, Nneoma Okeoma, Christopher Ordish, Nicholas L. Padilla, Christopher Patrick, Tyler Paterson, Elliott E Phillips, E. M. Phillips, Neha Rampally, Caitlin Ribeiro, Madelaine K Robertson, J. Rymer, S. Ryan, Megan Sammons, Anne K. Scott, Ashley L. Scott, A. Shinomiya, C. Smith, Kelsey Smith, N. L. Smith, Margaret A. Sobeski, Alia Suleiman, J. Swift, Satoko Takemura, Iris Talebi, Dorota Tarnogorska, Emily Tenshaw, Temour Tokhi, J. Walsh, Tansy Yang, J. Horne, Feng Li, Ruchi Parekh, P. Rivlin, V. Jayaraman, K. Ito, S. Saalfeld, R. George, I. Meinertzhagen, G. Rubin, H. Hess, Louis K. Scheffer, Viren Jain, and Stephen M. Plaza. A connectome of the adult drosophila central brain. *bioRxiv*, 2020.

[67] Kristin M Scaplen, M. Talay, John D Fisher, Raphael Cohn, Altar Sorkaç, Y. Aso, G. Barnea, and K. Kaun. Transsynaptic mapping of drosophila mushroom body output neurons. *bioRxiv*, 2020.

[68] Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *ArXiv*, abs/1910.05895, 2019.

[69] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, S. Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, L. Wang, and T. Liu. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745, 2020.

[70] W. Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961, 2021.

[71] Aäron van den Oord, Oriol Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017.

[72] M. Nour. Surfing uncertainty: Prediction, action, and the embodied mind. *British Journal of Psychiatry*, 210:301 – 302, 2017.

[73] Yongjae Lee and Woo Chang Kim. Concise formulas for the surface area and intersection of two hyperspherical caps. *KAIST Technical Report*, 2017.

[74] Louis A. Jaeckel. An alternative design for a sparse distributed memory. 1989.

[75] P. Chou. The capacity of the kanerva associative memory. *IEEE Trans. Inf. Theory*, 35:281–298, 1989.

[76] Gordon M. Shepherd. *The Synaptic Organization of the Brain*. Oxford University Press, 2004.

[77] E. D'Angelo and Stefano Casali. Seeking a unified framework for cerebellar function and dysfunction: from circuit operations to cognition. *Frontiers in Neural Circuits*, 6, 2013.

[78] D. Timmann and I. Daum. Cerebellar contributions to cognitive functions: A progress report after two decades of research. *The Cerebellum*, 6:159–162, 2008.

[79] Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[80] D. Krotov and J. Hopfield. Dense associative memory for pattern recognition. In *NIPS*, 2016.

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

[82] M. Demircigil, Judith Heusel, M. Löwe, Sven Upgang, and F. Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168:288–299, 2017.

[83] J. Hopfield and D. Tank. "neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 2004.

[84] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[85] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[86] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.