Foundations of Attention Mechanisms in Deep Neural Network Architectures

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the foundations of attention mechanisms in deep neural network architectures and present three main results. First, we provide a systematic taxonomy of all possible attention mechanisms within, or as extensions of, the McCulloch and Pitt standard model into 18 classes depending on the origin type of the attention signal, the target type of the attention signal, and whether the interaction type is additive or multiplicative. Second, using this taxonomy, we identify three key attention mechanisms: output gating, synaptic gating, and multiplexing. Output gating and synaptic gating are extensions of the standard model and all current attention-based architectures, including transformers, use either output gating or synaptic gating, or a combination of both. Third, we develop a theory of attention capacity and derive mathematical results about the capacity of basic attention networks. For example, the output gating of a linear threshold gate of n variables by another linear threshold gate of the same n variables has capacity $2n^2(1 + o(1))$. Perhaps surprisingly, multiplexing attention is used in the proofs of these results. Synaptic and output gating provide computationally efficient extensions of the standard model allowing for *sparse* quadratic activation functions. They can also be viewed as primitives enabling the concise collapsing of multiple layers of processing in the standard model.

1 Introduction

The motivation for studying attention in deep learning models, or artificial neural networks, is two-fold. The first motivation is to avoid getting bogged down by the complexity of biological systems. There is of course a vast literature on the neurobiology and psychophysics of attention (e.g. [13, 2, 19]) pointing to a variety of different phenomena and attention systems, leading some to conclude at the end of a review: "*The word"attention" is an inadequate, singular term for a multitude of inter-related processes. We use a host of adjectives to describe attention—-for example, we say that attention can be divided, oriented, sustained, or focused, and many of these descriptions likely reflect underlying, dissociable neural processes. Complicating matters, attentional resources can be allocated to either external stimuli, or to internal stimuli such as thoughts and memories. Furthermore, we often confuse the regulation of attention (a covert behavior) with the regulation of movement (an overt behavior) when discussing an "attentional disorder"*" [2]. In spite of this complexity and diversity of processes, we believe that at the most fundamental level attention mechanisms are built out of a small number of fundamental operations, which occur on time scales that are fast compared to the time scales for learning and long-term synaptic modifications. In particular, in order to exclude other stimuli, which is the hallmark of attention, neuronal machinery must exist that is capable of dynamically suppressing the activity of subsets of neurons, or subsets of connections, or both, associated with the non-attended information. These fundamental operations may be easier to identify and study using artificial neural networks. Thus, one of our goals here is to produce a systematic nomenclature of all such possible operations, within the standard deep learning formalism. While this is not the place to

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

discuss the relationship between artificial and biological neural networks, there is a body of evidence showing that, at least at some level, the former can provide useful information about the latter (e.g. [25, 18, 24]).

The second obvious motivation is that attention plays an increasingly important role in deep learning systems and their numerous applications. Over the past decade, various attention mechanisms such as content-based attention [12], speech recognition attention [8], or dot product attention [17], have been introduced and successfully deployed in applications. The current pinnacle of attention-based architectures is the transformer architecture [23, 22] which has led to state-of-the-art performance in NLP and is now widely used. Many of these attention mechanisms were initially developed for speech and natural language applications (NLP) (e.g. [3, 9, 20]), but they are now being adapted to other problems (e.g. [15, 11]). However, with rare exceptions [10], there is little theory to help us better understand the nature and computational capabilities of attention. To begin to address some of these issues, we first need to specify the computational framework within which attention mechanisms are to be studied. This is what we call the standard model.

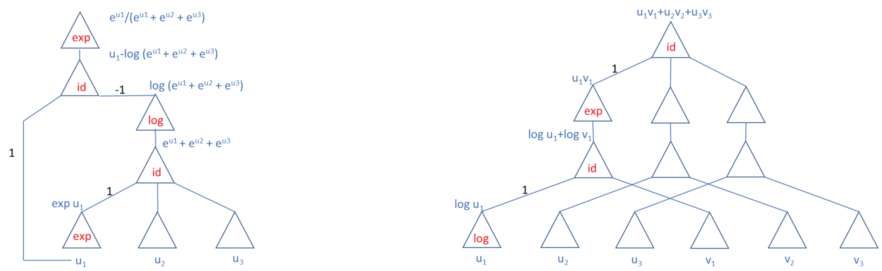
1.1 The Standard Model (SM)

The Standard Model is the class of all neural networks made of what are generally called McCulloch and Pitt neurons. Neural networks in the SM consist of directed weighted graphs of interconnected processing units, or neurons. The synaptic strength of the connection from neuron j to neuron i is represented by a single real-valued number w_{ij}. A neuron i produces an output O_i by first computing an activation $S_i = \sum_j w_{ij} O_j$, i.e the activation corresponds to the dot product of the incoming signal with the synaptic weights. In turn, the output of the neuron is produced in the form $O_i = f_i(S_i)$ where f_i is the transfer or activation function of neuron i. Typical activation functions include the identity function in the case of linear neurons, sigmoidal activation functions such as the logistic and tanh activation functions, and piece-wise linear functions ([21]), such as the Heaviside, sign, or ReLU functions. An encompassing, and more than sufficient, class of transfer functions for a formal definition of the SM is the class of functions that are differentiable everywhere except for a finite (and small) set of points. A fundamental, and easy to prove [5], property of the SM is that it has universal approximation properties: (1) any Boolean function can be implemented exactly by a feed-forward network in the SM; and (2) for any small $\epsilon > 0$, any continuous function from $\mathbb{R}^n$ to $\mathbb{R}^m$ defined on a compact set C can be approximated within ϵ everywhere over C by a feed-forward network in the SM. Several attention mechanisms described below can be viewed as extensions of the standard model, where new operations are added to the SM to obtain a richer model. Extending the SM is not a new procedure. For instance, using softmax layers is already an extension of the SM since the softmax is not a proper, single-neuron, activation function. Another example is the use of polynomial activation functions (e.g. [7]). Due to the universal approximation properties of the SM, these extensions are not meant to increase the approximating power of the SM. Rather, their value must be established along other dimensions, such as circuit size or learning efficiency. In the digital simulations of neural networks, these extensions correspond to new software primitives. In physical neural networks, these extensions must come with actual wires and physical mechanisms. For instance, a softmax operation is a new software primitive in a neural network software library but it requires a new physical mechanism for its physical implementation. It can be replaced by a network of depth 3 within the SM with weights set to ± 1 (Figure 1a), provided logarithm and exponential activation functions are available. Using other activations functions (e.g. ReLU) could require an even deeper circuit. Similar observations can be made for the dot product of two vectors (Figure 1b).

2 A Systematic Taxonomy of Attention Mechanisms

In the SM, there are three kinds of variable types: S (activations), O (outputs), and w (synaptic weights). At the most fundamental level, we can organize attention mechanisms (and more broadly new SM interactions) depending on: the type of variable associated with the *source* of an attention signal (3 possibilities), the type of variable associated with the *target* of an attention signal (3 possibilities), and on the *mechanism* of the interaction, i.e. on the algebraic operation used to combine the attending signal and the attended target. While many algebraic operations can be considered, the two most basic ones are addition and multiplication (two possibilities)–resulting in a total of 18 different possibilities.

Source: It is reasonable to assume that the source of the attending signal is a variable of type O corresponding to the output of one attending neuron, or a group (layer) of attending neurons. While other possibilities can be explored, e.g. a synapse directly attending another synapse, they would



(a) Softmax. (b) Dot product.

Figure 1: (a) Neural network for computing the softmax function for a vector (u_1, u_2, u_3) in the SM. For clarity, only the circuit for the first component of the softmax is shown in full. (b) Neural network for computing the dot product of $u = (u_1, u_2, u_3)$ with $v = (v_1, v_2, v_3)$ in the SM. Across both cases, all the weights are fixed and equal to either -1 or +1. The transfer functions used are $\log$, $\exp$, and the identity.

require new complex mechanisms in a physical implementation. Furthermore, they do not occur in current attention-based deep learning models. The same can be said for the activation being the direct source of the attending signal. Even more unlikely would be the case of mixed schemes where the attending signal would emanate, for instance, from both neuronal outputs and synapses. In short, the reasonable assumption that the attending signals emanate from neuronal outputs allows us to reduce the number of possibilities by a factor of three leaving 6 basic possibilities (Table 1.

Target: For the target of an attention signal, we will study all three possibilities. Thus attention signals can target activations (S), outputs (O), or synapses (w). We will call these three forms of attention activation attention, output attention, and synaptic attention respectively.

Mechanism: The most simple operations one can think of for combining the attending signal with its attended target are addition and multiplication. Note that both addition and multiplication are differentiable operations, and thus can easily be incorporated into the backpropagation learning framework. Attention requires excluding all other stimuli and possibly enhancing the attended stimulus (here we do not distinguish between external stimulus or internal representation). Intuitively, at the fundamental level, these exclusions and enhancements correspond to multiplicative operations where, for instance, the signals associated with non-attended stimuli are inhibited–i.e. multiplied by zero, and the attended stimuli are enhanced, i.e. mutliplied by a factor greater than one. We will reserve the term "gating" for multiplicative interactions. Thus, for instance, multiplicative synaptic attention will also be called synaptic gating (Figure 2). All multiplicative interactions, with the exceptions of terms of the form $w_{ij}O_j$, are not part of the SM and thus correspond to potential extensions of the SM. For completeness, we will also consider the case of additive interactions. Furthermore, in the case of additive activation attention, for several common activation functions such as logistic or ReLU, inhibition (and thus suppression of stimuli) can be achieved additively by sending a large negative signal towards the attended neuron. This is also called multiplexing since the attending signal is multiplexed with the regular signal. Unlike gating, additive activation attention is contained in the SM. Further inspection of Table 1 reveals that among the 6 possibilities some are uninteresting (additive output attention) or subsumed by other mechanisms. For instance, gating of a neuron's activation by an attending neuron is equivalent to synaptic gating all its incoming synapses.

Multiplicities: Finally, in each possible case, one must take into account multiplicity issues both at the level of the source and at the level of the target. For instance, in synaptic gating, can the attending output of a neuron gate more than one synapse? Can the attending output of several neurons gate the same synapse? And so forth. In the most simple cases, we will assume that the multiplicity is one both at the source and at the target, but greater multiplicities will also be considered.

In summary, we are left with six main cases, corresponding to two different mechanisms $(+, \times)$ and three different target types (S, O, w). These can be further reduced to three most important mechanisms, marked in bold in (Table 1): synaptic gating, output gating, and multiplexing.

3 Attention-Based Architectures and Transformers:All you Need is Gating

Although the descriptions of attention mechanisms in deep learning often seem complex and sometimes obscure the underlying neural architecture [12, 8, 17, 3, 9, 20], it can be checked that in all cases

Table 1: Organization of attention mechanisms. Assuming that the origin of the attention signal is the output of one or several neurons, there are 6 classes depending on the target of the signal and the interaction mechanism. We consider 3 kinds of targets: activation (S), output (O), and synapses (w). We consider 2 kinds of interaction mechanisms: addition and multiplication. Two of the classes (additive activation attention, or multiplexing, and additive output attention) are in the SM; the other 4 classes correspond to true extensions of the SM. Further inspection shows one can focus on three classes only: multiplexing, output gating, and synaptic gating (in bold).

	S	O	w
Addition	**multiplexing** (SM)	additive output att.(SM)	aditive synaptic att.
Multiplication	activation gating	**output gating**	**synaptic gating**

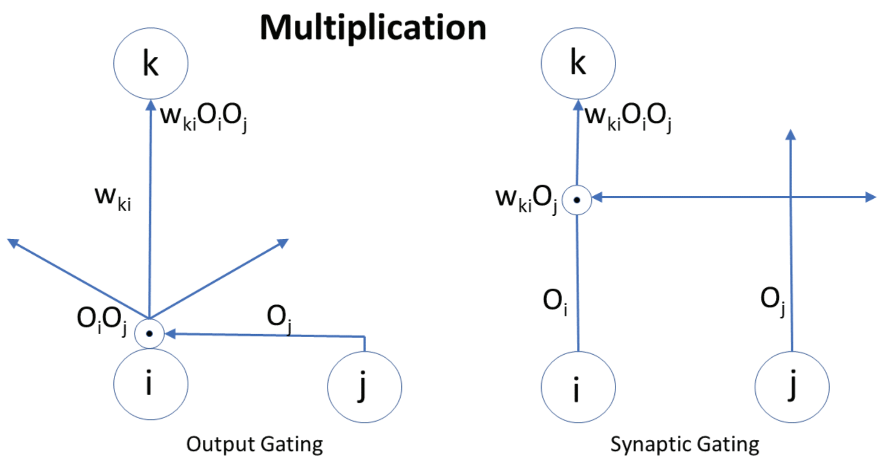


Figure 2: Multiplicative Interactions: Output and Synaptic Gating. Left: In output gating, neuron j gates the output of neuron i producing a new effective output O_iO_j. The signal O_iO_j is broadcasted to all the neurons downstream of neuron i, including neuron k. Right: In synaptic gating, neuron j gates the synapse between neuron i and neuron k, producing a new effective synaptic weight equal to $w_{ki}O_j$. In both cases, the signal O_j can be transmitted to other neurons and other synapses (higher multiplicity). In both cases, neuron k receives the same signal $w_{ki}O_iO_j$. However the effects of output versus synaptic gating on the rest of the network are different (see text).

these are built out of the output and synaptic gating attention mechanisms described in the previous section. For conciseness, here we demonstrate this briefly only for the transformer architectures [23, 22] (see also [16] for an MLP alternative to transformers). These architectures typically consist of stacks of encoder and decoder modules, with attention mechanisms in each module. The encoder and decoder modules are very similar so it is sufficient to examine an encoder module. Let us assume that an encoder module has n input vectors. Each vector is first transformed into three vectors, called the Query, Key, and Value Vectors, typically via a shared linear transformation which can easily be represented in the SM via weight sharing. They Query and Key vectors must have the same dimension m. Then a transformer computes all the dot products between the query vectors and the key vectors. Dot product operations are not directly available in the SM but can easily implemented by using output gating. The dot product of the layer of activities $(q_1, \ldots, q_m)$ with the layer of activities $(k_1, \ldots, k_m)$ is computed by letting k_1 gate q_1, k_2 gate q_2, and so forth. All the gated outputs are then connected to a linear unit, with all incoming weights equal to one, in order to produce the dot product $\sum q_ik_i$. The transformer then applies a softmax to each row of the matrix of n^2 dot products. Finally, each output vector of the encoder module is computed by taking a convex combination of the n value vectors, where the weights of the convex combination are provided by a softmax applied to the corresponding row of the matrix of n^2 dot products. This can be implemented by using synaptic gating, where all the weights between value vectors and output vectors are equal to one and each weight is modulated by the corresponding softmax component. The convex combination of the value vectors by the corresponding softmax weights determines how much each value vector influences each output vector, based on the corresponding similarities between Q vectors and K vectors. This is where the influence of some of the value vectors can be enhanced, while the influence of others can be suppressed. Thus in total there are mn^2 output gating operations, and n^2 synaptic gating operations (assuming n output vectors). In multi-head attention, the same mechanisms are replicated several times and the same analyses apply. In short, the fundamental building block of a transformer consist of a sequence of three macro operations–dot products, softmax, and convex combinations that would require a dozen of layers to implement in the SM. These can be implemented much more economically by using output gating to compute the dot products and synaptic gating to implement the convex combinations.

4 The Capacity of Attention

We have seen that attention mechanisms enable important functionalities with minimal depth compared to the equivalent SM circuits, at the cost of adding attention neurons and mechanisms. Here we want to better understand the trade offs between the computations that are enabled and the corresponding costs. One key concept for doing so is the concept of neuronal capacity [6].

4.1 Definition of Capacity:

Given a class of functions $\mathcal{A}$, we define its cardinal capacity $C(\mathcal{A})$, or just capacity, to be: $C(\mathcal{A}) = \log_2 |\mathcal{A}|$, where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$ in the finite case. In the continuous case, $|\mathcal{A}|$ can be defined as a volume, but here we will focus primarily on finite cases. The class $\mathcal{B}_n$ of all Boolean functions of n variables has capacity $C(\mathcal{B}_n) = 2^n$. Here we will consider sub-classes of $\mathcal{B}_n$, in particular those implemented by feed-forward networks of linear or polynomial threshold gates, with attention mechanisms, and compute the corresponding capacity. Using linear or polynomial threshold functions is not particularly restrictive since these are reasonably good approximations of linear- or polynomial-activation neurons with steep sigmoidal transfer functions. Furthermore, the universal approximation properties of the SM can be established while using only linear (or polynomial) threshold functions in the hidden layers.

4.2 Linear and Polynomial Threshold Functions

A polynomial threshold functions of degree d has the form $\operatorname{sgn} p(x)$, where $p(x)$ is a polynomial of degree d using a $-/+$ output representation. Alternatively, for a 0/1 output representation, we can use the form $H(p(x)$ where H is the Heaviside function equal to 0 for $x \leq 0$ and to 1 otherwise. Units with values in $0/1$ are similar to logistic sigmoidal units, and units with values in $-1/+1$ are similar to $\tanh$ sigmoidal units. We let $\mathcal{T}(n;d)$ denote the class of polynomial threshold functions of degree d. Thus $\mathcal{T}(n;1)$ denotes the class of linear threshold functions. When the inputs to a threshold function are binary, we use the term threshold gate. In the case of polynomial threshold gates, it does not matter whether their input is encoded using $0/1$ or $-/+$ (or for that many any two distinct real numbers). This is because there is an affine transformation between any two such encodings and the affine transformation can be absorbed in the synaptic weights, i.e. the coefficients of p. The same is generally true for the encoding of the output, however when attention gating is considered the $0/1$ and $-/+$ encodings behave differently. For instance, in the case of output gating, the product of two $0/1$ threshold gates behaves like an AND, whereas the output of two $-/+$ gates behaves like an NXOR.

Thus to derive more general results, we will consider the case where the gating mechanism is implemented by a Boolean function B, which could be an AND, an NXOR, or something else. In the most general setting, we let $B(z_1, \ldots, z_k) : \{-1,1\}^k \rightarrow \{-1,1\}$ be a Boolean formula in k variables. We are interested in the class of functions of the form $B(f_1, ..., f_k) : \{0,1\}^n \rightarrow \{-1,1\}$ where $f_j \in \mathcal{T}(n;d_j)$. We denote this class by $\mathcal{T}_B(n;d_1,\ldots,d_k)$.

4.3 Why Capacity is Important

The capacity $C(\mathcal{A})$ is a measure of what the class of functions $\mathcal{A}$ can do. As a single number, it is of course a very crude representation of the true functional capacity. However in the case of neural networks the capacity has a stronger significance. To see this, note first that the cardinal capacity is also the number of bits required to specify an element of $\mathcal{A}$. Thus in the case of neural networks, to a first order of approximation, the capacity is the number of bits that must be transferred from the training data to the synaptic weights during learning for the network to learn to implement a specific function in the class $\mathcal{A}$.

4.4 Capacity of Single Units: Review

Before we estimate the capacity of single units with attention mechanisms, we must review the known capacity results on single units without attention mechanisms. For a single linear threshold gate of n variables, we have [26, 27]:

$$\left(1 - \frac{10}{\log n}\right) n^2 \leq C(\mathcal{T}(n;1)) \leq n^2 \tag{4.1}$$

This result was refined to the form [14]:

$$C(\mathcal{T}(n;1)) = n^2 - n \log_2 n \pm O(n) \tag{4.2}$$

Similar results have been obtained for polynomial threshold gates of degree d [4, 7]. In particular, for any n and d satisfying $1 \leq d \leq n^\alpha$ (where α is fixed and $< \alpha < 1$) there exists a constant $D = D(\alpha)$ such that:

$$(1 - \frac{D}{\log n})^d n \binom{n}{\leq d} \leq C(\mathcal{T}(n;d)) \approx n \binom{n}{\leq d} \tag{4.3}$$

where:

$$\binom{n}{\leq d} = \sum_{k=0}^{d} \binom{n}{k} \tag{4.4}$$

For degree $d = o(\log n)$, including fixed degree d, Equation 4.3 yields:

$$C(\mathcal{T}(n;d)) = \frac{n^{d+1}}{d!}(1 - o(1)) \tag{4.5}$$

We can now move to the problem of estimating the capacity of attention mechanisms, first for single unit attention and then for layer-wise attention. Here, for conciseness, we focus on output gating alone, but we have derived similar results also for the case of synaptic gating.

4.5 Capacity of Attention: Single Unit Gating

Here we consider two linear threshold units with the same n inputs, where the output of one unit gates the output of the other units. We have seen that this corresponds to taking the AND or NXOR of the two units, depending on whether the outputs are coded using 0/1 or -/+. In short, we want to estimate how many Boolean functions can be written as the AND (or NXOR) of two linear (or polynomial) threshold gates?

The capacity of such a circuit has an obvious upperbound, given by the sum of the capacities of its components. Thus in the case of linear threshold gates, the capacity is upperbounded by $2n^2(1+o(1))$. The more difficult part is finding the lower bound. It turns out that the lower bound is equal to the upper bound so that we have the following theorem.

Theorem 4.1. *The capacity of a single linear threshold gate with n inputs, output-gated by another linear threshold gate of the same n inputs, is given by: $2n^2(1 + o(1))$.*

Note that this theorem shows that output gating is an *efficient* mechanism in the sense that no capacity is lost with respect to the maximum achievable capacity. In other words, the doubling of the number of parameters caused by the attending gate leads to a doubling of the capacity, which is the maximum achievable. This theorem is a special case of the following, more general theorem, that considers the combination of two or more, linear or polynomial, threshold gates combined using an arbitrary Boolean operator (not just AND and NXOR).

Theorem 4.2 (Composition)**.** *Let B be an irreducible Boolean operator in k variables.*[1] *Then:*

$$\prod_{j=1}^{k} \left|\mathcal{T}(n-k+1; d_j)\right| \leq \left|\mathcal{T}_B(n; d_1, \ldots, d_k)\right| \leq \prod_{j=1}^{k} \left|\mathcal{T}(n; d_j)\right| \tag{4.6}$$

[1] Irreducibility means that B can not be expressed as a Boolean operator in fewer than k variables.

Furthermore, if B is the set of all irreducible Boolean functions of two variables (there are 10 of them), we have:

$$\left|\bigcap_B \mathcal{T}_B(n; d_0, d_1)\right| \geq \left|\mathcal{T}(n-1; d_0)\right| \left|\mathcal{T}(n-1; d_1)\right| \tag{4.7}$$

where the intersection is over the ten irreducible binary Boolean operators.

By taking logarithms and applying Zuev's theorem, it is easy to see that Theorem 4.1 is a special case of Theorem 4.2, corresponding to $k = 2$, with AND or NXOR as the Boolean operator, and $d_1 = d_2 = 1$ for linear threshold gates. The complete proof of Theorem 4.2 is given in the Appendix. The key idea for proving this theorem is the use of multiplexing attention, which is used also in the proof of Theorem 4.3).

4.6 Capacity of Attention: Layer Gating

The previous attention results are obtained using only two neurons, a gating neuron and a gated neuron. We now extend the capacity analysis to the case where there is a layer of gating neurons output-gating a layer of attended neurons, as shown in Figure 3.

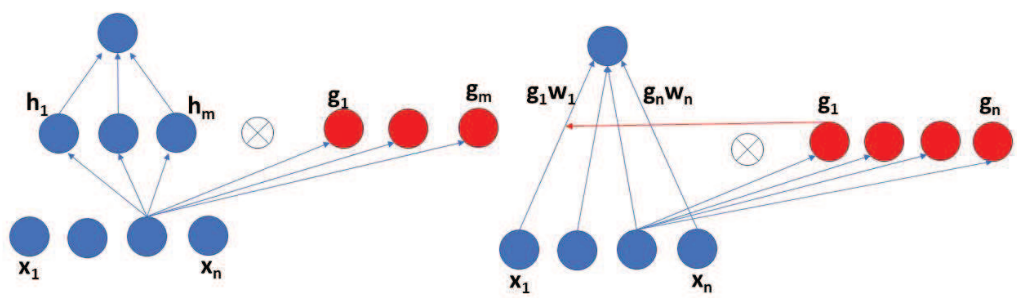


Figure 3: Left: output gating by a gating layer. For the same n dimensional input vector x, there are m hidden units computing functions $h_1(x), \dots, h_m(x)$, and m corresponding gating units computing functions $g_1(x), \dots, g_m(x)$. With the gating, the effective output of the hidden units is given by $h_1(x)g_1(x), \dots, h_m(x)g_m(x)$. The final output unit produces an output of the form $f(h_1(x)g_1(x), \dots, h_m(x)g_m(x))$. In the capacity analysis, we assume that the functions h, g, and f are linear threshold gates. Right: synaptic gating by a gating layer. In this case, there is a unit computing a function $f(x)$ with n weights $w_i, \dots, w_n$. There are n gating functions $g_1(x), \dots, g_n(x)$, each one multiplicatively gating one of the weights w. If $f = \text{sign}(\sum_i w_i x_i)$ then $f_g(x) = \text{sign}(\sum_i g_i(x) w_i x_i)$.

Thus we consider an architecture with n inputs, m hidden linear threshold units gated by m corresponding linear threshold units, and one final linear threshold output gate. All the linear threshold gates have $-/+$ outputs, although the following theorem is unchanged, and the method of proof is similar, if the gates have 0/1 outputs. We denote by $\mathcal{T}(n, m, 1; \times)$ the corresponding set of Boolean functions. Note that this is the same architecture for computing the dot product of the gated and the gating hidden layer outputs, except that the final unit is non-linear with variable weights, instead of being linear with fixed weights equal to one. We will also let $\mathcal{T}(n, 1; \times)$ denote the set of Boolean functions corresponding to one linear threshold gate of n variables output-gated by another linear threshold gate of the same variables.

Theorem 4.3. *The capacity $C(\mathcal{T}(n, m, 1; \times))$ of the set of Boolean functions corresponding to n inputs, m hidden linear threshold gates output-gated by m hidden linear threshold gates of the same inputs, followed by one linear threshold gate output satisfies:*

$$mn^2 \leq C(\mathcal{T}(n, m, 1; \times)) \leq 2mn^2 \left(1 + o(1)\right) \tag{4.8}$$

for $n \to \infty$, and for any choice of $m \in [1, 2^{o(n)}]$. Furthermore:

$$C(\mathcal{T}(n,m,1;\times)) = mC(T(n,1;\times))\left(1+o(1)\right) \tag{4.9}$$

Thus:

$$C(\mathcal{T}(n,m,1;\times)) = 2mn^2\left(1+o(1)\right) \tag{4.10}$$

Proof. Let us denote by f the map between the input layer and the hidden layer with gating, and by ϕ the map from the hidden layer to the output layer. For the upper bound, we first note that the total number of possible maps f is bounded by $2^{mC(\mathcal{T}(n,1;\times))} \leq 2^{2mn^2(1+o(1))}$, since f consists of m threshold gates gated by m threshold gates, and thus each gated unit corresponds to at most $2^{C(\mathcal{T}(n,1;\times))} \leq 2^{2n^2(1+o(1))}$ possibilities by Zuev's theorem. Any fixed map f, produces at most 2^n distinct vectors in the hidden layer. It is known [1] that the number of threshold functions ϕ of m variables defined on at most 2^n points is bounded by:

$$2\binom{2^n-1}{\leq m} = 2^{nm(1+o(1))} \tag{4.11}$$

using the assumption $m \leq 2^{o(n)}$. Thus, under our assumptions, the total number of functions of the form $\phi \circ f$ is bounded by the product of the bounds above which yields immediately:

$$C(\mathcal{T}(n,m,1;\times)) \leq mC(\mathcal{T}(n,1;\times))\left(1+o(1)\right) \leq 2mn^2\left(1+o(1)\right) \tag{4.12}$$

For the lower bound, we can force the gating units to be the identity (i.e. with a constant output equal to 1). In this particular case, the gating units can be ignored and we need to count the number of Boolean functions that can be implemented in the remaining architecture. A theorem in [6] shows that this number is equal to $mn^2(1+o(1))$.

To prove the rest of the theorem, we use attention multiplexing. The basic idea is to add a small (logarithmic) set of the input units that can be the source of a multiplexing attentional signal that can be used to select a particular function in the hidden layer. The same setting of these additional attention units will be used to select the corresponding functions in both the gating and gated layers. More formally, we decompose n as: $n = n^- + n^+$ where $n^- = \lceil \log_2 m \rceil$ corresponds to the attention units. Likewise, we decompose each input vector $x = (x_1, \ldots, x_n) \in \{-1,+1\}^n$ as: $x = (x^-, x^+)$, where:

$$x^- = (x_1, \ldots, x_{n^-}) \in \{-1,+1\}^{n^-} \quad \text{and} \quad x^+ = (x_{n^-+1}1, \ldots, x_n) \in \{-1,+1\}^{n^+} \tag{4.13}$$

For any gated Boolean linear threshold map f^+ from $\{-1,+1\}^{n^+}$ to $\{-1,+1\}^m$, we can uniquely derive a map $f = (f_1, \ldots, f_m)$ from $\{-1,+1\}^n$ to $\{-1,+1\}^m$ defined by:

$$f_i(x^-, x^+) = [x^- = i] \;\; AND \;\; [f_i^+(x^+)] \tag{4.14}$$

Here $x^- = i$ signifies that the binary vector x^- represents the digit i. In other words $x^- = i$ is used to select the i-th unit in the gated layer as well as in the gating layer, and filter f^+ by retaining only the value of f_i^+. This selection procedure can be expressed using a single linear threshold function of the input x^- for the gated layer, and similarly for the gating layer. We say that f is obtained from f^+ by multiplexing and f is a gated threshold map. It is easy to see that the filtering of two distinct maps f^+ and g^+ results into two distinct maps f and g. Now let us use $\phi = OR$ in the top layer–note that OR can be expressed as a linear threshold function. Then it is also easy to see that $\phi \circ f \neq \phi \circ g$. Thus the total number of Boolean functions that can be implemented in this architecture is lower-bounded by the number of all gated Boolean maps f^+. This yields:

$$C(\mathcal{T}(n,m,1;\times)) \geq mC(\mathcal{T}(n^+,1;\times))\left(1+o(1)\right) = 2mn^2\left(1+o(1)\right) \tag{4.15}$$

using the fact that $n^+ = n - \lceil \log_2 m \rceil$, and $\lceil \log_2 m \rceil = o(n)$ by assumption. Thus: $C(\mathcal{T}(n,m,1;\times)) = mC(\mathcal{T}(n,1;\times))\left(1+o(1)\right) = 2mn^2\left(1+o(1)\right)$. □

Remark 4.4. *In Theorem 4.3, we see again that both the capacity and the number of parameters approximately double at the same time.*

5 Conclusion

Within the framework provided by the SM, we have provided a taxonomy of possible attention mechanisms. Assuming three variable types and two kinds of interactions (additive or multiplicative) leads to 18 mechanisms, which can then be reduced to 6 by assuming that the attention signal emanates from the outputs of some neurons, and then down to three by removing redundancies: synaptic gating, output gating, and multiplexing. Synaptic gating and output gating are the fundamental building blocks of all existing attention-based architectures, including transformers. Finally, using the notion of capacity, we have analyzed attentional circuits in a quantitative manner, demonstrating their efficiency. Gating attentional mechanisms introduce quadratic activation terms, but in a parsimonious way that avoids the cost incurred by the use of full quadratic activations. They can also be viewed as coding primitives that effectively collapse multiple architectural layers into one construct.

Appendix: Detailed Proof of Theorem 4.2

Here a polynomial threshold function is a function of the form $f = \mathrm{sign}(p) : \{0,1\}^n \to \{-1,1\}$ where p is a polynomial in n real variables of degree at most d. The class of all such functions is denoted $\mathcal{T}(n;d)$. Let $B(z_1,\dots,z_k) : \{-1,1\}^k \to \{-1,1\}$ be a Boolean function in k variables. We are interested in the class of functions of the form $B(f_1,...,f_k) : \{0,1\}^n \to \{-1,1\}$ where $f_j \in \mathcal{T}(n;d_j)$. Denote this class by $\mathcal{T}_B(n;d_1,\dots,d_k)$. We want to prove that:

$$\prod_{j=1}^{k} \left|\mathcal{T}(n-k+1;d_j)\right| \le \left|\mathcal{T}_B(n;d_1,\dots,d_k)\right| \le \prod_{j=1}^{k} \left|\mathcal{T}(n;d_j)\right| \tag{5.1}$$

The upper bound is trivial from considering the total number of tuples $(f_1,...,f_k)$ with $f_j \in \mathcal{T}(n;d_j)$. The lower bound is nontrivial except for $k=1$ where both bounds become identical. The key to the proof is the multiplexing (activation attention) procedure, where k input units are viewed as attention units capable of producing a constant mask in the hidden layer, except for the attended function. Here for simplicity we use a sparse encoding in the k components, although dense encoding is also possible, as in the proof of Theorem 4.3. Dense encoding would lead to a reduction in the number of attending units from k to $\lceil \log_2 k \rceil$. As a side note, using more attention units than the minimal number required, can be used to reduce the size of the attention weights, or to make the attention mechanism less sensitive to each individual attention bit.

To prove the lower bound in Composition Theorem 4.2, let us restate it equivalently as:

$$\prod_{j=0}^{k} \left|\mathcal{T}(n-k;d_j)\right| \le \left|\mathcal{T}_B(n;d_0,\dots,d_k)\right| \le \prod_{j=0}^{k} \left|\mathcal{T}(n;d_j)\right|. \tag{5.2}$$

Irreducibility implies that if we select any input component i, the value of B cannot be determined entirely from the value of the remaining components alone. More formally:

Lemma 5.1. *Consider an irreducible Boolean operator $B = B(z_0,\dots,z_k)$ and an index $i \in \{0,\dots,k\}$. There exist signs $\theta \in \{-1,1\}$ and $\theta_j \in \{-1,1\}$, $j \in \{0,\dots,k\} \setminus \{i\}$, such that:*

$$B(z_0,\dots,z_k) = \theta z_i \quad \textit{whenever } z_j = \theta_j \textit{ for all } j \neq i. \tag{5.3}$$

Proof. Consider $B(z_0,\dots,z_k)$ as a function of z_i. If this function is constant in the variable z_i no matter how we fix the other variables, then the value of $B(z_0,\dots,z_k)$ is entirely determined by the values of these other variables, which contradicts irreducibility. Therefore, there exists some assignment $z_j = \theta_j$, $j \neq i$, so that the function $B(\theta_0,\theta_1,\dots,z_i,\dots\theta_k)$ is not constant in z_i. But there exists only two non-constant Boolean functions $f(x)$ in one variable: $f(x) = x$ or $f(x) = -x$, and this determines θ. □

The next lemma essentially states that we can fit an affine function of k variables to $k+1$ points.

Lemma 5.2. *Let $e_0 = 0$ and $e_1,\dots,e_k$ denote the canonical basis vectors in $\mathbb{R}^k$. Then, for any choice of index $j \in \{0,\dots,k\}$ and signs $\theta_i \in \{-1,1\}$, $i \in \{0,\dots,k\} \setminus \{j\}$ there exists an affine function $q : \mathbb{R}^k \to \mathbb{R}$ such that:*

$$q(e_i) = \begin{cases} 0, & i = j \\ \theta_i, & i \neq j \end{cases} \tag{5.4}$$

for all $i \in \{0, \ldots, k\}$.

Proof. It is straightforward to check that the function:

$$q(z) = \theta_0 - \theta_0 z_j + \sum_{i \in \{0,\ldots,k\} \setminus \{j\}} (\theta_i - \theta_0) z_i \tag{5.5}$$

satisfies the required property. ☐

We can now use the previous lemma to derive a lemma for consistently extending a function of $n-k$ variables to a function of n variables. Here k components are used as selector of filter variables, as in the proof of Theorem 4.3.

Lemma 5.3. *Consider a function $f \in \mathcal{T}(n-k; d)$, an index $j \in \{0, \ldots, k\}$, and signs $\theta \in \{-1, 1\}$ and $\theta_i \in \{-1, 1\}$, $i \in \{0, \ldots, k\} \setminus \{j\}$. There exists a function $F \in \mathcal{T}(n; d)$ such that:*

$$F(e_i \oplus x) = \begin{cases} \theta f(x), & i = j \\ \theta_i, & i \neq j \end{cases} \tag{5.6}$$

for all $x \in \{0,1\}^{n-k}$. Here $\oplus$ denotes the concatenation operator.

Proof. Express the polynomial threshold function f as:

$$f(x) = \operatorname{sign}(p(x)) \quad \text{for } x \in \{0,1\}^{n-k} \tag{5.7}$$

where p is a polynomial in n variables and of degree at most d. Let q be a function that satisfies the conclusion of Lemma 5.2. Fix a number M large enough so that $M > |p(x)|$ for all $x \in \{0,1\}^{n-k}$, and define:

$$F(z \oplus x) = \operatorname{sign}\big(Mq(z) + \theta p(x)\big) \tag{5.8}$$

for all $z \in \mathbb{R}^k$ and $x \in \mathbb{R}^{n-k}$. By construction, F is a polynomial threshold function on $\{0,1\}^n$ of degree at most d as required.

Let us check that F satisfies the conclusion of the lemma. If $z = e_j$, we have $q(z) = 0$ due to our choice of q (per the conclusion of Lemma 5.2), and we get $F(z \oplus x) = \operatorname{sign}(\theta p(x)) = \theta f(x)$. If $z = e_i$ with $i \neq j$, then our choice of q implies $F(z \oplus x) = \operatorname{sign}(M\theta_i + \theta p(x))$. The choice of M guarantees that the term $M\theta_i$ dominates the term $\theta p(x)$ in magnitude, so we have $F(s \oplus x) = \operatorname{sign}(M\theta_i) = \theta_i$. ☐

We can now use Lemma 5.3 for the simultaneous extension and filtering of several functions of $n-k$ variables relative to an irreducible Boolean function B.

Lemma 5.4. *For any $(k+1)$-tuple of functions $(f_0, \ldots, f_k)$ where $f_j \in \mathcal{T}(n-k; d_j)$ there exists a $(k+1)$-tuple of functions $(F_0, \ldots, F_k)$ where $F_j \in \mathcal{T}(n; d_j)$ such that:*

$$B(F_0, \ldots, F_k)(e_i \oplus x) = f_i(x) \tag{5.9}$$

for all $i \in \{0, \ldots, k\}$ and $x \in \{0,1\}^{n-k}$.

Proof. Lemma 5.1 yields the existence of signs $\theta_i \in \{-1, 1\}$ for $i \in \{0, \ldots, k\}$ and $\theta_{ij} \in \{-1, 1\}$ for distinct $i, j \in \{0, \ldots, k\}$, such that:

$$B(z_0, \ldots, z_k) = \theta_i z_i \quad \text{whenever } z_j = \theta_{ij} \text{ for all } j \neq i. \tag{5.10}$$

Now consider the functions $f_j \in \mathcal{T}(n-k; d_j)$, $j \in \{0, \ldots, k\}$. Lemma 5.3 yields the existence of functions $F_j \in \mathcal{T}(n; d_j)$, $j \in \{0, \ldots, k\}$, such that:

$$F_j(e_i \oplus x) = \begin{cases} \theta_i f_i(x), & i = j \\ \theta_{ij}, & i \neq j \end{cases} \tag{5.11}$$

for all $i, j \in \{0, \ldots, k\}$ and $x \in \{0,1\}^{n-k}$.

For any fixed $i \in \{0, \ldots, k\}$ and $x \in \{0,1\}^{n-k}$, by construction the variables $z_j := F_j(e_i \oplus x)$ satisfy the condition in (5.10). Therefore, (5.10) and (5.11) yield:

$$B(F_0, \ldots, F_k)(e_i \oplus x) = B(z_0, \ldots, z_k) = \theta_i z_i = \theta_i F_i(e_i \oplus x) = \theta_i^2 f_i(x) = f_i(x) \tag{5.12}$$

as claimed. ☐

Armed with this lemma, we can now prove Theorem 4.2.

Proof of Theorem 4.2. Lemma 5.4 demonstrates that for any tuple of functions $(f_0, \ldots, f_k) \in \prod_{i=0}^{k} \mathcal{T}(n-k; d_j)$ there exists a function $F \in \mathcal{T}_B(n; d_0, \ldots, d_k)$ such that $F(e_i \oplus x) = f_i(x)$ for all $i \in \{0, \ldots, k\}$ and $x \in \{0,1\}^{n-k}$. Thus, each component f_i of the original k-tuple can be uniquely recovered from F. Therefore, a map $(f_0, \ldots, f_k) \mapsto F$ (if there are multiple F corresponding to some f, select one arbitrarily) defines an injection from the cartesian product $\prod_{i=0}^{k} \mathcal{T}(n-k; d_j)$ into $\mathcal{T}_B(n; d_0, \ldots, d_k)$, completing the proof. ☐

As shown in Table **??**, there are 16 binary Boolean operators B. Ten of them are irreducible, including AND, OR and XOR and their negations. For each such operator, the Composition Theorem 4.2 gives:

$$\left|\mathcal{T}(n-1; d_0)\right| \left|\mathcal{T}(n-1; d_1)\right| \leq \left|\mathcal{T}_B(n; d_0, d_1)\right| \leq \left|\mathcal{T}(n; d_0)\right| \left|\mathcal{T}(n; d_1)\right| \tag{5.13}$$

Surprisingly, the intersection of all ten classes is still as large.

Proposition 5.5. *We have:*

$$\left|\bigcap_B \mathcal{T}_B(n; d_0, d_1)\right| \geq \left|\mathcal{T}(n-1; d_0)\right| \left|\mathcal{T}(n-1; d_1)\right| \tag{5.14}$$

where the intersection is over the ten irreducible binary Boolean operators.

In particular, there are many functions f (specifically, $2^{2n^2(1-o(1))}$) that can be simultaneously expressed as: $f = f_1$ AND $f_2 = f_3$ OR $f_4 = f_5\, XOR\, f_6$ where all the f_i are linear threshold gates.

Proof. In the proof of the Composition Theorem 4.2 above, we showed that for each irreducible Boolean operator B and pair of functions $(f_0, f_1) \in \mathcal{T}(n-1; d_0) \times \mathcal{T}(n-1; d_1)$, there exists $F \in \mathcal{T}_B(n; d_0, d_1)$ such that:

$$F(0 \oplus x) = f_0(x), \quad F(1 \oplus x) = f_1(x) \tag{5.15}$$

for all $x \in \{0,1\}^{n-1}$. Obviously, this pair of equations defines F uniquely on $\{0,1\}$, and F is independent of B. Thus, F lies in the intersection of $\mathcal{T}_B(n; d_0, d_1)$ over all irreducible B. ☐

References

[1] Martin Anthony. *Discrete mathematics of neural networks: selected topics*, volume 8. Siam, 2001.

[2] Amy F.T. Arnsten and Francisco X. Castellanos. Neurobiology of attention regulation and its disorders. *Pediatric Psychopharmacology*, page 95, 2010.

[3] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[4] P. Baldi. Neural networks, orientations of the hypercube and algebraic threshold functions. *IEEE Transactions on Information Theory*, 34(3):523–530, 1988.

[5] P. Baldi. *Deep Learning in Science*. Cambridge University Press, Cambridge, UK, 2021.

[6] Pierre Baldi and Roman Vershynin. The capacity of feedforward neural networks. *Neural Networks*, 116:288–311, 2019. Also: arXiv preprint arXiv:1901.00434.

[7] Pierre Baldi and Roman Vershynin. Polynomial threshold functions, hyperplane arrangements, and random tensors. *SIAM Journal on Mathematics of Data Science*, 1(4):699–729, 2019. Also: arXiv preprint arXiv:1803.10868.

[8] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*, 2021.

[11] M. Fenton, A. Shmakov, T. Ho, S. Hsu, D. Whiteson, and P. Baldi. Permutationless many-jet event reconstruction with symmetry preserving attention networks. *Physical Review D*, 2020. In press. Also arXiv:2010.09206.

[12] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[13] Laurent Itti, Geraint Rees, and John K Tsotsos. *Neurobiology of attention*. Elsevier, 2005.

[14] Jeff Kahn, János Komlós, and Endre Szemerédi. On the probability that a random$\pm$1-matrix is singular. *Journal of the American Mathematical Society*, 8(1):223–240, 1995.

[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[16] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021.

[17] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[18] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

[19] Michael I Posner. *Cognitive neuroscience of attention*. Guilford Press, 2011.

[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[21] Mohammadamin Tavakoli, Forest Agostinelli, and Pierre Baldi. Splash: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks*, 140:1–12, 2021.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[24] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[25] David Zipser and Richard A Andersen. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331(6158):679–684, 1988.

[26] Yu A Zuev. Asymptotics of the logarithm of the number of threshold functions of the algebra of logic. *Soviet Mathematics Doklady*, 39(3):512–513, 1989.

[27] Yu A Zuev. Combinatorial-probability and geometric methods in threshold logic. *Diskretnaya Matematika*, 3(2):47–57, 1991.