Inductive Biases and Variable Creation in Self-Attention Mechanisms

Scribes: Manuel Paez, Desmond DeVaul

April 7th 2025

1 Introduction

The **self-attention mechanism**, defined in [LPM15] for generative-sequencing modeling and used in SOTA natural language models starting from [VSP+17], is an architecture that directly models long-range interactions within the input context as part of its inductive or learning bias. This is unlike convolutional and recurrent models, which have an inductive bias towards locality and positional (translational) invariance. With this, the self-attention mechanism raised up a fundamental question about its inductive biases - the set of assumptions that the learner uses to predict outputs of given inputs that have not been encountered: ([EGKZ22]) *which functions do self-attention networks prefer to represent (What type of functions does the self-attention mechanism like to approximate)?*

This work initiates a theoretical analysis to answer the question of self-attention and its inductive biases and how it handles long-range dependencies. This builds off of previous work for norm-based capacity bounds for neural networks.

Norm-based capacity bounds for neural nets: Works such as [BM02, CLZ19] have established norm-based generalization bounds for various neural network architectures using cover-based arguments. This was further improved on [JNM+19] which showed how these bounds predict generalization in several of these architectures. This work complements these results by establishing the first norm-based capacity analysis for attention models and uses a covering number bound for linear function classes given by [Zha02].

The paper "Inductive Biases and Variable Creation in Self-Attention Mechanisms" [EGKZ22] shows that a bounded-norm self-attention head learning bias is towards a sparse function of a length-T context with sample complexity $\log(T)$, which they defined as the **sparse-variable creation**. This essentially means that attention mechanism inductive bias is towards functions dependent only on a small subset of input positions. The authors of the paper show these three main results:

1. **Covering number-based capacity bound for attention mechanism**: The paper shows that for bounded-norm attention heads, sample complexity to learn sparse functions is $O(\log(T))$.

2. **Representational results for attention heads**: The paper shows that bounded-norm attention heads or Transformers with bounded-norms can represent s-sparse functions with weights with norms $2^{O(s)}$ ($poly(s)$ for symmetric functions). This gives theoretical insight into why attention models can learn global-range dependencies without overfitting.

3. **Synthetic experiments**: The paper shows experiments confirming the theoretical bounds for learning parity functions with self-attention. In the paper, they train Transformer models to identify sparse Boolean functions with randomly chosen indices and corroborate the sample complexity scaling law predicted by the theory (i.e. $\log(T)$). They also show that via i.i.d samples of their experiment, Transformers can successfully learn XOR parity functions.

2 Preliminaries

The input $X := [x_1 x_2 \dots x_T]^\top \in \mathbb{R}^{T \times d}$ to an attention module will be a length-T sequence of tokens (embeddings) $x_t \in \mathbb{R}^d$; m refers to the sample size (i.e. the number of length-T sequences in a dataset). We will denote $z \in R^d$ to be the context for an attention-mechanism, and $z = x_t$ in self-attention. $\|\cdot\|_2$ denotes the spectral norm for matrices, and $\|\cdot\|_{p,q}$ denotes the (p,q)-matrix norm where the p-norm is over columns and q-norm over rows. For vectors, $\|\cdot\|_p$ denotes the ℓ_p norm; we drop the subscript for the ℓ_2 norm. B is generally used to quantify bounds on norms of matrices and L for Lipschitz constants.

2.1 Understanding Covering Number

In this work, the authors use the covering number to count the number of functions represented by a self-attention mechanism and a transformer. Taking the log-covering number bounds implies a generalization bound for the self-attention mechanism and transformer.

Covering numbers are important in statistical learning theory because it helps us measure the complexity of a class. In this case, we will consider function classes. Intuitively, the covering number tells us how many elements in a subset of our class do we need to approximate the entire class. If we need more elements to approximate this class, then we have a more complex class.

Before we talk about covering numbers of function classes, let's warm up with covering numbers of objects in pseudoometric spaces.

2.1.1 Covering Number Warm up

A psuedometric space is a set, X with a notion of distance $d(x,y)$, i.e. the metric, for all elements $x, y \in X$. In psuedometric space, we only require that the metric have the following two properties: $d(x,y) = d(y,x)$ and $d(x,y) \leq d(x,z) + d(z,y)$. Now let us consider a subset $T \in X$. The ϵ-covering number of T is the smallest possible subset $\widehat{T} \in T$ such that $\forall t \in T \ \exists \widehat{t} \in \widehat{T}$ such that $d(t,\widehat{t}) \leq \epsilon$. Let $N(T,\epsilon,d)$ refer to the ϵ-covering number of a subset T with metric d. Thus: $N(T,\epsilon,d) = min(|\widehat{T}| : \widehat{T})$ is an ϵ-cover of T

2.1.2 Intuitive Example

For an easy visualization, consider the vector space $\mathbb{R}^2$ and a set of points $T = \{(0.5, 0.7), (0.2, 3), (0.6, 2.5)\}$. Further, $d(x,y) = ||x - y||_2$. Let us also set $\epsilon = 0.05$. In line with the notation of this paper, let $N_2(T, \epsilon, ||\cdot||_2)$ refer to the covering number of the set T using the L_2 norm as our metric. In this case, we can imagine ϵ balls (or circles, in 2-dimensions) around each element of T. We can visualize this set up in 1

In this case, we have 3 points that represent each element in T. Further, we have ϵ balls around them. In this visualization, we need a set of elements $\widehat{T}$ within T such that we have at least 1

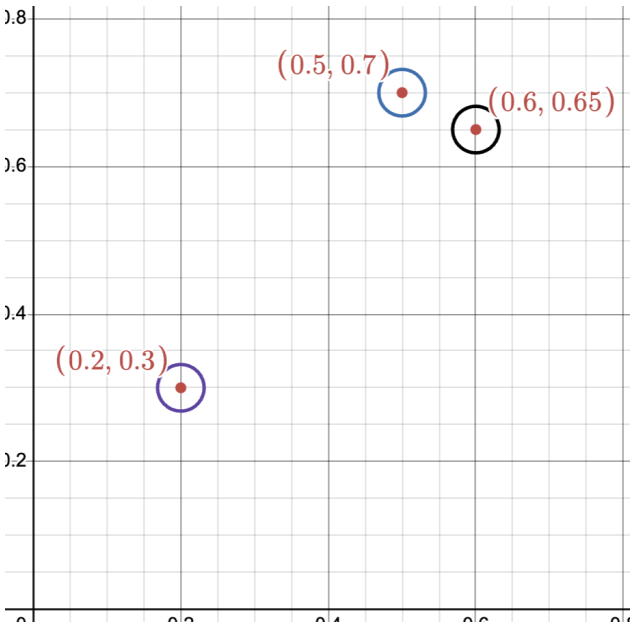


Figure 1: Set of points T with epsilon balls ($\epsilon = 0.05$) surrounding them

element in each of these ϵ balls, thus satisfying that $\forall t \in T \; \exists \widehat{t} \in \widehat{T}$ such that $d(t, \widehat{t}) \leq \epsilon$, where in this case $d(t, \widehat{t}) = ||t - \widehat{t}||_2$. Because the epsilon balls are relatively tight, no point in T will cover any other point except itself; thus the *only* subset (and therefore the minimum sized subset) $\widehat{T} \in T$ such that is actually just T itself. So $\widehat{T}$ is our minimum sized ϵ covering of T. Thus, $N_2(0.05, T, ||\cdot||_2) = |\widehat{T}| = 3$.

However, what if we change ϵ to be larger? If we make $\epsilon = 0.5$, as in 2, we can see here that the set $\widehat{T} = \{(0.5, 0.7), (0.2, 0.3)\}$ successfully covers the set T because $||(0.5, 0.7) - (0.6, 0.65)||_2 \leq \epsilon$ (and obviously $||(0.5, 0.7) - (0.5, 0.7)||_2 = 0 \leq \epsilon$ and $||(0.2, 0.3) - (0.2, 0.3)||_2 = 0 \leq \epsilon$). We can tell this visually because $(0.5, 0.7)$ is in the ϵ-ball around $(0.6, 0.65)$. Thus, $N_2(0.5, T, ||\cdot||_2) = |\widehat{T}| = 2$. Note that $\{(0.5, 0.7), (0.2, 3), (0.6, 2.5)\}$ and $\{(0.2, 3), (0.6, 2.5)\}$ also provide coverings, but the minimum size covering, and thus the covering number is still 2. As we see in this second example, we have a point that can cover itself *and* another point!

From these visualizations hopefully we have developed the following intuitions: Covering number is roughly measuring the "complexity" of a set (we will define this more formally later as we relate covering number to VC-dim), and $N(T, \epsilon, d) \leq N(T, \epsilon/2, d)$. The second point of intuition, described informally, is that as ϵ increases, the covering number can only increase. Also, we should have the intuition that as the space which the set spans increases (i.e. as the distance between the members of the set increases) then the covering number will also increase. Thus, covering number also encapsulates some notion of "closeness" of the set. If the set is more spread out, then the covering number will increase. Now let us shift our attention to the idea of a covering number of a function class. This is their definition as it appears in the paper:

Definition 2.1. (Covering number). For a given class of vector-valued functions $\mathcal{F}$, the covering number $\mathcal{N}_\infty(\mathcal{F}; \epsilon; \{z^{(i)}\}_{i=1}^m; \|\cdot\|)$ is the smallest size of a collection (a cover) $\mathcal{C} \subset \mathcal{F}$ such that $\forall f \in \mathcal{F}, \exists \widehat{f} \in \mathcal{C}$ satisfying

$$\max_i \|f(z^{(i)}) - \widehat{f}(z^{(i)})\| \leq \epsilon.$$

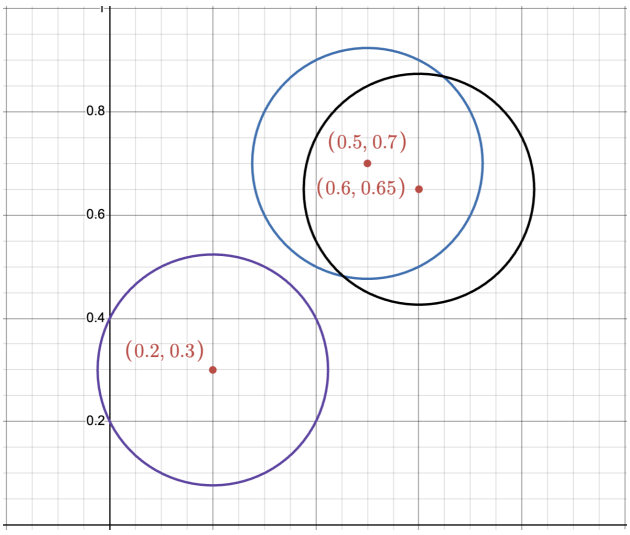


Figure 2: Set of points T with epsilon balls ($\epsilon = 0.5$) surrounding them

Further, define

$$\mathcal{N}_\infty(\mathcal{F}, \epsilon, m, \|\cdot\|) = \sup_{z^{(1)},\ldots,z^{(m)}} \mathcal{N}_\infty(\mathcal{F}; \epsilon; z^{(1)}, \ldots, z^{(m)}, \|\cdot\|).$$

In this definition of covering number, we are presented with a function class and sample points for evaluation. Our task is to identify a minimal subset of functions from this class (the "cover") such that for any function in the original class, we can find a function in our subset that closely approximates it across all sample points–with "closely" meaning that the maximum deviation in the infinity norm at any sample point doesn't exceed our specified error threshold ϵ. The other important point is that we assume must use the samples which maximize the covering number. Basically, we are considering the worst case samples.

Essentially by creating a cover, we're creating a representative collection that can stand in for the entire function class while maintaining approximation guarantees at every sample point. Before, when we were talking about covering numbers of just sets of objects in psuedometric space, we did not consider samples. Now we have to consider samples. We have to cover all of these functions evaluated at all of these samples. However, the same general principle of finding a subset such that we can approximate all the points within some epsilon tolerance remains the same. We can still use the same visualization where we imagine every $f(z^{(i)})\ \forall f \in \mathcal{F}\ \forall i$, as being some point, and we want to figure out how to approximate all these points within some epsilon approximation using some subset of $\mathcal{F}$.

As before when we were talking about covering numbers of objects in psuedometric space, the covering number is related to how "spread out" the set is. The same intuition applies here. If we bound the "spreadoutness" of the function class, then we can also bound the covering number. The authors of this paper do this in their assumptions section when they bound the Lipschitzness, the range, and the Jacobian of various function classes (see their "Assumpsions" section). Together, these constraints effectively "tame" the function class by restricting how rapidly functions can vary and how far their outputs can range. The more constrained these properties are, the fewer representative functions are needed in a cover—directly translating to a smaller covering number.

This mathematical formalization captures our intuition that "less spread out" function classes are intrinsically less complex and thus easier to approximate.

In this paper they also frequently talk about parameterized functions. Let's consider the function covering number applied to parameterized function classes like so:

Definition 2.2. (Covering number for parameterized functions). For a given class of vector-valued parameterized functions $\mathcal{F} = \{f(\cdot;\theta) : \theta \in \Theta\}$, the covering number $\mathcal{N}_\infty(\mathcal{F};\epsilon;\{z^{(i)}\}_{i=1}^m;\|\cdot\|)$ is the smallest size of a collection of parameters (a cover) $\widehat{\Theta} \subset \Theta$ such that $\forall\theta \in \Theta, \exists\widehat{\theta} \in \widehat{\Theta}$ satisfying

$$\max_i \|f(z^{(i)};\theta) - f(z^{(i)};\widehat{\theta})\| \leq \epsilon.$$

Further, define

$$\mathcal{N}_\infty(\mathcal{F},\epsilon,m,\|\cdot\|) = \sup_{z^{(1)},\ldots,z^{(m)}} \mathcal{N}_\infty(\mathcal{F};\epsilon;z^{(1)},\ldots,z^{(m)},\|\cdot\|).$$

Particularly in the case of linear functions, as they discuss in this paper, Θ is just some collection of possible vectors or matrices. Thus, our goal in constructing a covering of such a parameterized class is to find the smallest subset of possible parameters such that we can approximate our function at every sample with all possible parameters Θ. Considering the subset of parameters is perhaps easier to conceptualize.

2.2 Covering Number for Learning Theory

The covering number is important to this class and learning theory more generally because it, like VC dimension and Rademacher complexity, provides a fundamental measure of hypothesis class complexity that directly bounds generalization error. For instance, look at this generalization error bound with covering number that they give in the paper:

Lemma 2.3 (Generalization bound via covering number; informal)**.** *Suppose $\mathcal{F}$ is a class of bounded functions, and $\log\mathcal{N}_\infty(\mathcal{F};\epsilon;x^{(1)},\ldots,x^{(m)}) \leq C_\mathcal{F}/\epsilon^2$ for all $x^{(1)},\ldots,x^{(m)} \in \mathcal{X}^m$. Then for any $\delta > 0$, with probability at least $1-\delta$, simultaneously for all $f \in \mathcal{F}$, the generalization error ϵ_{gen} satisfies*

$$\epsilon_{\text{gen}}(f) \leq \widetilde{O}\left(\sqrt{\frac{C_\mathcal{F}}{m}} + \sqrt{\frac{\log(1/\delta)}{m}}\right).$$

This equation lets us bound the generalization error of a class of functions by the covering number.

We can also relate the covering number to several other measures of complexity.

2.2.1 Covering Number and VC Dim $\mathcal{F}$ from [Hau14]

Let $\mathcal{F} = \{f : \mathcal{X} \to \{0,1\}\}$ be a binary function class. For any $\epsilon \in [0,1)$:

$$\mathcal{N}_\infty(\mathcal{F},\epsilon,m) \leq \left(\frac{e\,m}{\text{VC}(\mathcal{F})}\right)^{\text{VC}(\mathcal{F})}$$

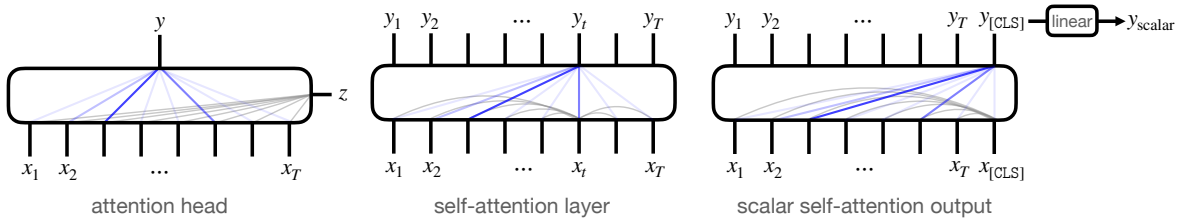


Figure 3: Diagrams of attention modules $f_{\text{tfHead}}, f_{\text{ffLayer}}, f_{\text{tfScalar}}$: alignment scores (grey edges) determine normalized attention weights (blue), which are used to mix the inputs $x_{1:T}$. *Left:* Attention with a general context z. *Center:* Self-attention layer, where both the input and the context come from $x_{1:T}$. *Right:* CLS token to extract a single scalar from a self-attention layer, providing a real-valued function class for classification or regression tasks.

where m is the number of samples, $\text{VC}(\mathcal{F})$ is the VC dimension of the function class $\mathcal{F}$ and e is Euler's constant.

2.2.2 Covering Number and Fat Shattering Dimension, from [Tew23]

Let $\mathcal{F} \subseteq [0,1]^{\mathcal{X}}$ and $\epsilon \in [0,1]$. Suppose $d = \text{fat}_{\epsilon/4}(\mathcal{F})$ where $\text{fat}_{\epsilon/4}(\mathcal{F})$ is the largest set of points which can be γ-shattered by $\mathcal{F}$, where $\gamma = \epsilon/4$.

$$\mathcal{N}_\infty(\mathcal{F}, \epsilon, m) < 2\left(m\left(\frac{2}{\epsilon}+1\right)^2\right)^{\lceil d \log\left(\frac{2em}{d\epsilon}\right)\rceil}$$

where m is the number of samples.

2.2.3 Covering Number and Rademacher complexity

Let $\mathcal{F}$ be a class of real-valued functions such that $|f| \leq A$ for all $f \in \mathcal{F}$. Suppose $\log \mathcal{N}_\infty(\mathcal{F}; \epsilon; z^{(1)}, \ldots, z^{(m)}) \leq C_{\mathcal{F}}/\epsilon^2$, then

$$\widehat{\mathcal{R}}(\mathcal{F}; z^{(1)}, \ldots, z^{(m)}) \leq c \cdot \sqrt{\frac{C_{\mathcal{F}}}{m}} \cdot \left(1 + \log\left(A\sqrt{m/C_{\mathcal{F}}}\right)\right)$$

where m is the number of samples.

Hopefully, we now have an intuition for how the covering number works, an intuition for how it measures complexity, and we have seen (albeit very briefly) that it can be related to other complexity measures which we have seen previously in this class. Let us now turn our *attention* to the next section, attention, which we will then connect with covering numbers in section 3.

2.3 Attention

The idea of attention is to capture the aspect that that an output variable selects a part of the input sequence on which it will depend, based on a learned function of global interactions.

Definition 2.4 (Attention head)**.** An attention head is a function $f : \mathcal{X} \to \mathcal{Y}$ that maps a input length-T sequence $X \in \mathcal{X}^T$ and an additional context $z \in \mathcal{Z}$ with:

- Alignment score function $Score : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$: $Score(x_t, z; \theta_s) \in R$

- Normalization function: $Norm : \mathbb{R}^T \to \Delta^{T-1}$
- Input map (Position-wise map) $\phi_{\text{in}} : \mathcal{X} \to \mathcal{V} : \phi_{\text{in}}(x; \theta_{in})$
- Output map (Position-wise map) $\phi_{\text{out}} : \mathcal{V} \to \mathcal{Y}$: $\phi_{\text{out}}(\cdot; \theta_{out})$

to the following output:

$$y = \phi_{\text{out}}\Big(\sum_{t=1}^{T}\Big[\mathsf{Norm}\Big(\mathsf{Score}(x_1, z; \theta_s), \ldots, \mathsf{Score}(x_T, z; \theta_s)\Big)\Big]_t \phi_{\text{in}}(x_t; \theta_{\text{in}});\ \theta_{\text{out}}\Big)$$
$$= \phi_{\text{out}}\Big(\phi_{\text{in}}(X; \theta_{\text{in}})^\top \mathsf{Norm}\Big(\mathsf{Score}(x_1, z; \theta_s), \ldots, \mathsf{Score}(x_T, z; \theta_s)\Big);\ \theta_{\text{out}}\Big)$$

where $\phi_{\text{in}}(X; \theta) = [\phi_{\text{in}}(x_1; \theta) \ldots \phi_{\text{in}}(x_T; \theta)]^\top$ denotes the row-wise application of ϕ_{in} and $\mathcal{V}$ being a vector space of input representations mixed by the normalized alignment scores.

Let's try to understand this intuitively. The attention head starts by computing a score function which compares how relevant each input token is to the context. A high score function means that it is very relevant. After normalizing the scores, we multiply the score for each token by the result of the input value, the input might be thought of as the value which the token represents. Finally, we add all of these together and map this intermediate space into the output which we desire. Essentially, we take the latent representations of each tokens and add them together depending on how relevant they are to our context–the more relevant, the more we consider their representation– and then we take this combined value and map it back out into whatever space that we want. One can think of this as selecting the most relevant inputs, finding their meaning, adding them together, and then computing some output. Considering self-attention, we simply replace the context vector z with a particular input token. We can thus write self-attention as

$$y = \phi_{\text{out}}\left(\phi_{\text{in}}(X; \theta_{\text{in}})^\top \text{Norm}\left(\text{Score}(X, x_t; \theta_s)\right); \theta_{\text{out}}\right). \tag{1}$$

With self-attention, we are not going to compute one y for the entire input sequence. Instead, we will compute a y_t where we use each individual input token as the "context" vector in the above scenario. Thus, we would get a T-length output, usually, from self attention, because we would apply this function to every input token. This explains the difference between the two first diagrams in 3. Let us now define the Transformer self-attention architecture:

Definition 2.5 (Transformer layer)**.** A Transformer layer is a collection of T attention heads (whose outputs are $y_1, \ldots, y_T$) with the following shared parameters:

- The context for head τ is x_τ, and the score function is:

$$\mathsf{Score}(x, x_\tau; \{W_Q, W_K\}) := x_\tau^\top W_Q W_K^\top x, \quad W_Q, W_K \in \mathbb{R}^{d \times k}.$$

- Input projection ϕ_{in} is linear:

$$\phi_{\text{in}}(x; W_V) := W_V^\top x, \quad W_V \in \mathbb{R}^{d \times k}.$$

- Output projection ϕ_{out} is a linear function, composed with an L_σ-Lipschitz activation function

$\sigma : \mathbb{R} \to \mathbb{R}$ such that $\sigma(0) = 0$:

$$\phi_{\text{out}}(x; W_C) := W_C^\top \sigma(x), \quad W_C \in \mathbb{R}^{k \times d}.$$

- Normalization function is the T-dimensional softmax:

$$\mathsf{Norm}(x) := \mathsf{softmax}\,(x) = \frac{\exp(x)}{\mathbf{1}^\top \exp(x)}.$$

Therefore, the self-attention computation for $Y := [y_1 y_2 \dots y_T]^\top \in \mathbb{R}^{T \times d}$ with $[\mathsf{RowSoftmax}(M)]_{t,:} := \mathsf{softmax}(M_{t,:})$ is

$$Y = \sigma\left(\mathsf{RowSoftmax}\left(XW_Q(XW_K)^\top\right) XW_V\right) W_C.$$

For Scalar output transformers, once adds a special token x_{CLS} and outputs $y = w^T y_{\text{CLS}}$ where $w \in R^d$.

The important thing to note about this section is that transformer attention is just a specific instance of the generalized attention which we were just considering. Instead of having abstract functions like Score, we replace them with slightly less abstract function classes defined with some matrix operations.

3 Capacity bounds for attention modules

In this section, we will attempt to gain a deeper intuition of Theorem 4.2 and Corollary 4.5, which are two of the most important conclusions in the paper.

3.1 Theorem 4.2: Attention Head Capacity

This is arguably the most brilliant part of this paper. The general overview is that we have a function class, $\mathcal{F}_{head}$ which we are going to break down into its constituent parts in order to analyze the covering number bounds.

Recall that the attention head architecture can be represented as a function $f_{\text{head}} : \mathbb{R}^{T \times d} \times \mathbb{R}^d \to \mathbb{R}^d$ parameterized by $\theta_s, \theta_{\text{in}}, \theta_{\text{out}}$ as

$$f_{\text{head}}(X, z; \theta_s, \theta_{\text{in}}, \theta_{\text{out}}) = \phi_{\text{out}}\left(\phi_{\text{in}}(X; \theta_{\text{in}})^\top \text{Norm}(\text{Score}(X, z; \theta_s)); \theta_{\text{out}}\right). \tag{2}$$

Denote the corresponding function class by

$$\mathcal{F}_{\text{head}} := \{(X, z) \mapsto f_{\text{head}}(X, z; \theta_s, \theta_{\text{in}}, \theta_{\text{out}}) : \theta_s \in \Theta_s, \theta_{\text{in}} \in \Theta_{\text{in}}, \theta_{\text{out}} \in \Theta_{\text{out}}\} \tag{3}$$

The first thing that the authors do is assume some properties of these function classes. Remember from when we were talking about covering number of classes that smoothness and bounds can "tame" the functions that are being represented to make the covering numbers managable. This is exactly what the authors do here with there assumptions:

Assumption 3.1. We make the following assumptions:

1. ϕ_{out} is L_{out}-Lipschitz in the ℓ_2-norm, that is,

$$\forall a, b \in \mathbb{R}^k : \|\phi_{\text{out}}(a) - \phi_{\text{out}}(b)\| \leq L_{\text{out}}\|a - b\|. \tag{4}$$

2. ϕ_{in} is B_{in}-bounded in ℓ_2-norm, that is,

$$\forall a \in \mathbb{R}^d, \theta_{\text{in}} \in \Theta_{\text{in}} : \|\phi_{\text{in}}(a; \theta_{\text{in}})\| \leq B_{\text{in}}\|a\|. \tag{5}$$

3. Norm is continuously differentiable and its Jacobian satisfies

$$\forall \theta \in \mathbb{R}^T, \|J\text{Norm}(\theta)\|_{1,1} \leq C_{\text{Norm}}. \tag{6}$$

Now that we have certain assumptions about these functions, we can consider that every head function is basically a composition of the score function class, the in function class, and the out function class. So, the authors begin by showing in Lemma 4.3 the "Lipschitzness" of the head function class. In other words, the "smoothness" (and thus how "tame" this function class is) is bounded by the score, in and out function classes.

Lemma 3.2 (ℓ_∞-Lipschitzness of f_{head})**.** *For any $\theta_s, \widehat{\theta}_s \in \Theta_s, \theta_{in}, \widehat{\theta}_{in} \in \Theta_{in}$; for all $X \in \mathbb{R}^{T \times d}$, such that $\|X^\top\|_{2,\infty} \leq B_X$,*

$$\|f_{head}(X, z; \theta_s, \theta_{in}, w) - f_{head}(X, z; \widehat{\theta}_s, \widehat{\theta}_{in}, w)\| \leq$$
$$C_{Norm} L_{out} B_{in} B_X \|Score(X, z; \theta_s) - Score(X, z; \widehat{\theta}_s)\|_\infty + L_{out}\|\phi_{in}(X; \theta_{in}) - \phi_{in}(X; \widehat{\theta}_{in})\|_{2,\infty}.$$

In fact, if we remember earlier from talking about the covering number of parameterized function classes, one way of thinking about constructing a covering is by picking a subset of the parameters that would allow the entire function class would be covered at every sample. What Lemma 4.3 shows is that for an arbitrary $\widehat{\theta}_{in}$ and $\widehat{\theta}_s$, we can bound the difference between the head function using these "approximate" values and some other values by the difference between the score function with the "approximate" parameters and the "in" function at these "approximate" parameters, multiplied by some nice constants that we derived from our assumptions.

The intuition from this lemma should be that, maybe, if we can select a subset of parameters $\widehat{\Theta}_{in}$ and $\widehat{\Theta}_s$ such that for all $\theta_{in} \in \Theta_{in}$ $\theta_s \in \Theta_s$ we can arbitrarily approximate the score and in functions for every $\theta_{in} \in \Theta_{in}$ and for every $\theta_s \in \Theta_s$, by creating covering of these individual functions, that we can then use the combination of these covers of the score and input functions to act as a covering of the head function.

This is what they prove in Theorem 4.2. Namely, that we can construct a covering of the head function, using the coverings of the in and score functions. Formally:

$$\begin{aligned}\log|\mathcal{C}_{\text{head}}| &= \log|\mathcal{C}_{\text{Score}}| + \log|\mathcal{C}_{\text{in}}| \\ &= \log\mathcal{N}_\infty\left(\mathcal{F}_{\text{Score}}; \epsilon_{\text{Score}}; \{(x_t^{(i)}, z^{(i)})\}_{i\in[m],t\in[T]}\right) + \log\mathcal{N}_\infty\left(\mathcal{F}_{\text{in}}; \epsilon_{\text{in}}; \{x_t^{(i)}\}_{i\in[m],t\in[T]}; \|\cdot\|_2\right)\end{aligned}$$

There is another important part of Theorem 4.2, namely, they show why the infinity norm is useful in this paper for allowing us to break up the input sequences. Namely, they point out that:

For all $\theta_s, \widehat{\theta}_s$

$$\max_{i\in[m]} \|\text{Score}(X^{(i)}, z^{(i)}; \theta_s) - \text{Score}(X^{(i)}, z^{(i)}; \widehat{\theta}_s)\|_\infty = \max_{i\in[m],t\in[T]} |\text{Score}(x_t^{(i)}, z^{(i)}; \theta_s) - \text{Score}(x_t^{(i)}, z^{(i)}; \widehat{\theta}_s)|.$$

Similarly, for all $\theta_{\text{in}}, \widehat{\theta}_{\text{in}}$,

$$\max_{i\in[m]} \left\| \left(\phi_{\text{in}}(X^{(i)}; \theta_{\text{in}}) - \phi_{\text{in}}(X^{(i)}; \widehat{\theta}_{\text{in}}) \right)^\top \right\|_{2,\infty} = \max_{i\in[m],t\in[T]} \|\phi_{\text{in}}(x_t^{(i)}; \theta_{\text{in}}) - \phi_{\text{in}}(x_t^{(i)}; \widehat{\theta}_{\text{in}})\|.$$

This crucially allows us to aggregate over the i and t dimensions together. Therefore, we can consider $\mathcal{N}_\infty$ covers for the above to bound the overall covering number.

In other words, because we are using infinity norms, we needn't compare the entire sequences, but can consider the maximum distance at a particular input sample and input token. This is important particularly for removing a linear dependence on T.

3.2 Capacity bounds for Single-Layer Transformers

Another very important conclusion from this paper is that capacity of attention in transformers can be bounded by the log if the input sequence. This is done largely using work from [Zha02]. In that paper, the author proves that

For the class of linear functions,

$$\mathcal{F}_{\text{lin}} = \{x \mapsto w \cdot x : w \in \mathbb{R}^d, \|w\|_2 \leq B_W\}, \tag{7}$$

there is the covering bound of

$$\mathcal{N}_\infty(\mathcal{F}_{lin}; \epsilon; \{x^{(i)}\}_{i=1}^m) \leq O\left(\frac{B_X^2 B_W^2}{\epsilon^2} \cdot \log\left(\frac{B_X B_W m}{\epsilon} \right) \right), \tag{8}$$

In other words, [Zha02] proved that for a linear function class, the covering number is bounded by the number of samples.

From earlier, we hopefully developed an intuition that covering number was almost a measure of "spreadoutness." Because of this, we might naturally assume, then, that "spreadoutness" should be bounded by the dimension, not the number of samples. In fact, this is what the original proof in [Zha02] proved. However, if we have fewer samples than dimensions, then really we can consider the "effective" dimension of the number of samples. In other words, if we have only m samples, then they only span some m-dimensional subspace. Thus, if we have fewer samples than number of dimensions, using the nubmer of samples actually provides a tighter bound.

This paper then essentially extends the above result into higher dimensional parameters. In other words, the aforementioned bound is for an operation between vectors. In this paper, they consider that w is not a vector, but a matrix.

They then prove that the bounds for a linear function class with a matrix parameterization instead of a vector parameterization is still bounded logarithmically by the number of samples:

Lemma 3.3 (4.6)**.** *Let* $\mathcal{W} : \{W \in \mathbb{R}^{d_1 \times d_2} : \|W^\top\|_{2,1} \leq B_W\}$*, and consider the function class*

$\mathcal{F} : \{x \mapsto Wx : W \in \mathcal{W}\}$. *For any* $\epsilon > 0$ *and* $x^{(1)}, \ldots, x^{(N)} \in \mathbb{R}^{d_1}$ *satisfying* $\forall i \in [N], \|x^{(i)}\| \leq B_X$,

$$\log \mathcal{N}_\infty(\mathcal{F}; \epsilon; x^{(1)}, \ldots, x^{(N)}; \|\cdot\|_2) \lesssim \frac{(B_X B_W)^2}{\epsilon^2} \log(d_1 N). \tag{9}$$

This lemma is important for generalization bounds for transofmrers, because the Score and In functions for transformers are of this same form. So, the basic intuition here is that with generalized attention, we can bound the covering number of the head function by the sum of the covering number of the score and in functions, essentially:

$$\begin{aligned}\log |\mathcal{C}_{\text{tf-head}}| &= \log |\mathcal{C}_{\text{tf-Score}}| + \log |\mathcal{C}_{\text{tf-in}}| \\ &= \log \mathcal{N}_\infty \left(\mathcal{F}_{\text{tf-Score}}; \epsilon_{\text{tf-Score}}; \{(x_t^{(i)}, z^{(i)})\}_{i\in[m],t\in[T]}\right) + \log \mathcal{N}_\infty \left(\mathcal{F}_{\text{tf-in}}; \epsilon_{\text{in}}; \{x_t^{(i)}\}_{i\in[m],t\in[T]}; \|\cdot\|_2\right)\end{aligned}$$

Well, if we now have this lemma that shows:

$$\log \mathcal{N}_\infty(\mathcal{F}_{tf-score}; \epsilon; x^{(1)}, \ldots, x^{(N)}; \|\cdot\|_2) \lesssim \frac{(B_X B_W)^2}{\epsilon^2} \log(d_1 N). \tag{10}$$

and

$$\log \mathcal{N}_\infty(\mathcal{F}_{tf-in}; \epsilon; x^{(1)}, \ldots, x^{(N)}; \|\cdot\|_2) \lesssim \frac{(B_X B_W)^2}{\epsilon^2} \log(d_1 N). \tag{11}$$

then we can combine these are basically rewrite the above to be:

$$\begin{aligned}\log |\mathcal{C}_{\text{tf-head}}| &= \log |\mathcal{C}_{\text{tf-Score}}| + \log |\mathcal{C}_{\text{in}}| \\ &= \log \mathcal{N}_\infty \left(\mathcal{F}_{\text{tf-Score}}; \epsilon_{\text{tf-Score}}; \{(x_t^{(i)}, z^{(i)})\}_{i\in[m],t\in[T]}\right) + \log \mathcal{N}_\infty \left(\mathcal{F}_{\text{tf-in}}; \epsilon_{\text{tf-in}}; \{x_t^{(i)}\}_{i\in[m],t\in[T]}; \|\cdot\|_2\right) \\ &\lesssim \frac{(B_X B_W)^2}{\epsilon^2} \log(d_1 mT) + \frac{(B_X B_W)^2}{\epsilon^2} \log(d_1 mT)\end{aligned}$$

And we should see that we can basically combine these terms to get the overall result that:

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-head}}; \epsilon; X^{(1)}, \ldots, X^{(m)}; \|\cdot\|_2) \lesssim (L_\sigma B_X)^2 \cdot \frac{\left((B_V^{2,1})^{\frac{2}{3}} + (B_{QK}^{2,1} B_V)^{\frac{2}{3}}\right)^3}{\epsilon^2} \cdot \log(mT)$$

Note that we can use $N = mT$ because, as mentioned above, we were able to use the infinity norm to treat each sample and input token separately, so thus the number of samples because the length of the input sequence times the number of samples!

4 Capacity bounds for multi-layer Transformers

The final large conclusion from the first section is that the covering number of multi-layer transformers also scales logarithmically with the input sequence.

The basic intuition behind this section is the idea of creating a covering for each individual layer of the multi-layer transformer. We first define the function for a multi-layer transformer as

follows:

$$f_{\text{tf-heads}}\left(X; \{W_V^{[h]}, W_{QK}^{[h]}\}_{h=1}^H\right) := \sum_{h=1}^H f_{\text{tf-head}}\left(X; W_V^{[h]}, W_{QK}^{[h]}\right)$$

Let us define the class of multi-head self-attention with H heads as

$$\mathcal{F}_{\text{tf-heads}} := \Big\{X \mapsto f_{\text{tf-heads}}\left(X; \{W_V^{[h]}, W_{QK}^{[h]}\}_{h=1}^H\right) : \\ \forall h \in [H], \left\|W_V^{[h]}\right\|_{2,1} \leq B_V^{2,1[h]}, \left\|W_V^{[h]}\right\| \leq B_V^{[h]}, \left\|W_{QK}^{[h]\top}\right\|_{2,1} \leq B_{QK}^{2,1[h]}\Big\}.$$

Basically, we can create a covering per-layer and bound the errors to be appropriately small. Each of these coverings will be basically of the same form as the single layer:

$$\log \mathcal{N}_\infty(\mathcal{F}_{\text{tf-head}}; \epsilon; X^{(1)}, \ldots, X^{(m)}; \|\cdot\|_2) \lesssim (L_\sigma B_X)^2 \cdot \frac{\left((B_V^{2,1})^{\frac{2}{3}} + (B_{QK}^{2,1} B_V)^{\frac{2}{3}}\right)^3}{\epsilon^2} \cdot \log(mT)$$

If have H transformer layers, then there we can basically sum up H of these individual layer coverings, each of which are bounded by the log of the input sequence length, so we can just construct our ϵ errors at each layer to be small enough such that the error over the sum of these layers is as small as we want.

Ultimately, this allows us to write:

$$\log|\mathcal{C}| = \sum_{h=1}^H \log|\mathcal{C}_h| \leq \sum_{h=1}^H \leq (L_\sigma B_X)^2 \cdot \frac{\left(\sum_{h=1}^H (B_V^{2,1[h]})^{\frac{2}{3}} + (2B_{QK}^{2,1[h]} B_V^{[h]})^{\frac{2}{3}}\right)^3}{\epsilon^2} \cdot \log(dmT).$$

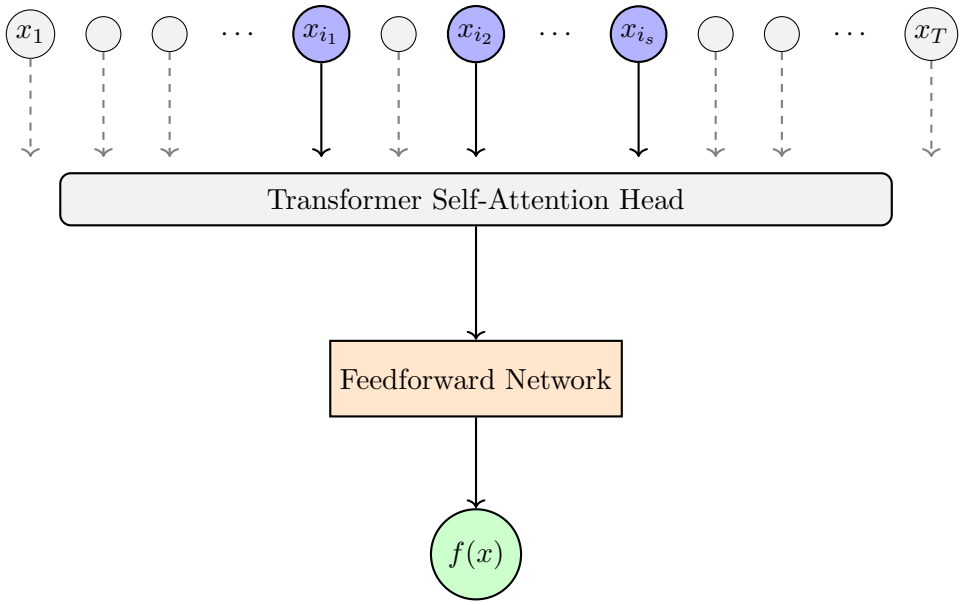
So, again, we have a logarithmic dependence on the input sequence.

5 Attention approximates sparse functions

This section addresses which functions are in the function classes that self-attention can represent. Building off of the theoretical findings of section 4 that Transformers have a limited capacity which scales with $\log(T)$, this section shows that this limited capacity is nonetheless sufficient to represent s-sparse Boolean functions, i.e. functions that depend on only a subset $s << T$ of input variables. This is important, as many real-world problems (e.g., key phrases in text, important tokens in a sequence) involve learning from a small number of relevant inputs within a large context. This section gives theoretical insight that Transformers can do this efficiently, or that self-attention **has an inherent inductive bias towards learning sparse interactions**.

For our definitions, consider a Boolean function $f : \{0,1\}^T \to \mathcal{Y}$. The authors of the paper introduce the following definitions:

Definition 5.1. (s-sparse Boolean function) The boolean function f is s-sparse if it only depends



Transformer learns an s-sparse function with $s \ll T$

Figure 4: Illustration of a Transformer learning an s-sparse Boolean function. Only a few inputs (in blue) affect the output, while most tokens (in gray) are ignored.

on a fixed subset $I \subset [T], |I| = s$ of its inputs:

$$b_i = b'_i \quad \forall i \in I \implies f(b) = f(b')$$

Essentially, the function f can "focus" on only a subset of the inputs and ignores the rest.

Definition 5.2. (symmetric) An s-sparse Boolean function f is symmetric if its value is invariant under permutation of the indices in I:

$$|\{i \in I : b_i = 1\}| = |\{i \in I : b'_i = 1\}|$$

Definition 5.3. (B-bounded-norm) f is B-bounded norm if

$$\|f(b)\|_\infty \leq B \quad \forall b \in \{0,1\}^T$$

This is the standard definition for bounding a function with respect to the l_∞ norm.

Definition 5.4. (Uniform approximation where $\mathcal{Y} = \mathbb{R}$): For some $\epsilon \geq 0$ and a function $f : \{0,1\}^T \to \mathbb{R}^d$, we say that $\widehat{f} \in \mathcal{F}$ ϵ-uniformly approximates f under the mapping $b \mapsto X(b)$ if

$$\left\| \widehat{f}(X(b)) - f(b) \right\|_\infty \leq \epsilon, \quad \forall b \in \{0,1\}^T$$

Essentially, an estimator mapping $\widehat{f}$ which takes in the input $X(b)$ would be approximately close to the function f by at most an ϵ error.

This paper constructs a Transformer block to approximate or exactly represent s-sparse Boolean functions f. For this, there are two main components of the transformer block it uses:

- **Self-Attention Head ($f_{\text{tf-head}}$)**: The self-attention head learns to "attend" only to the s relevant inputs. This is done by using alignment scores (based on learned projections to give high attention weights to those positions, and near-zero to the other $T - s$ relevant inputs.

- **Feedforward Network (f_{mlp})**: The feedforward-networks acts as a memory function, taking the relevant input s and mapping it to the desired output using a small ReLU network. Depending on the s-sparse Boolean function, it will need $poly(s)$ (if symmetric) or $2^s \cdot poly(s)$ parameters.

For the transformer block, the authors consider $f_{\text{tf-scalar}}$, a transformer block with self-attention head only (no feed-forward network), or $f_{\text{tf+mlp}}$, a transformer block with a sefl-attention head composed with the feed-forward network.

The authors of the paper want to consider the class of s-sparse Boolean functions $f : \{0,1\}^T \to \mathbb{R}$ representable by a bounded-norm scalar output Transformer heads $f_{\text{tf-scalar}} : \{0,1\}^T \to \mathbb{R}$, which will allow them to filter based on the small subset of relevant s inputs. For this to happen, one must fix a mapping from length-T boolean strings $b \in \{0,1\}^T$ to the real-valued Transformer inputs $X \in \mathbb{R}^{T \times d}$. They must also take into account the Transformer network's permutation-equivariant symmetry (mentioned in [VSP+17], which means that X needs to be partition by assigning different embeddings to different indices of b. They consider these choices for input encoding:

- **Deterministic Positional Embeddings:** Fix positional embedding matrix $P \in \mathbb{R}^{T \times d}$, token embedding matrix $E \in \mathbb{R}^{\{0,1\} \times d}$ and a special direction $v_{[CLS]} \in \mathbb{R}^d$. With this, we have $T+3$ vectors $\{P_{t,:}\}_{i=1}^T \cup \{E_{j,:}\}_{j \in \{0,1\}} \cup \{v_{[CLS]}\}$ are an approximately orthonormal basis for $\mathbb{R}^d$. Thus, the input to the Transformer is

  $$X = E_b + P$$

  where $E_b \in \mathbb{R}^{T \times d}$ with $[E_b]_{t,:} = E_{b_t,:}$ for each $t \in [T]$. For $f_{\text{tf-scalar}}$, $x_{[CLS]} = v_{[CLS]}$

- **Trainable Positional Embeddings:** Similar to above except P is trainable.

- **Bag of vectors:** For a fixed matrix $V \in \mathbb{R}^{T \times d}$ with approximately orthogonal rows, the Transformer input is

  $$X := V \text{diag}(b)$$

  which replaces positional embeddings with positional indicator functions. This establishes symmetry with respect to the permutation of the Transformer's input positions:

  $$f_{\text{tf-scalar}}(V \text{diag}(b)) = f_{\text{tf-scalar}}(V \prod \text{diag}(b))$$

  for a $T \times T$ permutation matrix $\prod$.

The embedding scheme is important; since Attention mechanism relies on the dot products to align queries and keys, orthogonality ensures that dot products between different positions can align attention scores, allowing the model to focus on what *what the bit is* and *where it is*. After choosing a mapping $X(b)$, the setup of the representation problem is as follows: given $f(b)$, find Transformer weights $\theta_{\text{tf-Head}}$ and feedforward network weights θ_{mlp} such that

$$f_{\text{mlp}}(X(b); \theta_{\text{tf-Head}}, \theta_{\text{mlp}}) := \quad f_{\text{mlp}}\left(f_{\text{tf-Head}}(X(b); \theta_{\text{tf-Head}}); \theta_{\text{mlp}}\right) \approx f(b), \forall b \in \{0,1\}^T.$$

From this, the authors of [EGKZ22] show that for any size-s subset of indices $\mathcal{I} \subseteq [T]$, Transformers blocks can represent all $\mathcal{I}$-sparse Boolean functions, whose values only depend on the inputs at the coordinates in $\mathcal{I}$. For a formal statement, they show the following:

Proposition 5.5 (Sparse Variable Creation from Transformers)**.** *For any input mapping from a boolean $b \in \{0,1\}^T$ to $X(b)$, we have the following guarantees:*

- *Given matrix norm bounds $\|W_Q\|_F \leq O(\log(Ts))$; $\|W_K\|_F, \|W_V\|_F, \|W_C\|_F \leq O(s)$, $f_{\text{tf-scalar}}$ can approximate a **monotone symmetric s-sparse** Boolean function. This scales with $O(\log(T))$*
- *Given matrix norm bounds $\|W_Q\|_F \leq O(\log(Ts))$; $\|W_K\|_F, \|W_V\|_F, \|W_C\|_F \leq O(s)$ and feedforward network weights satisfy $\|W_1\|_F, \|W_2\|_F, \|w\|_F \leq O(\text{poly}(s))$, $f_{\text{tf-mlp}}$ can exactly represent **symmetric s-sparse** functions. This scales with $O(\log(T))$.*
- *Given matrix norm bounds $\|W_Q\|_F \leq O(\log(Ts))$; $\|W_K\|_F, \|W_V\|_F, \|W_C\|_F \leq O(s)$ and the feedforward network weights satisfy $\|W_1\|_F, \|W_2\|_F, \|w\|_F \leq O(2^s \cdot \text{poly}(s))$ $f_{\text{tf-mlp}}$ can exactly represent **general s-sparse functions**. This scales with $O(\log(T))$.*

Proof. (Proof sketch). For all constructions of the proof have the following idea:

1. Design the attention weights W_Q, W_K query-key matrices t construct an attention head that selects the s relevant tokens. This is done by choosing only the relevant bits receiving significant attention weights, and outputing a sum over the relevant token embeddings. This means that the dot product as part of the score function $x_t^T W_k^T W_Q z$ is high for the relevant s tokens, and low for the remaining $T - s$ tokens. Applying the softmax, the attention head produces a sparse attention vector $v \in \mathbb{R}^d$ that encodes the sum of s relevant token embeddings
2. Decode the produced vector v into the Boolean output $f(x)$. This is down as follows: The MLP (feedforward network) uses ReLU layers to partition the space and map input combinations to correct outputs, acting like a decoder over the selected bits. This allows the layer to memorize all distinct values of f, using $O(s^2)$ neurons in the worst case and $O(poly(s))$ size for symmetric or structured function (as discussed in Proposition 5.5)

□

Given that there are $\binom{T}{s}$ s-sparse subsets of input indices, the sample complexity of learning a sparse boolean function must scale at least $\Omega(s \log T)$. This matches the capacity bounds in terms of $\log T$ dependence. There are other realizable functions for the bounded-norm Transformer setup, but these are not considered in the paper.

6 Synthetic Experiments

The authors of [EGKZ22] show empirical evidence supporting that the theoretical claims from sections 4 and 5, with sample complexity scaling as $\log(T)$ for the input length T.

Experimental Setup: The authors create synthetic tasks: learning s-sparse Boolean functions (e.g., 3-bit AND or XOR) over a large number of inputs bits T (context T). Let m be the sample size. For an input $\{0,1\}^T$ and label $y = f(x)$ where some boolean function f depends only on a random subset of s bits, they consider an i.i..d. Bernoulli input distribution $\mathcal{D}$ on $\{0,1\}^T$. They

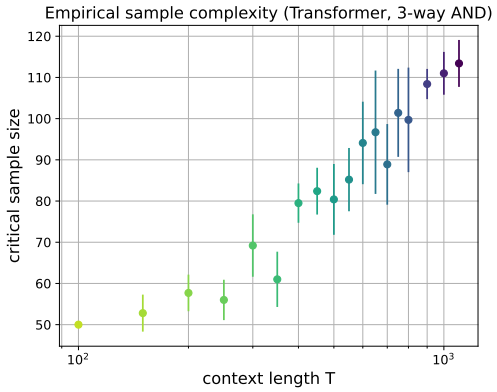


Figure 5: The sample complexity of learning a 3-sparse AND function of T input bits with Transformers. For each T, the authors measure the smallest sample size m necessary to reach 100% validation accuracy on $\geq 80\%$ of random trials.

consider a family of distinct functions $\{f_i : \{0,1\}^T \to \{0,1\}\}_{i=\in[N]}$, where N is proportional to T and f_i to be the set of all $\binom{T}{s}$ conjunctions of s inputs, i.e. $y = x_2 \wedge x_3 \wedge x_{10}$. The model learns which subsets of s features are relevant, which requires at least $m \geq \Omega(s \log T)$ samples for $\binom{T}{s}$ possibilities, as proven in the previous section. The model used is a 1-layer Transformer network, which is trained as a binary classifier. The goal of the experiment is to know how many samples m are needed for the Transformer to generalize accurately as T increases. The experiments run as follows:

- Choose an $i^* \in [N]$ uniformly at random to get labels given by f_{i^*}.
- Train a one-layer Transformer (in binary classifier configuration) on m samples from $\mathcal{D}$ with labels given by f_{i^*}.
- Evaluate generalization error.
- Measure the empirical scaling measure - the smallest sample size $m(T)$ at which the model training succeeds with 100% accuracy - in terms of N.

For this, the author shows that $m \geq \Omega(\log N)$ samples are required for any learner to reach 100% accuracy on this sample.

In the experimental setup, the authors set $s = 3$. this means that in each trial, there are $\binom{T}{3}$ subsets of indices selected at random. They measure how the empirical scaling measure correlates with T.

Results: Given architecture and training hyperparameters typical of a Transformer setup (Adam, full-batch gradients of the cross entropy loss for binary classification, etc), the authors of the paper obtain the following results from their experiments with the 1-layer Transformer network:

1. Empirical sample complexity scales as $log(T)$
2. Attention weights vanish on the $T - s$ irrelevant coordinates, implying sparse solutions.

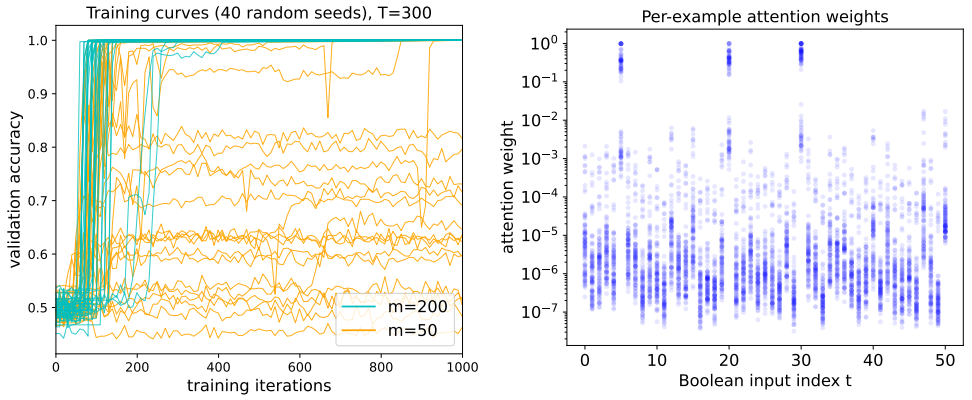


Figure 6: Visualizations for the sparse function learning experiments. *Left:* Examples of validation accuracy curves on the same problem instance ($T = 300$), with sample sizes $m = 200$ and $m = 50$ the threshold (≈ 70 from Figure 6). Training accuracy goes to 100% in both cases, but the Transformer overfits (orange curves) when m is too small. *Right:* Per-example attention weights for a successfully trained model ($T = 50$, $m = 300$, $I = \{5, 20, 30\}$). Attention weights approximately zero out the irrelevant bits.

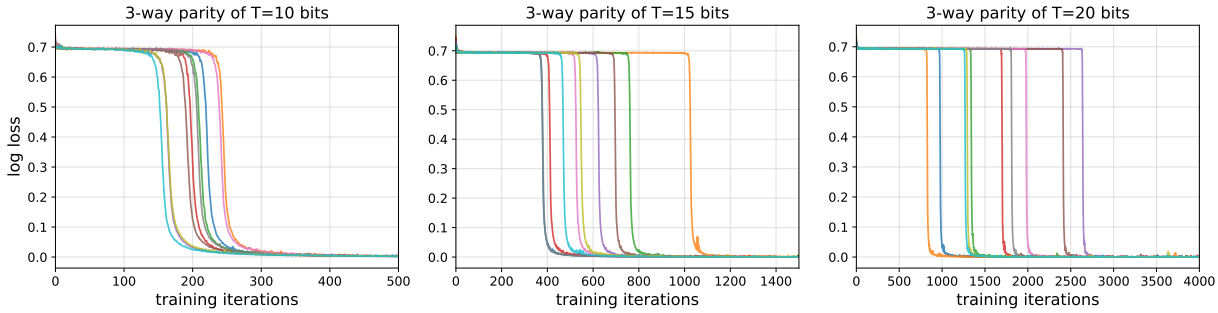


Figure 7: Transformers can learn sparse parities XOR. Loss curves (across 10 random seeds for initialization and SGD samples) are shown for this setup with $s = 3, T \in \{10, 15\}$.

Sparse parities (XOR): Although the XOR function is computationally hard to learn in statistical query models, [EGKZ22] shows that a Transformer succeeds in learning the XOR function. However, there are currently no tools to provide a theoretical analysis of why this is the case.

7 Future Directions

The authors of [EGKZ22] discuss potential extensions to the this paper:

- Refining the covering number bounds given that the paper obtained optimal dependence on T only.
- Refining the representation results, which only used MLP's capacity for exhaustive memorization.
- Analyzing the activations of SOTA Transformer language models's attention mechanisms.

References

[BM02] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[CLZ19] Minshuo Chen, Xingguo Li, and Tuo Zhao. On generalization bounds of a family of recurrent neural networks. *arXiv preprint arXiv:1910.12947*, 2019.

[EGKZ22] Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pages 5793–5831. PMLR, 2022.

[Hau14] David Haussler. Lecture 8: Vc dimension and covering numbers. Course Notes for CS6783, 2014. Accessed: April 14, 2025.

[JNM+19] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[Tew23] Ambuj Tewari. Lecture 16: Covering numbers. Course Lecture Notes, 2023.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Zha02] Tong Zhang. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2(Mar):527–550, 2002.