# Empirical Capacity Model for Self-Attention Neural Networks

Aki Härmä[0000−0002−2966−3305], Marcin Pietrasik[0000−0001−7559−8658], and Anna Wilbik[0000−0002−1989−0301]

Department of Advanced Computing Sciences (DACS), Faculty of Science and Engineering, Maastricht University, The Netherlands
aki.harma@maastrichtuniversity.nl

**Abstract.** Large pretrained self-attention neural networks, or *transformers*, have been very successful in various tasks recently. The performance of a model on a given task depends on its ability to memorize and generalize the training data. Large transformer models, which may have billions of parameters, in theory, have a huge capacity to memorize content. However, the current algorithms for the optimization fall short of the theoretical capacity, and the capacity is also highly dependent on the content. In this paper, we focus on the memory capacity of these models obtained using common training algorithms and synthetic training data. Based on the results, we derive an empirical capacity model (ECM) for a generic transformer. The ECM can be used to design task-specific transformer models with an optimal number of parameters in cases where the target memorization capability of the task can be defined.

## 1 Introduction

The central processing element of the transformer model [28] is the self-attention circuit [4], which computes weighted sums of input vectors based on content. Large transformer models, which may have billions of parameters, typically have multiple layers of parallel sets, or *multihead*, self-attention circuits, and several other processing units. The parameters are optimized using stochastic gradient backpropagation methods [10]. Neural networks based on transformer architecture have been very successful in recent years in various tasks such as natural language processing [7,13], speech recognition [19], and image processing [14].

The memory capacity discussed in this paper does not refer to the ability of a model to learn a causal pattern in the sequences, which has been studied, for example, in [30,18]. With a sufficient number of parameters, a neural network can memorize, or *shatter* [26], the machine learning problem. The self-attention circuit can be considered as an associative memory that has a certain capacity determined by the number of parameters. Transformer models have a known relation to Hopfield networks [21] and associative memories [20,6,22]. In theory, transformer models may have a large storage capacity that depends on the architecture choices [9,15]. However, translating those results into actual attainable capacity is difficult. Allen-Zhu *et al.* have recently published a series of preprints about the

performance of transformer models using synthetic data derived from a knowledge base [3,2,1]. The capacity in those works was not defined by exact memorization, which is the scope of the current paper, but the ability to retrieve "one piece of human knowledge" [2], which is naturally a meaningful goal but difficult to define in a more general case than synthetic content. Interestingly, they concluded that a transformer model can store 2 bits of knowledge per parameter [2].

The *scaling laws* for large transformer models have been discussed, based on the analysis of training loss, for example, in [8], but those results do not directly translate to capacity estimates. Kim *et al.* showed that the capacity of transformer models follows a trend similar to an asymptotic theoretical model [9]. Mahdavi *at al.* proposed a model for the capacity for a single self-attention layer which was also compared to numerical performance [15]. However, there seem to be few previous studies that address the actual attainable storage capacity of complete transformer models from the perspective of deriving practical design rules.

In this paper, we measure the capacity of transformer models computationally by training models of different sizes using synthetic data. Based on various experiments with different sizes and architecture parameters, we fit a function that gives the expected capacity of the model. The function outperforms a polynomial function at a fraction of the trainable parameters. The prediction of the model can be considered as an empirical lower bound for the capacity, which can make it a useful tool for designing architectures that meet the required performance criteria of a particular application. We demonstrate that the model can be used to predict the capacity of unseen model architectures. The use of the function can lead to savings in processing costs and also facilitate the development of new Retrieval Augmented Generation, RAG, methods [12] based on the memorization capacity of large models.

## 2   Capacity in networks

The basic quality measure for the Hopfield networks is the storage capacity, understood as the maximum number of patterns that are remembered as fixed points by the network. The storage capacity of a Hopfield network with $N$ nodes is $CN/log(N)$ where $C < 1/2$ [17], and the stored *memories* are binary vectors of length $N$. It is interesting to note that a Hopfield network of the size of GPT-3 [7], with 175B parameters, could theoretically have the capacity to memorize 3.4B sequences of 2048 tokens. The reported number of GPT-3 training data tokens is 500B, that is, about 7 percent of the theoretical capacity of the network of this thought experiment. Based on the rule of 2 bits per parameter in [3] this corresponds to 44 Gb of memorized *knowledge*.

The storage can be measured as mean squared errors (MSEs) between the original and retrieved patterns across different numbers of patterns the model was trained to memorize [24]. Some authors use recall rate [31], measured as a rate of how many patterns can be reconstructed correctly given a pattern with a certain number of missing bits. Steinberg *et al.* [23] found that there is a

minimum initial cue (relative to the total length) that gives a very small recall error. This minimum initial cue is defined as

$$l_c = \frac{L_0}{L},$$

where $L_0$ is the cue length and $L$ is the pattern length.

Besides storage capacity, many authors were also interested in measuring the so-called basin of attraction [23,25]. The basin of attraction is understood as a region around a pattern in which all states are attracted to this pattern within a prescribed time. There is no analytical way to study the basin of attraction except by drawing a graph which shows the number of successes out of many trials against the number of input errors.

An interesting characteristic of Hopfield network is a degree of symmetry, introduced by Krauth [11] and calculated as

$$\frac{\Sigma_{i=1}^n \Sigma_{j=1}^n w_{ij} wji}{\Sigma_{i=1}^n \Sigma_{j=1}^n w_{ij}^2} \tag{1}$$

Many researchers believed that the maximal number of connections and 100% symmetry of the weight matrix disabled to achieve its maximal storage capacity. Others claim that due to the asymmetry of the weights matrix, the number of false attractors increases.

Baum *et al.* [5] have shown that a feedforward net of one hidden layer with $N/d$ internal units can perfectly learn an arbitrary two-class classification problem, a *dichotomy*, of $N$ $d$ dimensional vectors [5]. The storage capacity of RELU feedforward networks was studied by Vardi *et al.* [27]. In their model, the capacity in the big-O notation is $\tilde{O}(\sqrt{N})$. [9] recently proposed that in a transformer architecture, $\tilde{O}(d+n+\sqrt{nN})$ parameters are enough to memorize $N$ sequences of length $n$ with a token vector dimension $d$. However, this does not seem to translate into a practical design rule that gives a memorization requirement. Mahdavi et al. [15] proposed that a single multihead self-attention layer of $\Theta(Hd^2)$ parameters can memorize $O(HN)$ sequences of length $N$. However, their model was based on $d = d_h$ which is not a typical choice in transformer models.

## 3   Transformer models

The input $\mathbf{X}$ into the model is a sequence of $N$ discrete symbols $t_i$, or tokens in the NLP terminology. In the model, each sequence of tokens is represented by the vectors of $B$ elements $x_i, i = 0, \cdots, N-1$. The core component is the self-attention circuit which in the conventional form is given by

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2}$$

where $Q = XW_Q$, $K = XW_K$, and $V = XW_V$, and matrices $W_Q$, $W_K$, and $W_V$ are $B \times d_h$ matrices of a selected $x_i$ token vector size $B$, and a head dimension

$d_h$. The coefficients of the three matrices are trained in the model optimization process. For example, the model of [15] suggests that the capacity to memorize $C$ independent data sets is related to $O(HN)$, where $H$ is the number of attention heads.

The self-attention circuit is the key component of the transformer architecture and is mainly responsible for the memorization of training data. The $QK^T$ matrix operation can be rewritten as

$$QK^T = XW_Q(XW_K)^T$$
$$= XW_Q W_K^T X^T = XW_A X^T, \quad (3)$$

that is, the rightmost *quadratic form* with one $d \times d$ matrix $W_A$ is equivalent with the original $QK^T$ key-query form that has two $d_h \times d$ matrices. Usually in LLMs $d >> 4d_h$, and therefore the second form of (3) is more efficient than the last form with a matrix $d \times d$. However, if $d \leq 2d_h$, the effective number of parameters in a single attention circuit is $d^2 + dd_h \leq 3d^2$.

Moreover, omitting $W_V$, which is commonly done in some transformer implementations, we may write the

$$\text{Attn}(Q, K, V) = MX \qquad (4)$$

where the matrix $M = f(X)$, where $f()$ is some function learned by the transformer network. This operation of multiplying the array of input vectors in $X$ by a matrix that depends on $X$, based on learning, can be seen as the key operation in transformer models.

A complete transformer system, see Fig. 1 consists of multihead attention modules, which have $H$ parallel and independent self-attention (SA) circuits. An array of $H$ SA blocks in parallel on the same input is called a multi-head attention block. Often, large models consist of several layers of multi-head blocks. The normalization and dropout models have been left out of the figure, but they are also a part of the implementation used in the following experiments.

After each layer, there is a dense feedforward (FFN) layer with a GELU activation function. One may suggest that most of the memorization capacity of the model is in the SA circuits. To show this, models were trained with trainable FF layers and by freezing the coefficients of the FF layer to an identity operation that maps the inputs to the output without modification. The results of a two-head, two-layer transformer for different sizes of the token vector $B$ are shown in Fig. 2. For small vector sizes, the curves are identical, which confirms that the FF layers do not contribute to the memorization of the synthetic content used in this paper. The capacity at the larger values of $B$ seems to even increase after freezing the FFN parameters. It is likely that the fully trainable model would require more iterations to reach the same capacity as the simpler model where the FF layer is replaced by an identity operation.

The total number of parameters of a self-attention network depends on the sizes of the vectors $d_h$ and $d$, the maximum length of the input sequence $N$, and the number of heads $H$ and layers $L$. Fig. 3 shows the total number of *trainable*
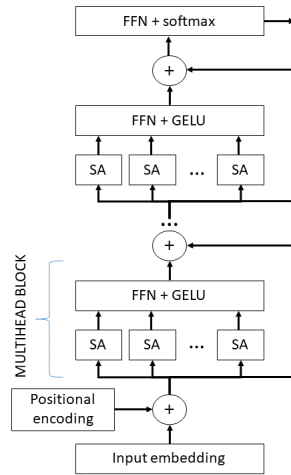
**Fig. 1.** A self-attention model with multiple layers and multiple self-attention heads.
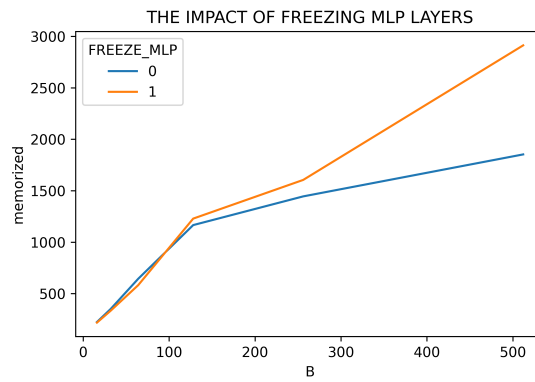


**Fig. 2.** A self-attention model with trained (0) and frozen (1) MLP layers.

*parameters* for different configurations. In the experiments reported in this paper, we freeze the coefficients corresponding to the input embedding and the final linear model at the end. The trainable parameters are therefore those of the SA and FFN modules.
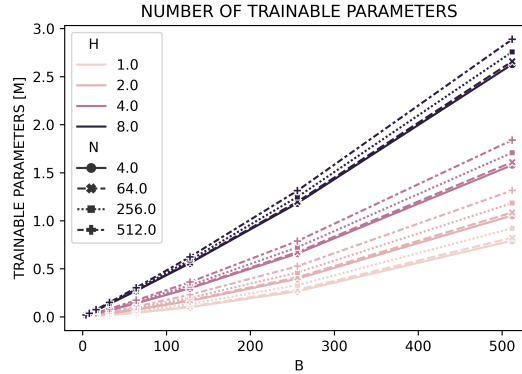


**Fig. 3.** Total numbers of trainable parameters (millions) for different variants of the transformer network.

## 4    Data sequences

Let us consider the system of $T$ tokens in an autoregressive task of predicting the next token

$$t_p = \text{argmax}_k F(t_k | t_i, i = 0, \cdots, k-1) \tag{5}$$

where the model $F()$ is a transformer followed by a softmax layer

$$F(x_i, i \neq k) = \text{softmax}(\text{Attn}(Q, K, V)H) \tag{6}$$

and $H$ is an $T \times N$ matrix with fixed coefficients.

The modeling task is an autoregressive task to predict the next token from the previous $N-1$ tokens. In this paper, we define the empirical capacity measure $C$ of a network as the number of sequences in which the network can correctly predict the $N^{th}$ token from a sequence of preceding tokens $N-1$. In a system of $T$ different tokens, the probability of *guessing* the correct token is $p = 1/T$ and the probability that a system gives the guess correctly $r \leq R$ times in a system of $K$ sequences is

$$P(r < R) = \sum_{r=0}^{R-1} \binom{K}{r} \frac{1}{T}^r (1 - \frac{1}{T})^{K-r} \tag{7}$$

For example, with $T = 128$ and $K = 2048$ the probability of 25 correct guesses (of an untrained network) is $P(r < 25) = 0.96$. Furthermore, if the probability of more than, say, 40 right guesses in this system is very unlikely, that is, $1 - P(r < 40) \approx 2.8 \times 10^{-7}$. We may use Equation 7 to assess the validity of the empirical capacity measures discussed in this paper. Furthermore, for a library size $K$, the expectation for the number of correct guesses is:

$$c_{\text{offst}} = \frac{K}{T}. \tag{8}$$

## 5    Experiments

There are several ways to address capacity measurement in learning experiments. In this paper, we consider two approaches. In the Maximum Library Size (MLS) method, the goal is that the network memorizes every item of the input vector library, and the measurement is performed by measuring how large the library can be fully memorized. In the Maximum Attainable Capacity (MAC) method, the model is trained with a large library, and the goal is to measure the maximum number of samples the network can memorize. The MLS method is obviously more tedious than the MAC method.

### 5.1    Comparison of the MAC and MLS methods

As shown in Fig. 4 the MLS and MAC capacity measurements have a similar overall shape but there is an offset in the MAC values. The offset is due to the correct guesses in the MAC method. In this case, the size of the library $K = 32000$ and with the token count of $T = 128$ we get, by (8), $K/T = 250$, which agrees with the offset.

For larger vector sizes $B$, the MAC capacity, after compensating for the offset, seems to be lagging behind the MLS capacity estimates. Therefore, we may consider MLS to represent a lower bound for storage capacity. Since the MAC case represents a more realistic use case, in this paper, we focus on modeling the results of MAC experiments.

The probability that the model training takes $k$ epochs before the model shutters (MLS condition) can be expected to follow the negative binomial distribution

$$P_r(X = k) = \binom{k + r - 1}{k}(1 - p)^k p^r \tag{9}$$

For example, Fig. 5 shows the histogram of the number of epochs for a transformer model to *shutter*, that is, learn perfectly all sequences in 1000 independent trials in case of 16 sequences of length $N = 8$ and with a token vector size $d = 16$, and therefore $2 \times d^2 = 512$ parameters in total.
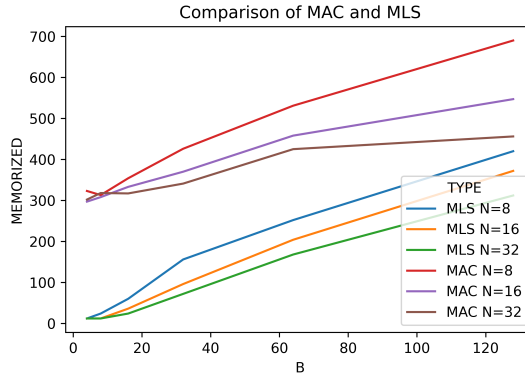
**Fig. 4.** Comparison of the capacity measurements with $H = 4, L = 1$ in MSL and MAC methods
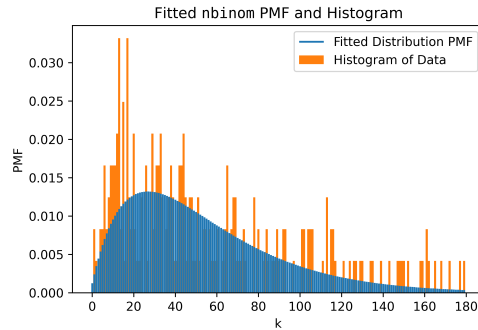


**Fig. 5.** Number of epochs needed to shutter the entire data set

### 5.2   Impact of batch size

The number of MAC memorized sequences of 32 or 128 tokens, as a function of batch size over a long training time, is shown in Fig. 6. The memorization is low with small batch sizes, which is also reflected in the large variation in the gradient noise. The capacity grows when the batch size increases and saturates, for the model sizes studied in this paper, after the batch size of 512.

In this paper, the transformer model is based on the implementation in the popular x-transformers Python library [29] and is trained using the Adam [10] optimizer available in the PyTorch library, for which we use default settings. The selection of the optimal batch size can be estimated based on noise gradients; see, e.g., [16]. In this paper, we use a batch size of 512. In total, we trained approximately 500 models, the total training time was 260 hours on an Nvidia A100-SXM4-40GB GPU. Each training was restarted five times, and we selected the run with the highest number of memorized vectors. A run was terminated
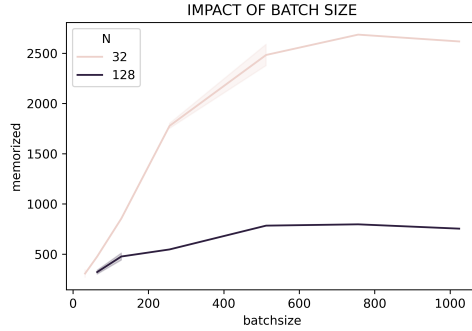
**Fig. 6.** The impact of batch size on the performance for a model with $H = 1$ and $B = 512$
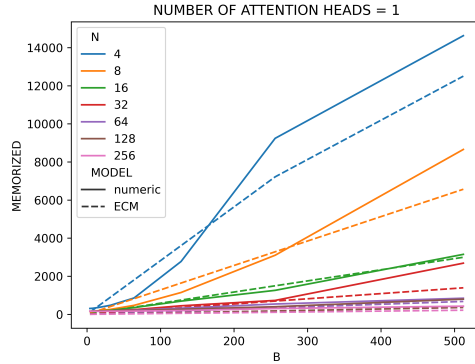


**Fig. 7.** Results of our ECM with $H = 1$. Solid lines represent obtained capacity values whereas dashed lines represent our model predictions.

when there was no improvement in the number of memorized sequences after several epochs.

Fig. 7 shows the number of sequences fully memorized for MAC experiments in a one-layer network of $H = 1$ attention heads, for different vector sizes $B$. The experiments reported in this paper were performed using a library size of 16000 sequences. Furthermore, the core model uses absolute positional embedding for the tokens and the dimension of the attention head is $d_h = 128$, which is a common choice in several published models.

### 5.3 Detailed figures of measured model capacities

The measured and predicted capacity figures for different transformer model architectures are shown in Figs. 8.
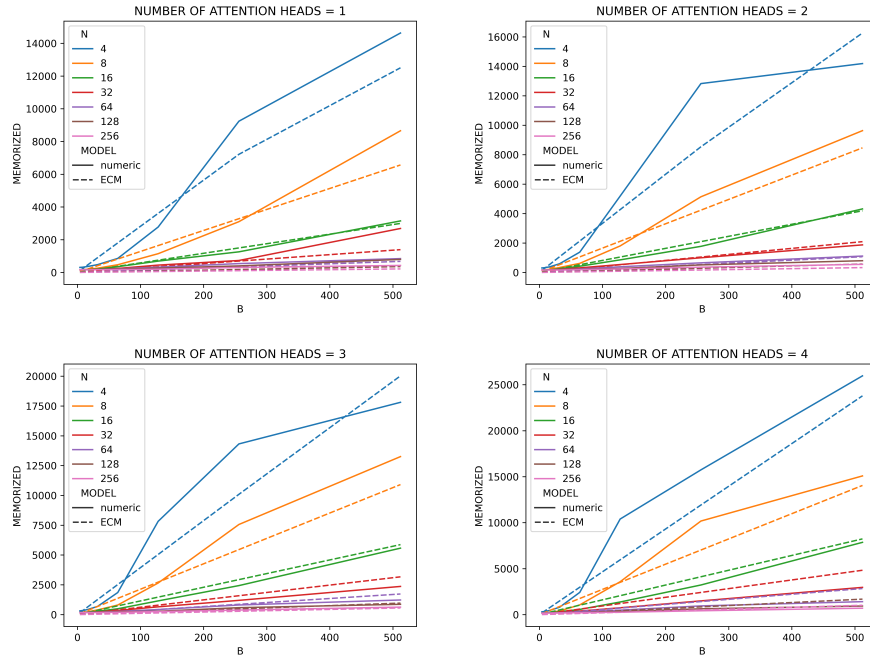
**Fig. 8.** Results of our ECM. Solid lines represent obtained capacity values whereas dashed lines represent our model predictions.

## 6   Empirical capacity model

We utilize the results obtained from the experiments to develop an empirical capacity model for self-attention transformers. The model describes the relationship between the number of memorized sequences as a function of the transformer's hyperparameters. We build our model sequentially, that is, by decomposing it into intuitions obtained from examining the marginal effects of each of its inputs. These effects are then modeled using low-order algebraic functions. This allows for a capacity model that is interpretable and low in the number of parameters. We show that even such a simple formulation outperforms more complex, high-order polynomial models. What follows is a description of the marginal impacts of the model's inputs and how they are leveraged to derive a capacity function.

### 6.1   Marginal impacts of hyperparamaters

We begin by capturing the marginal impact of $B$, that is, its behavior independent of the values of $H$ and $N$. This is shown by the solid lines of Fig. 7, which plots the number of memorized sequences as a function of $B$ at differing $H$ and $N$. We notice that this number increases monotonically with larger $B$ values until a saturation point is reached. In our experiments, this saturation occurs at or

slightly below the total number of sequences. We reflect this intuition in our modeling. Specifically, we first model the initial rise using a linear function. A logistic function was also considered to capture the curves' slight sigmoidal nature; however, we found that the increase in the parameters in such a modeling was not empirically justified by a closer fit. In modeling a linear rise, it is necessary to determine the value of two parameters, the slope and the intercept. The intercept is trivial since a value of $B = 0$ implies a transformer with dimensionless features and thus no memorized sequences, resulting in an intercept of 0. The slope is determined by the values of $H$ and $N$, which we model by the function $f(H, N)$. We can thus model the presaturation of memorized sequences $C_{pre}$ as $C_{pre} = f(H, N) * B$. To model the slope values at the corresponding $H$ and $N$, $f(H, N)$, we first calculated them as the slope from the origin to the value at the highest $B$ before saturation. This then allows us to model the impact of $H$ and $N$ on $C$ as captured in Fig. 9. In this regard, we notice an exponential decay in the slope with increasing $N$ values. Indeed, during our modeling, the exponential decay function was our first consideration. We found, however, that it does not capture the tapering that occurs at higher $N$ values. As such, we opted for a generalized rational function with a power function denominator. This allows us to better capture the changes in slope, whilst still maintaining an explainable formulation with few parameters. Finally, we notice that $H$ influences the rate of decay in the slope with the rate linearly rising at an increasing number of heads. This phenomenon corresponds to the exponent of the power term and can be modeled accordingly. Specifically, we raise $N$ to a linear function of $H$, resulting in the following slope function:

$$f(N, H) = \frac{a}{N^{b*H+c} + d} + e,$$ (10)

where $a$, $b$, $c$, $d$, and $e$ are parameters learned from our experiments. Their values are summarized in Table 1. Our slope formulation lends itself to a simple explanation for both input variables. $N$ represents the exponential decay in the rate at which sequences are memorized as a function of $B$, and $H$ influences the rate of this decay with higher values of $H$, resulting in a less pronounced drop.

**Table 1.** Learned parameters of our capacity models (slope and saturation).

| Layers | a | b | c | d | e | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|---|---|
| 1 | 145.27 | -0.13 | 1.29 | 0.13 | 0.20 | 3762.70 | 8741.00 |
| 2 | 2.45 | -0.002 | 0.02 | -0.99 | -29.08 | 4413.10 | 14787.00 |

## 6.2 Empirical capacity model formulation

Having assessed the marginal impacts of the hyperparameters, the Empirical Capacity Model can be formulated. Presaturation $C$, which is modeled by a linear function, increases without bound with respect to $B$ and thus does not
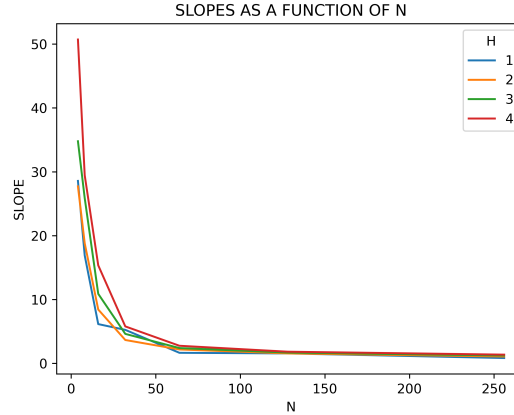
**Fig. 9.** Slopes corresponding to the linear rise in $C$ as a function of $B$ with respect to $H$ and $N$.

model the point of saturation. This saturation point is a function of $H$, with the number of memorized sequences saturating at higher values with more attention heads. This rise is modeled by the linear function $\alpha * H + \beta$ which we found to offer a good fit with minimal additional parameters. The learned parameters of this saturation function on our data are shown also in Table 1. This allows us to formulate our Empirical Capacity Model as the synthesis of the pre- and post-saturation states as follows:

$$C = max(f(H, N) * B, \alpha * H + \beta) \tag{11}$$

The fit of this model to our data is depicted graphically in Fig. 7. We can compare the fit of our model against a fifth-order polynomial of $H$, $N$, and $B$ as shown in the bottom row of this figure. We notice that despite the lower number of parameters (our model's seven compared to the polynomial's 56), our model provides a better fit. This is reflected in the mean absolute percentage errors (MAPE) of the two models. Specifically, our model has a MAPE of 0.497 compared to 1.0785 of the fifth-order polynomial. We analyze our proposed model in the following section.

## 7   Analysis of the capacity model

The parametric model of Equation 11 was optimized based on experiments with a library size of $K = 16000$. The ECM was fitted with a selection of parameter ranges, and it is reasonably accurate inside the region. However, extrapolation outside of the range often leads to an under / overestimation of the measured capacity; see, for example, Fig.10 for $H = 5$.
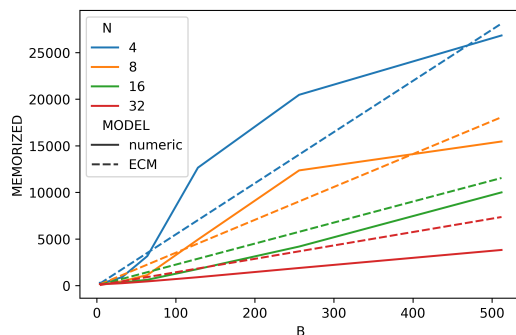
**Fig. 10.** Real capacity measures (solid) and ECM predictions (dashed) for $H = 5$.

The ECM formula can be used to compare different architecture choices. For example, Fig. 11 shows the total capacity of the model as a function of the total number of coefficients for the sequences of length $N = 64$. The plot shows that the capability increases significantly with more attention heads, $H = 1 \cdots 4$. For example, the capacity of a four-head model with 2M parameters is twice as high as that of a two-head model, that is, 1600 and 3300 fully memorized sequences, respectively. This agrees with the $O(HN)$ rule of [15].
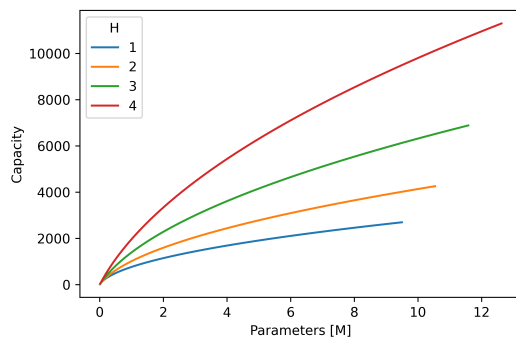


**Fig. 11.** The model memory capacity as a function of the total number of trainable coefficients.

We can leverage the ECM to examine the cost of adding additional parameters. In order to do so, we first define a measure that relates the number of parameters to the capacity of the model. Specifically, we define the memory per parameter (MPP) as the number of memorized tokens for each parameter in the model:

$$MPP = \frac{max(f(H, N) * B, \alpha * H + \beta)}{\#parameters} \tag{12}$$

By fixing the value of a hyperparameter and calculating the average MPP across the remaining hyperparameters, we can better understand the cost of

adding additional parameters to the model. We notice that smaller networks use their parameters more efficiently as per the MPP than do larger ones. Furthermore, increasing the number of heads results in a lesser drop in efficiency than increasing $N$. As such, the head parameters are less computationally costly than those of $N$.

## 8   Discussion and conclusions

In this paper, we evaluated the capacity of self-attention networks through an empirical analysis of their memorization potential in different settings. Such an analysis departs from the existing literature focusing on theoretical big-O bounds through the use of empirically derived capacities from experiments with synthetic data. In performing this work, our aim was to close the gap between the theoretical storage capacity and the factual, observed capacity of the transformer models that have garnered widespread interest and use in recent years. To this end, we developed an empirical capacity model which can be used to infer the capacity of a network given its hyperparameters. This allows practitioners to have a principled tool for a priori hyperparameter selection that ensures the maximal utility of each parameter is achieved. This model was developed through an analysis of the marginal impacts of hyperparameters on network capacity. Specifically, we tested models on a synthetic autoregressive task in which the transformer must predict a subsequent token given a previously observed token sequence. The number of correctly predicted sequences was used as a proxy measurement of the capacity of the model. We ran models to capture the network capacities at different points in the hyperparameter space. The results of these models served as the basis for a model constructed to reflect insights about the behavior of each hyperparameter. Specifically, insights were synthesized to obtain the model as described in Equation 11. This model, due to its simplicity and low number of parameters, is intuitive and outperforms more complex, higher-order polynomial models.

In the future, we plan to expand our work to include more comprehensive and realistic data and testing. Specifically, we will obtain denser results that span a larger hyperparameter range and introduce the effects of the number of network layers. Furthermore, we will expand the data used to include natural language content. This will ensure that our model's conclusions are relevant outside of synthetic experiments and can be applied in real-world contexts. Finally, we will present a more thorough analysis of our capacity model and include guidelines for a priori hyperparameter selection.

# References

1. Allen-Zhu, Z., Li, Y.: Physics of Language Models: Part 3.1, Knowledge Storage and Extraction (Jul 2024). `https://doi.org/10.48550/arXiv.2309.14316`, `http://arxiv.org/abs/2309.14316`, arXiv:2309.14316 [cs]
2. Allen-Zhu, Z., Li, Y.: Physics of Language Models: Part 3.2, Knowledge Manipulation (Jul 2024). `https://doi.org/10.48550/arXiv.2309.14402`, `http://arxiv.org/abs/2309.14402`, arXiv:2309.14402 [cs]
3. Allen-Zhu, Z., Li, Y.: Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws (Apr 2024). `https://doi.org/10.48550/arXiv.2404.05405`, `http://arxiv.org/abs/2404.05405`, arXiv:2404.05405 [cs]
4. Bahdanau, D., Cho, K.H., Bengio, Y.: Neural machine translation by jointly learning to align and translate: 3rd International Conference on Learning Representations, ICLR 2015 (Jan 2015), `http://www.scopus.com/inward/record.url?scp=85062889504&partnerID=8YFLogxK`
5. Baum, E.B.: On the capabilities of multilayer perceptrons. Journal of Complexity **4**(3), 193–215 (Sep 1988). `https://doi.org/10.1016/0885-064X(88)90020-9`, `https://www.sciencedirect.com/science/article/pii/0885064X88900209`
6. Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., Bottou, L.: Birth of a Transformer: A Memory Viewpoint (Nov 2023). `https://doi.org/10.48550/arXiv.2306.00802`, `http://arxiv.org/abs/2306.00802`, arXiv:2306.00802 [cs, stat]
7. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language Models are Few-Shot Learners. arXiv:2005.14165 [cs] (Jul 2020), `http://arxiv.org/abs/2005.14165`, arXiv: 2005.14165
8. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling Laws for Neural Language Models (Jan 2020), `http://arxiv.org/abs/2001.08361`, arXiv:2001.08361 [cs, stat]
9. Kim, J., Kim, M., Mozafari, B.: Provable Memorization Capacity of Transformers (Sep 2022), `https://openreview.net/forum?id=8JCg5xJCTPR`
10. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), `http://arxiv.org/abs/1412.6980`
11. Krauth, W., Nadal, J.P., Mezard, M.: The roles of stability and symmetry in the dynamics of neural networks. Journal of Physics A: Mathematical and General **21**(13), 2995 (jul 1988). `https://doi.org/10.1088/0305-4470/21/13/022`, `https://dx.doi.org/10.1088/0305-4470/21/13/022`
12. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. pp. 9459–9474. NIPS'20, Curran Associates Inc., Red Hook, NY, USA (Dec 2020)
13. Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C.D., Ré, C., Acosta-Navas, D., Hudson, D.A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K.,

Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S.M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., Koreeda, Y.: Holistic Evaluation of Language Models (Nov 2022). `https://doi.org/10.48550/arXiv.2211.09110`, `http://arxiv.org/abs/2211.09110`, arXiv:2211.09110 [cs]

14. Liu, Y., Zhang, Y., Wang, Y., Hou, F., Yuan, J., Tian, J., Zhang, Y., Shi, Z., Fan, J., He, Z.: A Survey of Visual Transformers. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2023). `https://doi.org/10.1109/TNNLS.2022.3227717`, `https://ieeexplore.ieee.org/document/10088164`, conference Name: IEEE Transactions on Neural Networks and Learning Systems

15. Mahdavi, S., Liao, R., Thrampoulidis, C.: Memorization Capacity of Multi-Head Attention in Transformers (Mar 2024). `https://doi.org/10.48550/arXiv.2306.02010`, `http://arxiv.org/abs/2306.02010`, arXiv:2306.02010 [cs]

16. McCandlish, S., Kaplan, J., Amodei, D., Team, O.D.: An Empirical Model of Large-Batch Training (Dec 2018). `https://doi.org/10.48550/arXiv.1812.06162`, `http://arxiv.org/abs/1812.06162`, arXiv:1812.06162 [cs, stat]

17. McEliece, R., Posner, E., Rodemich, E., Venkatesh, S.: The capacity of the Hopfield associative memory. IEEE Transactions on Information Theory **33**(4), 461–482 (Jul 1987). `https://doi.org/10.1109/TIT.1987.1057328`, `https://ieeexplore.ieee.org/document/1057328`, conference Name: IEEE Transactions on Information Theory

18. Nichani, E., Damian, A., Lee, J.D.: How Transformers Learn Causal Structure with Gradient Descent (Feb 2024). `https://doi.org/10.48550/arXiv.2402.14735`, `http://arxiv.org/abs/2402.14735`, arXiv:2402.14735 [cs, math, stat]

19. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust Speech Recognition via Large-Scale Weak Supervision (Dec 2022). `https://doi.org/10.48550/arXiv.2212.04356`, `http://arxiv.org/abs/2212.04356`, arXiv:2212.04356 [cs, eess]

20. Radhakrishnan, A., Belkin, M., Uhler, C.: Overparameterized neural networks implement associative memory. Proceedings of the National Academy of Sciences **117**(44), 27162–27170 (Nov 2020). `https://doi.org/10.1073/pnas.2005013117`, `https://www.pnas.org/doi/abs/10.1073/pnas.2005013117`, publisher: Proceedings of the National Academy of Sciences

21. Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G.K., Greiff, V., Kreil, D., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: Hopfield Networks is All You Need (Apr 2021). `https://doi.org/10.48550/arXiv.2008.02217`, `http://arxiv.org/abs/2008.02217`, arXiv:2008.02217 [cs, stat]

22. Schaeffer, R., Zahedi, N., Khona, M., Pai, D., Truong, S., Du, Y., Ostrow, M., Chandra, S., Carranza, A., Fiete, I.R., Gromov, A., Koyejo, S.: Bridging Associative Memory and Probabilistic Modeling (Feb 2024), `http://arxiv.org/abs/2402.10202`, arXiv:2402.10202 [cs]

23. Steinberg, J., Sompolinsky, H.: Associative memory of structured knowledge. Scientific Reports **12**(1), 21808 (2022)

24. Tang, M., Salvatori, T., Millidge, B., Song, Y., Lukasiewicz, T., Bogacz, R.: Recurrent predictive coding models for associative memory employing covariance learning. PLoS computational biology **19**(4), e1010719 (2023)

25. Tavan, P., Grubmüller, H., Kühnel, H.: Self-organization of associative memory and pattern classification: recurrent signal processing on topological feature maps. Biological Cybernetics **64**, 95–105 (1990)

26. Vapnik, V.N., Chervonenkis, A.Y.: On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. Theory of Probability & Its Applications **16**(2), 264–280 (Jan 1971). `https://doi.org/10.1137/1116025`, `https://epubs.siam.org/doi/10.1137/1116025`, publisher: Society for Industrial and Applied Mathematics
27. Vardi, G., Yehudai, G., Shamir, O.: On the Optimal Memorization Power of ReLU Neural Networks (Oct 2021), `https://openreview.net/forum?id=MkTPtnjeYTV`
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need (Dec 2017). `https://doi.org/10.48550/arXiv.1706.03762`, `http://arxiv.org/abs/1706.03762`, arXiv:1706.03762 [cs]
29. Wang, P.: lucidrains/x-transformers (May 2024), `https://github.com/lucidrains/x-transformers`, original-date: 2020-10-24T22:13:25Z
30. Xie, S.M., Raghunathan, A., Liang, P., Ma, T.: An Explanation of In-context Learning as Implicit Bayesian Inference (Jul 2022). `https://doi.org/10.48550/arXiv.2111.02080`, `http://arxiv.org/abs/2111.02080`, arXiv:2111.02080 [cs]
31. Zhong, C., Pedrycz, W., Li, Z., Wang, D., Li, L.: Fuzzy associative memories: A design through fuzzy clustering. Neurocomputing **173**, 1154–1162 (2016). `https://doi.org/https://doi.org/10.1016/j.neucom.2015.08.072`, `https://www.sciencedirect.com/science/article/pii/S0925231215012928`