Capacity Matters: Investigating Transformer Models for Real-World Data Memorization

Anonymous ACL submission

Abstract

Transformer models' memorization capacity studies often focus on theoretical bounds or use synthetic datasets that lack real-world complexity. This study systematically evaluates how model architecture and data configurations influence the capacity of decoder transformers using datasets derived from the Systematized Nomenclature of Medicine (SNOMED) knowledge graph: triplets, representing static connections, and sequences, simulating complex relation patterns.

Our findings highlight key factors affecting training dynamics and memorization. Embedding size is the primary determinant of learning speed and capacity, while additional layers provide limited benefits and may hinder performance on simpler datasets. Activation functions play a crucial role, with Softmax demonstrating greater stability and capacity. Additionally, increased dataset complexity enhances final memorization. These insights improve our understanding of transformer memory mechanisms and provide a framework for optimizing model design with structured real-world data.

1 Introduction

Transformer-based Large Language Models (LLMs) have revolutionized natural language processing by demonstrating remarkable capabilities in tasks ranging from text generation and translation to question answering and summarization. Despite these advances, the fundamental mechanisms underpinning their capacity to memorize and retrieve structured knowledge remain an active area of research. Understanding these mechanisms is crucial for optimizing model performance, making it computationally cheap in order to apply to real-world problems. One particularly impactful example is healthcare, where LLMs could assist clinicians through wearable devices such as smart glasses or watches (Gupta et al., 2024; Wu et al., 2024; Balloccu et al., 2024). Due to privacy and reliability, the preferred system would be a local on-edge LLM with minimal computational requirements, but with a capacity to memorize all relevant facts in the relevant area of healthcare.

Recent theoretical and empirical studies have sought to quantify the memorization capacity of transformers. Kim et al. (2023) introduced mathematical bounds for memory capacity, demonstrating that transformers could memorize $O(d + n + \sqrt{nN})$ parameters, where d, n, N correspond to embedding dimensions, dataset size, and model size, respectively. Additionally, Kajitsuka and Sato (2024) proved, that $\tilde{O}(\sqrt{nN})$ parameters are not only sufficient, but also necessary for some types of transformers. Mahdavi et al. (2024) extended this work by analyzing the effects of multi-head attention on memorization, revealing the interplay between architectural components and the model's ability to store and recall information. The experiments in Härmä et al. (2024) used randomly generated sequences of numbers to evaluate the memorization capabilities of the transformer models on unstructured data. Most capacity studies use synthetic datasets because accurate capacity measurement becomes very difficult in the case of uncontrolled free text content.

The experiments reported in the current paper use sentence data generated from the knowledge graph which, while being controlled, has some of the hierarchical and relational complexity of real-world text content. More specifically, GPT-like transformer models (Brown et al., 2020) were trained to memorize structured sentences derived from the Systematized Nomenclature of Medicine (SNOMED) knowledge graph (KG) (El-Sappagh et al., 2018). SNOMED, a comprehensive medical ontology, encodes semantic relationships between medical concepts, offering a rich dataset to explore memory and retrieval mechanisms under realistic conditions. Exact memorization of a selection of such relations would be critical, for example, in the healthcare use case described above.

By employing both theoretical insights and empirical evaluation, this study seeks to answer three key research questions. How can real-world data, such as knowledge graphs, be used to investigate transformers' memorization capacity? How do architectural variations affect the efficiency and scalability of memorization in transformer models? How do dataset structure and complexity influence memorization behavior during training?

To measure the memorization capacity of transformer models, the Maximum Attainable Capacity (MAC) method was used. It is a computationally efficient alternative to the Maximum Library Size (MLS) method. While MLS involves iteratively training models on progressively larger datasets to determine the largest library size that can be fully memorized, MAC evaluates the practical limit of samples a model can retain when trained on a large dataset. Previous research has shown a strong correlation between MLS and MAC (Härmä et al., 2024), making MAC an effective and time-efficient choice for this study.

Our approach leverages structured datasets constructed through two methods: triplet generation and sequence generation. Triplets represent static relationships in the form `(Concept, Property, Related Concept)`, providing a baseline for assessing memorization. Sequences extend this by simulating graph traversal paths, capturing relationship patterns between concepts. These datasets allowed us to empirically analyze how model architecture, training configurations, dataset size, and complexity influence training dynamics and final memorization performance.

2 Methods

2.1 Data

2.1.1 Data Source and Preprocessing

To evaluate transformer-based models' memory and retrieval capabilities, we used SNOMED KG, which encodes medical concepts and their relationships as nodes and edges of a graph. It was accessed using the `owlready2` library (Lamy, 2017), filtering out non-informative or overly specific properties to ensure meaningful relationships. While graph transformers leverage Graph Neural Networks (Shehzad et al., 2024), our approach prioritizes a universal architecture applicable across diverse datasets. Hence, the graph was transformed to: (1) triplets, representing concept-property relationships (see 2.1.2), and (2) sequences, simulating graph traversal paths (see 2.1.3).

2.1.2 Triplets Generation

The goal of triplet generation was to create a dataset of the form `(Concept, Property, Related Concept)`, capturing semantic relationships in the SNOMED KG. This process (see Figure 1A) involves graph initialization and the exclusion of non-informative properties. After the algorithm extracts triplets: for each concept in the KG, it retrieves all allowed properties and their associated related concepts. Additionally, when multiple related concepts are associated with a `(Concept, Property)` pair, one is selected randomly to maintain uniqueness.

2.1.3 Sequences Generation

The sequence generation simulated graph traversal and encoded local and global graph structures. The complete algorithm is depicted in Figure 1B.

The extended graph (`G`) is constructed from an ontology by: (1) excluding banned properties, as in the triplets generation; (2) along with each relationship, adding an edge with opposite direction with a corresponding `reversed_` prefix for bidirectional traversal. Additionally, labels were cleaned (metadata were removed) to standardize their format. The sequences were generated to reflect the traversal path in the graph, capturing both nodes and edges: $(\texttt{node}_1, \texttt{edge}_1, \texttt{node}_2, \ldots, \texttt{node}_{n-1}, \texttt{edge}_{n-1}, \texttt{node}_n)$

For each sequence, the algorithm first selects a random starting node from the full graph `G`, ensuring that the node has at least one unused edge. A subgraph is then created around the starting node using a breadth-first search (BFS) with a depth defined by the `hops` parameter. This step limits the scope of the traversal to a manageable subset of the graph, improving performance by focusing on local neighborhoods.

Step 2 of the algorithm generates a sequence of nodes and edges by traversing the subgraph. The algorithm starts from a randomly selected node and go through available edges (neighbors are chosen randomly to introduce variability). Every time, check that the same `(node, edge)` pair is not already visited before, maintaining global uniqueness. The traversal stops when: a randomly chosen number of edges within a predefined range (`edge_count_range`) is reached, or no valid neighbors (those, that maintain uniqueness) remain

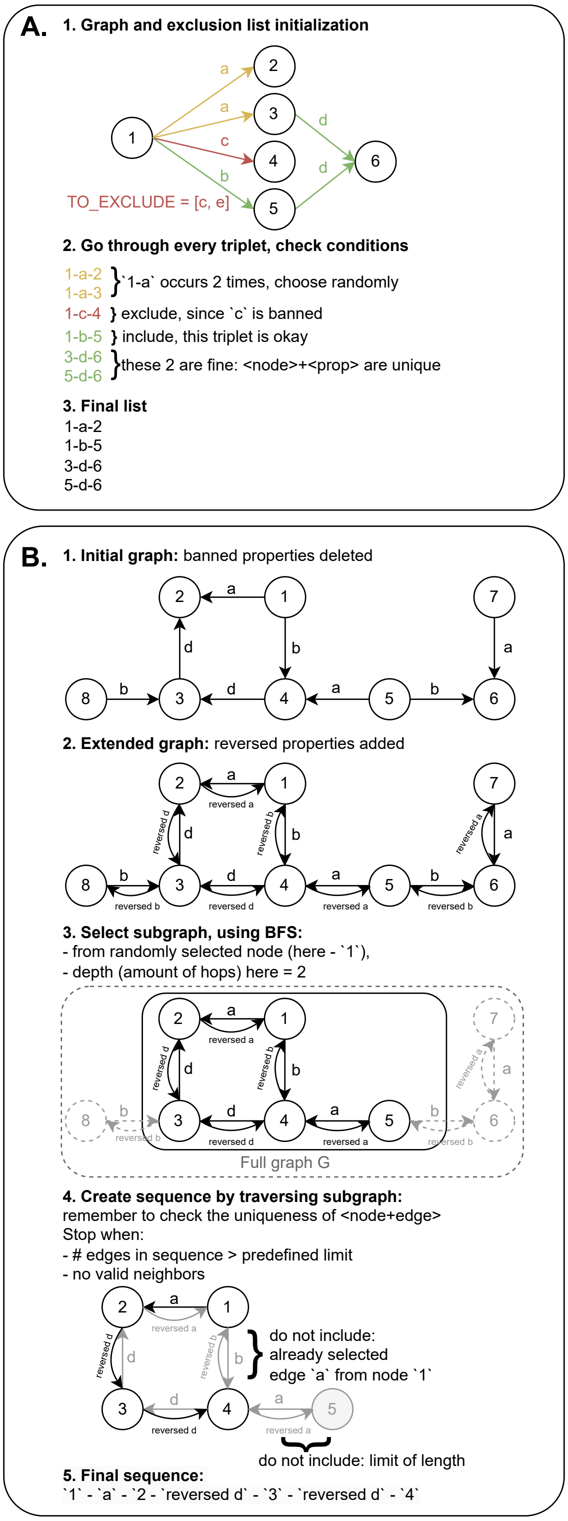

Figure 1: Algorithms of triplets (A) and sequences (B) data generation.

for further traversal.

The above steps are repeated for a specified number of iterations (rows), generating the desired number of sequences.

2.2 Transformers training

To evaluate the ability of transformer models to memorize and retrieve structured data, decoder-only transformer models with variations in architecture were implemented. Each unique node and edge was assigned a distinct integer identifier (ensuring that repeated elements were consistently tokenized), followed by the learned positional encoding. The core architecture consisted of three main components: an embedding layer to map tokenized inputs into continuous vector representations, transformer decoder layers with multi-head attention mechanisms, and a linear output layer to predict target tokens.

For all experiments, the task was to predict a concept, based on the previous concepts and relations. The accuracy was evaluated as: $\frac{\#correct\_predictions}{\#total\_predictions}$ – the proportion of correctly predicted related concepts to the total number of predictions. Additionally, Maximum Attainable Capacity (MAC) was used as a more suitable metric for measuring the model capacity. As detailed in the introduction, MAC is a computationally efficient alternative to Maximum Library Size (MLS), with results strongly correlated to MLS, making it the preferred choice for this research.

To minimize the effect of randomness, each experiment was repeated 10 times for the first and second setups, and 3 times for the third and fourth setup (see below). All figures and tables present mean values with doubled standard deviations. Training and evaluation followed a consistent protocol for all setups, with the training accuracy evaluated every second epoch, which allowed meaningful comparisons between different configurations.

All code was written in PyTorch `v1.13.1+cu117` (Paszke et al., 2017) and `Transformers v4.30.2` (Wolf et al., 2019). The cross-entropy loss function was used for optimization, along with the Adam optimizer (Kingma and Ba, 2017) and a learning rate of 0.001. All other settings were kept at their default library implementations, except where specified in experiment configurations. In total, 546 models were trainded on NVIDIA A100 GPU with 16GB memory, totaling approximately 3,100 hours of training time. Model sizes ranged from 2.9 to 44.5 million parameters, primarily varying with embedding size and layer count, but also influenced by vocabulary size.

All data and code pertinent to the methods and results presented in this work will be made available at the time of the conference.

2.2.1 Triplets memorization

Three experimental setups on the dataset with triplets were devised to explore the models' behavior. For all of them, since the prediction of the related concept is based on unique combinations of concept and relation, it is straightforward to unambiguously determine whether a related concept was predicted correctly or not.

In the first setup, dataset sizes ranged in $\{50,000;\ 60,000;\ \ldots;\ 100,000\}$ samples. The model architecture consisted of a single transformer layer with an embedding size of 128, four attention heads, and a Rectified Linear Unit (ReLU) activation function (Agarap, 2019) with the batch size of 64, and 500 training epochs. This setup focused on evaluating memorization performance under a fixed architecture while varying dataset sizes.

The second experimental setup introduced variations in the transformer architecture, allowing a deeper investigation into the impact of model depth and activation functions. Dataset sizes included $50,000$, $70,000$, and $100,000$ samples, with the numbers of transformer layers set to 1, 2, or 4. Activation functions were varied across ReLU, Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel, 2023), Randomized Leaky Rectified Linear Unit (RReLU) (Xu et al., 2015), and Softmax (Boltzmann, 1868). To ensure fair comparisons, the total number of model parameters was kept constant across configurations by adjusting the embedding size (`d_model` parameter in PyTorch implementation of Transformers) proportionally to the number of layers, using the formula: $\texttt{embedding\_size} = \left\lfloor \frac{\texttt{base\_number\_of\_parameters}}{\texttt{n\_layers}} \right\rfloor$ with a base number of parameters of 128. This approach ensured that variations in performance could be attributed solely to architectural differences rather than changes in the total parameter count. For this setup, however, the batch size was increased to 128, and the number of training epochs was 1000, since it was required for achieving a plateau.

The third setup focused on evaluating the interplay between model depth, and embedding size while keeping other hyperparameters the same. Dataset sizes ranged in $\{1,000;\ 10,000;\ 50,000;\ 100,000\}$ samples. The architectural variations included transformer layers set to 1 or 2 and base numbers of parameters for embedding sizes in $\{16; 32; 64; 128\}$ (calculated as in the second experiment). Only the Softmax activation function and a fixed number of 4 attention heads were used. To ensure fair comparisons, configurations were designed to evaluate the impact of increasing embedding sizes and model depth on memorization performance. The total parameter count was recalculated for each configuration using the same formula as in the second experiment. For this setup, as previously, the batch size of 128 was used, and the number of training epochs was 500.

2.2.2 Sequences memorization

The dataset for sequence memorization tasks was prepared using the same tokenization process. However, to standardize sequence lengths, padding with zeros was applied at the end of each sequence, serving both as a filler and a marker for sequence termination. The task required distinguishing between nodes and edges, and a node mask was generated to identify the positions of node tokens within the sequence. It enabled the computation of metrics by isolating node positions during the training and evaluation processes. Notably, each node was predicted based on all preceding tokens in the sequence, meaning the last node in a sequence benefited from the most context. This setup provided deeper insights into the transformer model's ability to handle more structured data and its patterns.

The experimental setup was consistent with the previous experiments described in 2.2.1: the embedding size was fixed at 64, with four attention heads, the batch size was set to 128, and the number of epochs to 400. The number of layers varied across $\{1, 2, 4\}$, and the activation functions used were RReLU and Softmax. As before, the model incorporated a learned positional encoding. The dataset sizes were varied in $\{20,000;\ 50,000;\ 100,000\}$, representing the number of sequences. Each sequence was limited to 4-6 nodes (and 3-5 edges, respectively), selected randomly. During dataset construction, 5 hops were used to isolate the subgraph (see 2.1.3 for details).

For sequence memorization, accuracy and capacity were measured similarly to the triplet-based experiments, with slight adaptations to account for the sequential structure of the data. Accuracy was defined as the proportion of correctly predicted to-

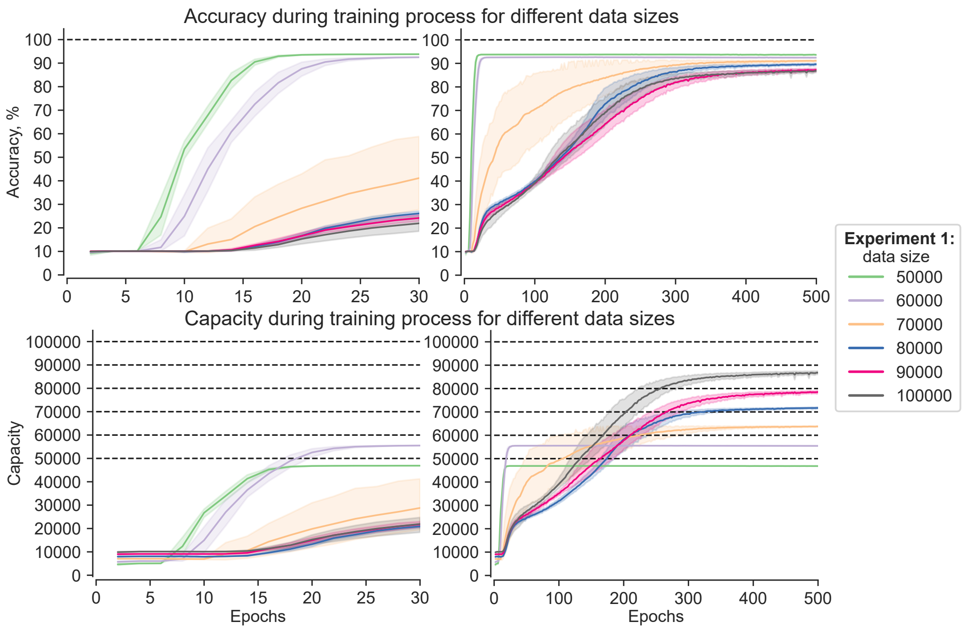

Figure 2: Trends in training accuracy (upper) and capacity (lower) for the first setup (different data sizes, for triplets dataset). Left: first 30 epochs; right: full training process of 500 epochs.

kens at node positions to the total number of node predictions in the dataset and are equal to all nodes across all sequences, excluding starting points. Total correct predictions also represent MAC.

3 Results

3.1 Dataset Size Influence

Figure 2 illustrates capacity and accuracy trends across dataset sizes in the first setup. Smaller datasets learn faster, with accuracy and capacity increasing rapidly within the first 5–6 epochs, reaching maximum capacity by epoch 20. In contrast, larger datasets show minimal improvement in the first 15 epochs but exhibit a later inflection point, leading to higher final accuracy and capacity. This suggests a threshold existence ($\sim$ 70,000 rows for this case), beyond which the training process changes and a lot more epochs are required for full memorization.

The final accuracy and capacity (Table 1) indicate that although smaller datasets initially achieve higher accuracy, their capacity remains well below the size of the dataset (e.g., 50,000 rows yield only 46,811 samples). In contrast, larger datasets, such as 100,000 rows, significantly improve memorization (86,776 samples), highlighting the model's ability to use more data. The progressive capacity increase suggests that dataset size plays a crucial role in optimizing memorization; however, the reasons behind the unlearned data, despite available capacity, remain unclear.

data size	accuracy, %	capacity
$50,000$	93.62 ± 0.3	$46,811 \pm 149$
$60,000$	92.42 ± 0.2	$55,455 \pm 126$
$70,000$	91.1 ± 1.08	$63,773 \pm 756$
$80,000$	89.63 ± 1.66	$71,706 \pm 1326$
$90,000$	87.24 ± 1.66	$78,517 \pm 2173$
$100,000$	86.78 ± 2.42	$86,776 \pm 2484$

Table 1: Final results after the full training process for the first setup (data sizes, for triplets dataset).

3.2 Architectural Variations Influences

In the second experimental setup, the batch size was increased from 64 to 128, since larger batch sizes seem reduce gradient noise and improve memorization. Consequently, the one-layer models in this setup converged faster and achieved higher capacity than in the first setup.

Softmax consistently outperformed other activation functions, yielding the highest average capacity, fewer outliers, and more stable training behavior. Notably, four-layer models with Softmax achieved capacities comparable to one- or two-layer models without sacrificing convergence speed (Figure 3), suggesting its scalability with depth.

In contrast, ReLU and RReLU showed moderate performance, but suffered from increased variability and decreased capacity as the layers increased,

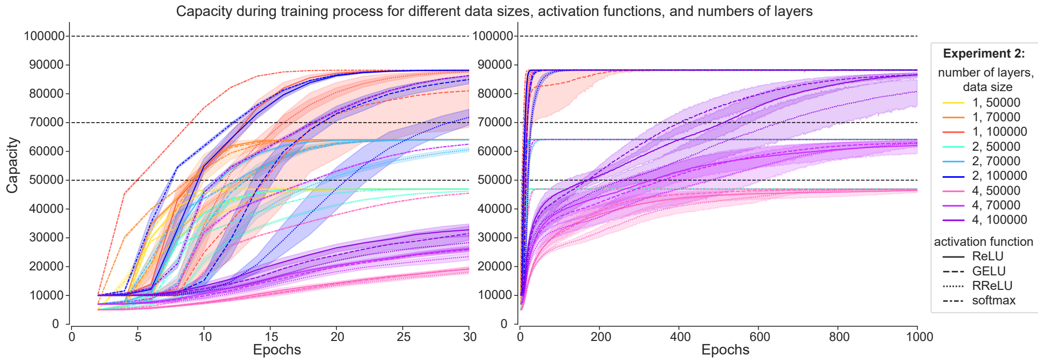

Figure 3: Trends in training capacity for the second setup (different data sizes, activation functions, and numbers of layers for triplets dataset). Left: first 30 epochs; right: full training process of 1000 epochs.

aligning with the findings of Paik and Choi (2023) and Chen and Ge (2024). These activations exhibited inconsistent learning patterns, with unexpected slowdowns in capacity improvements (Fu et al., 2024). GELU followed a similar trend, though it performed better in the early training stages with larger datasets.

As previously, the size of the dataset significantly affected training: larger datasets required longer warm-up phases, initially achieving lower capacities than smaller datasets under the same conditions. This suggests the existence of distinct learning phases where improvements depend on architectural depth, dataset size, and activation function.

Furthermore, adding more layers did not improve performance; instead, it slowed training and reduced final capacity. This is likely due to the simplicity of the dataset (triplets), where additional layers do not provide any advantage in capturing patterns. Although deeper architectures benefit more complex datasets (He et al., 2024), their impact may be reduced for data with simple relationships.

3.3 Numbers of Parameters influence

The third experiment further confirmed that, for simple datasets, learning dynamics depend primarily on embedding size, not the number of layers. Models with the same embedding size but different layer counts exhibited nearly identical accuracy improvement rates. For instance, as shown in Figure 4, a one-layer model with 16 parameters (embedding size is 16, light green) converged at almost the same rate as a two-layer transformer with 32 parameters (embedding size is 16 per layer, dark blue). Similar trends were observed for models with embedding sizes of 32 and 64, regardless of layer count.

These results highlight that embedding size is the key factor influencing learning speed, while adding layers without increasing embedding size neither accelerates convergence nor improves final capacity. In fact, additional layers often slow the training, as evidenced by the faster growth of accuracy of one-layer models (Figure 4). Smaller embedding sizes further reduced the learning speed, consistent with previous experiments. However, all configurations ultimately reached similar accuracy, highlighting that the simplicity of the dataset allows embedding size to dominate training dynamics.

Final capacity values remained nearly identical across configurations, regardless of embedding size or layer count: with a dataset size of $1,000$ samples, capacities for one- and two-layer models were near-accurate. Similarly, at $10,000$ and $50,000$ samples, one-layer models achieved $9,874 \pm 11$ and $46,939 \pm 105$, while two-layer models reached $9,875 \pm 7$ and $46,911 \pm 117$, respectively. However, at $100,000$ samples, a capacity "barrier" emerged. Two-layer transformers with an embedding size of 8 (16 total parameters) showed the capacity drop to $85,935 \pm 153$, compared to $\sim 88,200$ for other configurations, while one-layer models maintained a higher capacity of $88,240 \pm 62$. This suggests that larger datasets, smaller embeddings, and deeper architectures may introduce limitations due to slower convergence or suboptimal capacity utilization.

Evaluating consistency with the 2 bits per parameter rule (Allen-Zhu and Li, 2024b) was challenging due to dataset size limitations. While most configurations achieved similar capacities, the drop in the two-layer model with 16 parameters likely reflects incomplete convergence, possibly caused by slower learning dynamics in deeper models.

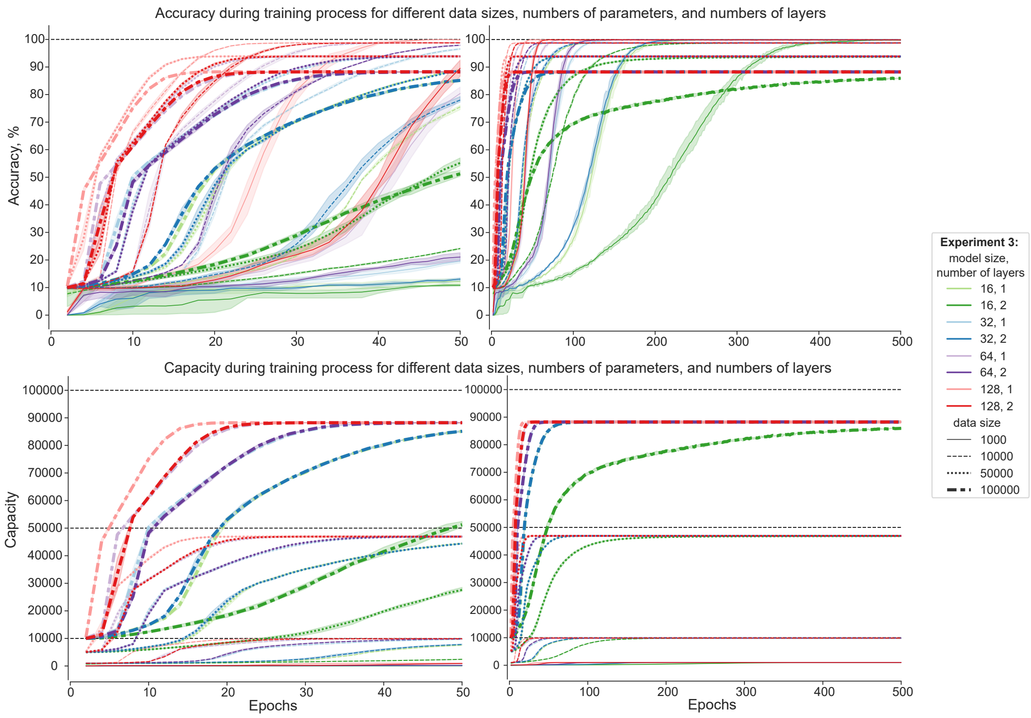

Figure 4: Trends in training accuracy (upper) and capacity (lower) for the third setup (different data sizes, numbers of parameters, and numbers of layers for triplets dataset). Left: first 50 epochs; right: full training process of 500 epochs. Light color corresponds to 1 layer, dark – to 2; number of parameters is a total number for all layers: green – 16, blue – 32, violet – 64, red – 128; embedding size can be computed by dividing it by layer count.

3.4 Insights from Sequence Datasets

In the fourth setup, model capacity was evaluated by testing its ability to memorize each node in a sequence using the full preceding sequence of nodes and edges (instead of triplets). This required multiple predictions per sequence: $34,908$, $85,972$, and $167,965$ for datasets containing 20, 50, and 100 thousand sequences, respectively.

Compared to triplet datasets, models trained on sequences achieved near-perfect memorization in significantly fewer epochs, with most configurations plateauing within 150 epochs (Figure 5). The sequential structure likely facilitated more efficient learning, though it also increased training time due to the higher information per sequence. Training exhibited greater capacity fluctuations across epochs, likely reflecting the dataset's increased complexity, as sequences encode more intricate patterns than triplets. Nonetheless, models demonstrated exceptional memorization, achieving 100% capacity for the 20 thousand sequence dataset and over 99.5% for 50 and 100 thousand sequences.

Regarding activation functions: as previously, RReLU converged more slowly than Softmax, though final capacities were nearly identical for one- and two-layer models: with 100 thousand sequences, RReLU achieved $166,934 \pm 243$ (one layer) and $166,995 \pm 118$ (two layers), while Softmax reached $166,992 \pm 110$ and $166,985 \pm 904$, respectively. In deeper models (4 layers), RReLU showed lower final capacities and greater fluctuations ($165,271 \pm 1,068$ vs. $166,825 \pm 319$ for Softmax). This contrasts with previous findings (Shen et al., 2023), which suggest ReLU outperforms Softmax. The discrepancy may indicate that activation function effectiveness may vary based on dataset structure and task, therefore it needs further investigation. Despite the increased complexity of sequence datasets, models adapted quickly and demonstrated strong memorization performance.

4 Discussion

This study provided insights into the memorization capacity of transformer models trained on real-world structured datasets. Returning to the medical domain of SNOMED, the original KG contains over a million relations, integrating diverse fields of medicine (e.g., substances, diseases, anatomical structures). However, in mobile applications, e.g. LLMs in smart glasses or smartwatches, models must efficiently retain only specific subsets of information. For instance, a cardiac surgeon's smart

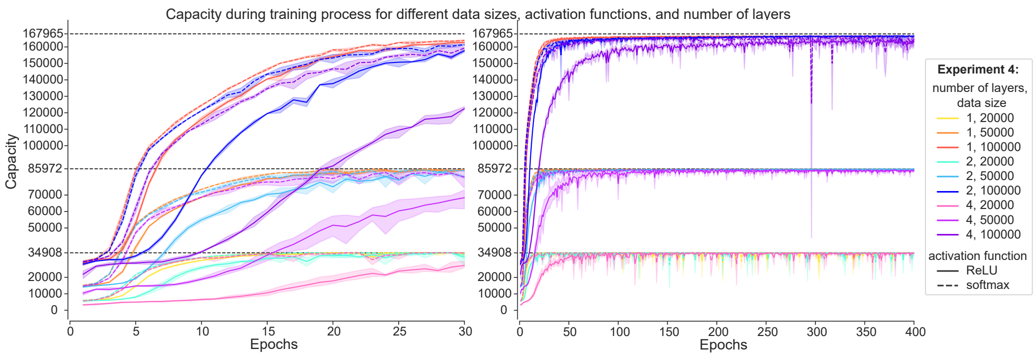

Figure 5: Trends in training capacity for the fourth setup (different data sizes, activation functions, and numbers of layers for sequences dataset). Left: first 30 epochs; right: full training process of 400 epochs.

glasses would require an LLM specialized in cardiology, while a smartwatch may store personalized health data for a specific user, limiting the dataset size to 10–100 thousand triplets or sequences.

This research offers insights into efficient training strategies for such models, analyzing how dataset characteristics and architectural choices impact convergence speed and capacity utilization.

4.1 Effect of Dataset Structure

Smaller datasets led to faster convergence but lower capacity, while larger datasets required longer warm-up periods but improved retention. Beyond a certain size, training slowed significantly, indicating optimization bottlenecks.

Sequence-based datasets outperformed triplets, achieving near-perfect memorization with fewer epochs. Sequences aided and complicated learning, reinforcing relationships between data but also introducing greater training fluctuations, aligning with Ju et al. (2021). This suggests that longer traversal sequences could further improve memorization in domain-specific medical applications.

4.2 Architectural Influence

Embedding size was the key factor in learning speed and capacity, while adding layers provided little benefit and sometimes reduced performance, likely due to dataset simplicity. This aligns with He et al. (2024), who found that many transformer layers exhibit high similarity and, sometimes, redundancy and can be pruned without performance loss, reducing computational overhead.

For larger datasets, smaller embeddings struggled to reach full capacity, particularly in deeper architectures, suggesting that increasing embedding size is more beneficial than adding depth, at least for structured, domain-specific memorization.

Softmax led to greater stability and capacity, while ReLU-based activations showed higher variability and performance drops in deeper models, aligning with Paik and Choi (2023); Chen and Ge (2024). However, this contrasts with prior work by Shen et al. (2023), emphasizing that activation effectiveness is highly dependent on the task and dataset structure.

5 Conclusions

This study explored the memorization capacity of transformer models on structured datasets from SNOMED KG, analyzing how architecture and dataset structure affect learning efficiency and capacity retention.

Key findings show that embedding size and activation function were more influential than depth, while larger datasets improved memorization but required longer training. Triplets performed well in simpler models, whereas sequences excelled but introduced fluctuations. Challenges remain in efficiency, layer-specific contributions, and generalization, necessitating further research on scalability, compression, and architecture optimization.

For real-world applications, such as LLMs in medical smart devices, models must efficiently store specialized knowledge while maintaining computational feasibility. Future work should explore longer sequences, adaptive memory compression, and layer-wise analysis to enhance structured knowledge retention in practical deployments.

6 Limitations

While this study provides meaningful insights, several open questions remain:

- It is unclear why certain samples remain unlearned within the same model architecture despite available capacity. Future research should explore optimization strategies to improve memorization efficiency.

- Future research should test these findings and hypotheses on longer sequences and larger datasets to confirm them at scale.
- The specific role of each layer in memorization was not investigated, missing insights from probing methods as suggested in Allen-Zhu and Li (2024a). Future studies could apply probing techniques to analyze layer-specific role in memorization capacity.
- Additionally, sparse autoencoders (Bricken et al., 2023) or transcoders (Paulo et al., 2025) could be integrated into transformer layers to distinguish memorization from generalization, helping determine whether certain layers store specific relationships or contribute to broader model generalizability.

By addressing these limitations, future work can further refine transformer optimization strategies for structured data modeling and knowledge retention.

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024a. Physics of language models: Part 3.1, knowledge storage and extraction. *Preprint*, arXiv:2309.14316.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024b. Physics of language models: Part 3.3, knowledge capacity scaling laws. *Preprint*, arXiv:2404.05405.

Simone Balloccu, Ehud Reiter, Vivek Kumar, Diego Reforgiato Recupero, and Daniele Riboni. 2024. Ask the experts: sourcing high-quality datasets for nutritional counselling through Human-AI collaboration. *arXiv preprint*. ArXiv:2401.08420 [cs].

Ludwig Boltzmann. 1868. Studien über das gleichgewicht der lebendigen kraft zwischen bewegten materiellen punkten. *Wiener Berichte*, 58:517–560. Studies on the balance of living force between moving material points.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zach Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Wenlin Chen and Hong Ge. 2024. Neural characteristic activation analysis and geometric parameterization for relu networks. *Preprint*, arXiv:2305.15912.

Shaker El-Sappagh, Francesco Franda, Farman Ali, and Kyung-Sup Kwak. 2018. Snomed ct standard ontology based on the ontology for general medical science. *BMC Medical Informatics and Decision Making*, 18(1):76.

Jingwen Fu, Tao Yang, Yuwang Wang, Yan Lu, and Nanning Zheng. 2024. Breaking through the learning plateaus of in-context learning in transformer. *Preprint*, arXiv:2309.06054.

Bhumika Gupta, Pralaypati Ta, Keerthi Ram, and Mohanasankar Sivaprakasam. 2024. Comprehensive Modeling and Question Answering of Cancer Clinical Practice Guidelines using LLMs. In *2024 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. ISSN: 2994-9408.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024. What matters in transformers? not all attention is needed. *Preprint*, arXiv:2406.15786.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus). *Preprint*, arXiv:1606.08415.

Aki Härmä, Marcin Pietrasik, and Anna Wilbik. 2024. Empirical capacity model for self-attention neural networks. *Preprint*, arXiv:2407.15425.

Yue Ju, Alka Isac, and Yimin Nie. 2021. Chunkformer: Learning long time series with multi-stage chunked transformer. *Preprint*, arXiv:2112.15087.

Tokio Kajitsuka and Issei Sato. 2024. Optimal memorization capacity of transformers. *Preprint*, arXiv:2409.17677.

Junghwan Kim, Michelle Kim, and Barzan Mozafari. 2023. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Jean-Baptiste Lamy. 2017. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80:11–28.

Sadegh Mahdavi, Renjie Liao, and Christos Thrampoulidis. 2024. Memorization capacity of multi-head attention in transformers. *Preprint*, arXiv:2306.02010.

Inyoung Paik and Jaesik Choi. 2023. The disharmony between bn and relu causes gradient explosion, but is offset by the correlation between activations. *Preprint*, arXiv:2304.11692.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Gonçalo Paulo, Stepan Shabalin, and Nora Belrose. 2025. Transcoders beat sparse autoencoders for interpretability. *Preprint*, arXiv:2501.18823.

Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. 2024. Graph transformers: A survey. *Preprint*, arXiv:2407.09777.

Kai Shen, Junliang Guo, Xu Tan, Siliang Tang, Rui Wang, and Jiang Bian. 2023. A study on relu and softmax in transformer. *Preprint*, arXiv:2302.06461.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Jinlin Wu, Xusheng Liang, Xuexue Bai, and Zhen Chen. 2024. SurgBox: Agent-Driven Operating Room Sandbox with Surgery Copilot. *arXiv preprint*. ArXiv:2412.05187 [cs].

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *Preprint*, arXiv:1505.00853.

A Appendix: Additional Representations of the Results

This appendix provides supplementary visualizations and tables for the experiments conducted:

- Second experiment:
  - Figure 6: Accuracy trends during training.
  - Table 2: Final capacities.
- Third experiment:
  - Table 3: Final capacities.
- Fourth experiment:
  - Figure 7: Accuracy trends during training.
  - Table 4: Final capacities.

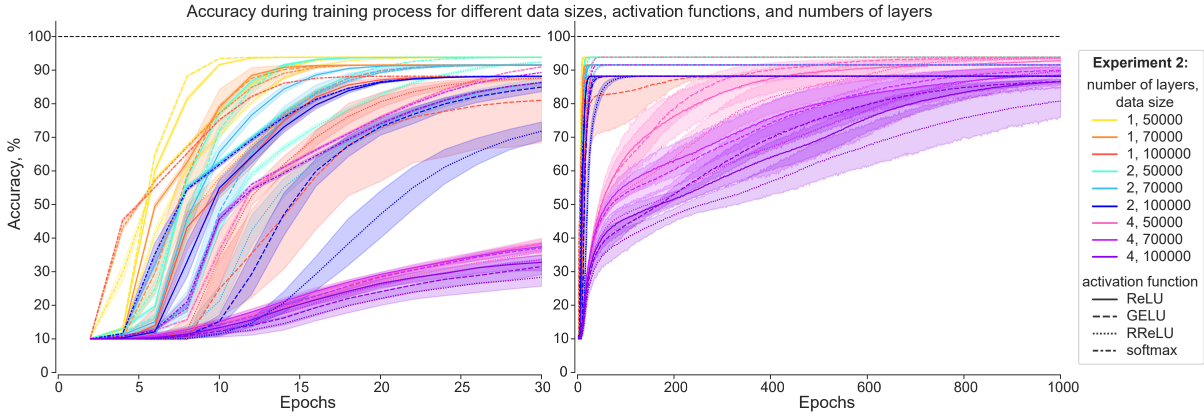

Figure 6: Trends in training accuracy for the second setup (different data sizes, activation functions, and numbers of layers for triplets dataset). Left: first 30 epochs; right: full training process of 1000 epochs.

activation function	**layers count**	**data sizes**		
		$50,000$	$70,000$	$100,000$
ReLU	1	$46,898 \pm 158$	$64,091 \pm 192$	$88,148 \pm 312$
	2	$46,920 \pm 112$	$64,086 \pm 130$	$88,217 \pm 125$
	4	$46,391 \pm 2,268$	$61,931 \pm 8,480$	$86,558 \pm 3,291$
GELU	1	$46,925 \pm 105$	$64,096 \pm 184$	$88,195 \pm 123$
	2	$46,926 \pm 115$	$64,080 \pm 120$	$88,215 \pm 128$
	4	$46,798 \pm 156$	$62,949 \pm 1,906$	$86,589 \pm 2,202$
RReLU	1	$46,930 \pm 125$	$64,080 \pm 122$	$88,180 \pm 180$
	2	$46,927 \pm 121$	$64,088 \pm 117$	$88,208 \pm 132$
	4	$46,730 \pm 223$	$62,818 \pm 3,680$	$80,755 \pm 15,844$
softmax	1	$46,924 \pm 87$	$64,082 \pm 166$	88211 ± 192
	2	$46,908 \pm 127$	$64,074 \pm 134$	$88,213 \pm 171$
	4	$46,923 \pm 104$	$64,085 \pm 131$	$88,197 \pm 134$
all	1	$46,919 \pm 119$	$64,087 \pm 162$	$88,183 \pm 210$
	2	$46,920 \pm 115$	$64,082 \pm 121$	$88,213 \pm 135$
	4	$46,710 \pm 1169$	$62,945 \pm 4,92$	$85,525 \pm 9,720$

Table 2: Final capacity after the full training process for the second setup (different numbers of layers, data sizes, and activation functions for triplets dataset).

embedding parameters	**layers count**	**data sizes**			
		$1,000$	$10,000$	$50,000$	$100,000$
16	**1**	$1,000 \pm 1$	$9,870 \pm 10$	$46,937 \pm 148$	$88,236 \pm 74$
	2	998 ± 3	$9,875 \pm 4$	$46,858 \pm 93$	$85,935 \pm 153$
32	**1**	998 ± 3	$9,872 \pm 11$	$46,955 \pm 119$	$88,234 \pm 62$
	2	999 ± 3	$9,876 \pm 9$	$46,927 \pm 128$	$88,252 \pm 82$
64	**1**	999 ± 2	$9,878 \pm 9$	$46,932 \pm 122$	$88,242 \pm 102$
	2	999 ± 3	$9,876 \pm 7$	$46,919 \pm 96$	$88,237 \pm 58$
128	**1**	999 ± 2	$9,877 \pm 12$	$46,930 \pm 85$	$88,248 \pm 29$
	2	999 ± 3	$9,872 \pm 6$	$46,938 \pm 131$	$88,214 \pm 53$
all	**1**	999 ± 2	$9,874 \pm 11$	$46,939 \pm 105$	$88,240 \pm 62$
	2	999 ± 3	$9,875 \pm 7$	$46,911 \pm 117$	$87,660 \pm 2,082$

Table 3: Final capacity after the full training process for the third setup (different data sizes, numbers of parameters, and numbers of layers for triplets dataset).

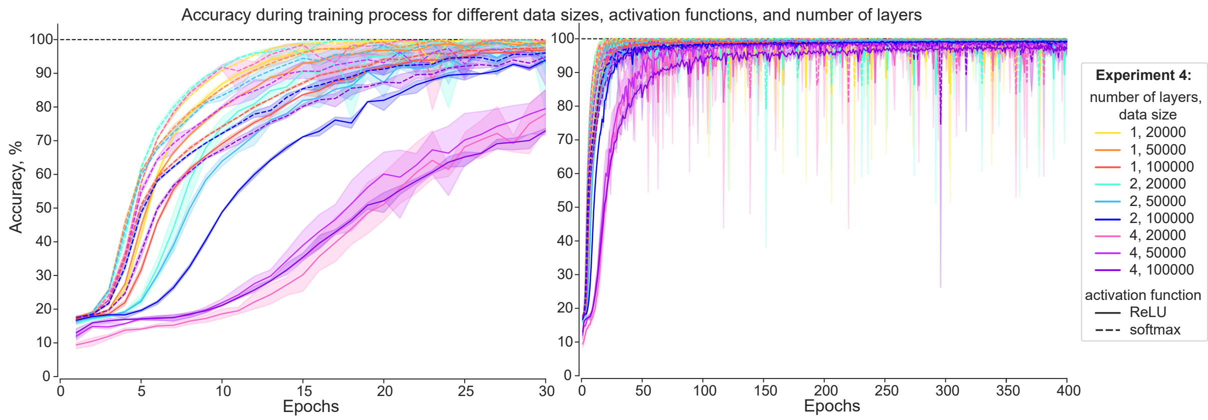

Figure 7: Trends in training accuracy for the fourth setup (different data sizes, activation functions, and numbers of layers for sequences dataset). Left: first 30 epochs; right: full training process of 400 epochs.

activation function	**layers count**	**# of sequences (# of predictions)**		
		$20,000$ $(34,908)$	$50,000$ $(85,972)$	$100,000$ $(167,965)$
RReLU	**1**	$34,908 \pm 0$	$85,936 \pm 31$	$166,934 \pm 243$
	2	$34,908 \pm 0$	$85,917 \pm 34$	$166,995 \pm 118$
	4	$34,908 \pm 0$	$85,647 \pm 270$	$165,271 \pm 1,068$
softmax	**1**	$34,908 \pm 0$	$85,931 \pm 18$	$166,992 \pm 110$
	2	$34,908 \pm 0$	$85,888 \pm 33$	$166,985 \pm 904$
	4	$34,908 \pm 0$	$85,771 \pm 42$	$166,825 \pm 319$
all	**1**	$34,908 \pm 0$	$85,934 \pm 23$	$166,963 \pm 180$
	2	$34,908 \pm 0$	$85,903 \pm 44$	$166,990 \pm 577$
	4	$34,908 \pm 0$	$85,709 \pm 220$	$166,048 \pm 1,842$

Table 4: Final capacity after the full training process for the fourth setup (different data sizes, activation functions, and numbers of layers for sequences dataset).