Understanding transformer-based language models through n-gram statistics

Bjarni Haukur
bhbj@kth.se

Jonas Rosengren
jonrose@kth.se

Botond Lovász
botondl@kth.se

Felix Krings
fkrings@kth.se

Abstract

Motivated by the need to demystify transformer-based language models, this study evaluates how effectively n-gram statistics can model transformer predictions. By reimplementing Nguyen's framework, we examined the extent to which n-gram rules approximate transformer behavior across datasets of varying complexity, including TinyStories, Wikipedia, and StrangeTextbooks. Results indicate that as datasets become more complex, it becomes increasingly difficult to explain transformer predictions using simple statistical rules, while simpler datasets rely more on local patterns that are easier to model. For simpler datasets like TinyStories, transformer predictions align closely with 6-gram rules, validating their applicability in modeling straightforward statistical patterns. However, as dataset complexity increases, n-gram models fail to capture the nuanced contextual dependencies leveraged by transformers, highlighting a shift from reliance on simple rules to deeper contextual understanding. The analysis also reveals a direct correlation between the complexity of n-gram rules and prediction accuracy, providing insights into the progression of language model training—starting with simpler statistical patterns and evolving toward mastering intricate contextual relationships. These findings offer valuable perspectives on the dynamics of transformer learning and the limitations of rule-based approximations in complex scenarios.

1 Introduction

Transformer-based large language models (LLMs) have achieved state-of-the-art performance across a wide range of Natural Language Processing (NLP) tasks. Despite their impressive capabilities, a precise understanding of how these models models learn and make use of their context remains elusive. A promising direction in understanding these models is to approximate their behavior through simpler, interpretable frameworks. In particular, recent work by Nguyen [21] has proposed that the next-token predictions of transformers can be approximated by n-gram models—simple statistical rules derived from the training data. This idea builds on earlier findings by numerous authers including Roy et al. [26] and Qi et al. [23], which had already indicated a similar pattern.

Nguyen's work introduced a novel overfitting detection criterion that does not require a holdout set, addressing limitations in conventional validation approaches. By evaluating predictions against shortened contexts, overfitting is identified when predictions on partial contexts diverge from those on the full context. This serves as a stopping criterion, relying solely on the structure within the training set. Additionally, Nguyen's study quantified the alignment between transformer predictions and n-gram statistics, though the lack of publicly available code limits the reproducibility and broader application of these findings. To address this gap, we revisited Nguyen's approach and re-implemented the approach from scratch. Our objective was to replicate and expand upon the original results by applying them to a broader set of datasets. This approach aimed to provide deeper

DD2430 Project in Data Science

insights into how transformer models pick up the statistical patterns of their training data by their relation to n-gram statistics.

We compared transformer-based model predictions with n-gram models (1-7 grams), experimenting with smoothing techniques [14] to assess their predictive accuracy.

The structure of this report is as follows: We first review the related work in Section 2, focusing on the body of research that informs our approach regarding transformers and n-gram statistics. Section 3 presents the problem description, laying out the core questions we seek to answer. Section 4 details the methodology and implementation, including how we built and trained the transformer model and constructed efficient n-gram models. Section 5 offers an evaluation of our experimental results, comparing the performance of transformer and n-gram models. We conclude with a discussion of the implications of our findings in Section 6, followed by a summary of our discussions with the opposing group in Section 7, which highlights differing perspectives and insights gained through collaboration.

2 Related Work

Transformers are advanced neural network architectures optimized for sequence transduction tasks, relying solely on self-attention mechanisms instead of recurrence or convolution. At their core, they implement multi-head self-attention to compute relationships across all positions in an input sequence, enabling global context modeling with $O(1)$ path lengths for dependencies, irrespective of their distance. This is achieved via the scaled dot-product attention mechanism, where query-key compatibility determines weighted value summation. Transformers integrate positional encodings to preserve sequential order and employ feed-forward networks at each layer for non-linear transformations. Their highly parallelizable structure not only accelerates training but also supports efficient scaling for large datasets, making them state-of-the-art in natural language processing tasks like machine translation and text generation [27], [11].

Rule extraction from neural networks has been explored in a variety of ways, often focusing on simplifying the internal processes of these networks to make them more interpretable. Early work by Fukumi and Akamatsu [7] introduced a random optimization method (ROM) to reduce connection weights in neural networks, enabling the extraction of simpler logical functions. This approach was further developed by Fukumi et al. in 2001 [8], who integrated if-then rules into neural networks and used structure learning to generate compact network architectures, allowing for easier rule extraction. These studies emphasize reducing network complexity for better interpretability, particularly in classification tasks, echoing earlier efforts to create symbolic representations from neural networks.

More recent work in rule extraction has shifted towards applying these methods in complex architectures, such as Convolutional Neural Networks (CNNs). Bologna [3] presents a method for extracting symbolic rules from CNNs used in sentiment analysis. This approach leverages Discretized Interpretable Multi Layer Perceptrons (DIMLP) to represent the rules in an interpretable form, demonstrating better fidelity and classification accuracy than traditional methods like support vector machines and decision trees. Similarly, Waldis et al. [28] trained CNNs to identify concepts from N-grams in text data, achieving high precision by combining CNN-based feature extraction with a TF-IDF ranking mechanism. These methods have successfully enhanced the interpretability of CNNs in domain-specific tasks, such as sentiment analysis and concept extraction.

The problem of rule extraction from recurrent neural networks (RNNs) has been approached from a different angle. Jacobsson [12] examined RNNs by modeling their behavior as finite state machines, allowing for a deeper analysis of their dynamic processes. This approach offers insights into the underlying structure of RNNs, moving beyond traditional performance metrics to provide a more transparent understanding of their inner workings. Earlier, McMillan et al. [17] introduced the concept of extracting symbolic condition-action rules from networks using the Connectionist Scientist Game. This method focuses on inducing explicit symbolic rules from connectionist systems, with applications to tasks such as string-to-string mapping.

In addition to rule extraction, N-gram models have been widely studied in the context of natural language processing and language models. Early work by Kneser and Ney [14] introduced optimized backing-off strategies for N-gram language models, improving their perplexity and word error rates. Goodman [10] further explored advanced N-gram modeling techniques, including smoothing and clustering, to significantly reduce perplexity and enhance language modeling performance.

More recently, the relevance of N-gram models has been revisited in the era of large language models. Liu et al. [15] introduced the “infini-gram” model, which computes N-gram probabilities at an unprecedented scale, enabling the largest N-gram models ever built. This modern approach shows how N-grams can still play a crucial role in complementing neural LLMs by improving their perplexity and offering insights into machine-generated text. Akyürek et al. [2] also examined N-gram processing in the context of in-context learning, where large language models utilize specialized "N-gram heads" to predict token sequences based on input data. These developments highlight the continuing importance of N-grams in both traditional and contemporary language modeling frameworks.

This work collectively demonstrates the evolution of rule extraction and N-gram modeling, from early neural network methods to their integration into more advanced architectures like CNNs and LLMs. The methods used across these studies show a consistent focus on enhancing interpretability and efficiency, whether through simplification of network architectures, symbolic rule extraction, or the scaling of N-gram models.

3 Problem Description

Transformer-based language models have become the cornerstone of modern Natural Language Processing (NLP) tasks, achieving remarkable success in various applications such as translation, summarization, and content generation. Despite these advancements, a comprehensive understanding of transformers' internal mechanisms remains elusive. Instead, our focus is on evaluating their predictions against the training data. Prior research, such as Nguyen's work, suggests that transformers rely on statistical patterns from the training dataset, similar to n-gram models, to predict token distributions and outputs [21]. This hypothesis presents an intriguing opportunity to bridge the gap between complex model behavior and more interpretable, simpler models like n-grams. By investigating the extent to which transformers can be described through n-gram rules, this project aims to enhance our understanding of how transformers process and generate language.

This project replicates the fundamental assumption of Nguyen's study and extends its scope by comparing the performance of a transformer-based language model to various n-gram models (n=1 to 7). The core question revolves around understanding how much of the transformer's predictions can be attributed to simple n-gram statistics. Specifically, the project investigates the following:

1. Validate how well can n-gram rules model transformer predictions?

2. How does the complexity of n-gram rules correlate with model behavior, particularly in terms of prediction accuracy?

3. What insights can be gained into the dynamics of language model training, particularly the transition from learning simple patterns to more complex ones?

This project involves training transformer models on multiple heterogeneous datasets. Namely TinyStories [5], Wikipedia [6] and StrangeTextbooks [20] dataset, relating them to n-gram statistics computed for each individual dataset, and comparing the performance of both models. Additionally, it will extend the investigation by evaluating the relevance of different smoothing techniques and alternative rule-based models to better understand the underlying mechanisms at play.

Relevance to Explainable AI (XAI)
Explainability in AI is a critical concern, especially for large-scale models like transformers, which often function as black boxes. Understanding how these models utilize dataset statistics is crucial for developing more interpretable and transparent AI systems. By comparing transformers to n-gram models, this project contributes to the field of XAI, offering insights into when and why model predictions align with simpler statistical rules. This understanding is key to improving model transparency, especially in contexts where model decisions need to be trusted by human users.

Contribution
This project will replicate and extend existing work in the field, providing practical experience in language modeling while exploring new avenues for understanding transformer behavior. The results will contribute to ongoing discussions about the balance between complexity and interpretability in AI systems, paving the way for future research into more transparent and trustworthy language models.

4 Methodology and Implementation

4.1 Experimental Setup

To execute our experiments, we utilized NVIDIA H100 GPUs with 80GB HBM3 memory, leveraging their computational efficiency to handle the extensive data and processing demands. Training was conducted using the LLama transformer model architecture, specifically the *LLamaForCausalLM* implementation from Hugging Face [1]. For tokenization, a Byte Pair Encoding (BPE) tokenizer was employed, maintaining consistency with the model's vocabulary. Details regarding the transformer configuration can be found in Appendix A.1. The choice of a learning rate with linear warm-up followed by cosine decay ensured stable optimization during the training process.

This experimental setup underpins the subsequent methodology, enabling the evaluation of transformer behavior in diverse contexts (see Section 4.4) and forming the basis for comparisons with n-gram models (elaborated in Section 4.5). Moreover, real-time parameter logging facilitated iterative refinements, aligning our approach with prior experimental frameworks.

4.2 Datasets

We employed a set of heterogeneous datasets to test our hypotheses. For fairness, 1GB partitions of each dataset was used. A.2 shows N-gram characteristics for each dataset, such as nodes and branching factors.

The first, TinyStories [5], consists of 480 million tokens representing children's stories synthetically generated from GPT-4 using templated prompts. This dataset provides a linguistically straightforward foundation, allowing us to explore n-gram dependencies and evaluate their predictive alignment with transformer models (see Section 5 for results).

The second dataset, TinyTextbooks [19], is a collection of synthetically generated educational material. It introduces greater structural complexity, thereby challenging the models to adapt to nuanced contextual dependencies. Section 5 also discusses the differing outcomes across these datasets, highlighting the transition from statistical reliance to deeper contextual understanding.

The Tiny StrangeTextbooks dataset [20] comprises 2.7 million AI-generated synthetic textbooks, encompassing 16GB of structured raw text data across diverse subjects. These compact yet information-dense documents are designed to emulate high-quality, efficient training data, as outlined in the foundational papers "Textbooks Are All You Need" and its sequel. Each textbook in the dataset represents a tightly focused exploration of specific topics, making the collection ideal for experiments requiring structured and clean textual inputs. This dataset challenges models with its diverse subject matter and offers insights into how well n-gram methods capture structured language representations compared to transformer-based approaches.

The Wikipedia dataset [6] includes cleaned and parsed articles in multiple languages, sourced from Wikimedia's comprehensive database. For our experiments, we utilized two distinct subsets: one in English and the other in Hungarian, with the latter filtered using a parser tool. The Hungarian subset serves as a testbed for evaluating model robustness in handling morphologically complex languages, contrasting with the relatively simpler English subset. Each article in the dataset represents a complete, cleaned document, stripped of markup and extraneous sections, ensuring high-quality input. The dataset's multilingual structure and diversity provide a fertile ground for analyzing contextual nuances and linguistic complexities.

4.3 *N*-gram statistics

N-grams are fundamental constructs in computational linguistics and natural language processing, representing contiguous sequences of n items, typically words or characters, from a given text. The concept of n-grams has been extensively applied in language modeling, information retrieval, and machine translation, among other fields.

Definition and Basic Concept
Formally, an n-gram is defined as a subsequence of n items from a larger sequence. For a given text $T = w_1 w_2 \dots w_m$, where w_i are individual tokens (e.g., words or characters), an n-gram is a sequence $w_i, w_{i+1}, \dots, w_{i+n-1}$ for $i = 1, 2, \dots, m - n + 1$ [16].

The statistical representation of n-grams is essential for estimating the probability distribution of sequences. Under the n-gram assumption, the conditional probability of a token w_t, given its

preceding context $w_1 w_2 \dots w_{t-1}$, is approximated as:

$$P(w_t|w_1 w_2 \dots w_{t-1}) \approx P(w_t|w_{t-n+1} \dots w_{t-1}).$$

This simplification reduces the dimensionality of the problem and enables practical computation [25].

Mathematical Foundation
The n-gram probability estimation is typically based on frequency counts in a corpus. For an n-gram $(x_{t-n+1}, \dots, x_t)$, the maximum likelihood estimation (MLE) is given by:

$$P(x_t|x_{t-n+1} \dots x_{t-1}) = \frac{\text{count}(x_{t-n+1} \dots x_t)}{\text{count}(x_{t-n+1} \dots x_{t-1})}.$$

This formulation assumes a finite dataset where probabilities are normalized over observed sequences [16].

Directly using Maximum Likelihood Estimation (MLE) in n-gram models can lead to overfitting, as it assigns zero probability to unseen n-grams. This limitation is particularly problematic for unseen tokens, which frequently occur in real-world applications. To address this, smoothing techniques are applied to ensure nonzero probabilities for all n-grams and better handle data sparsity.

Add-λ smoothing is a basic approach where a small constant λ is added to all observed counts:

$$P_{\text{smoothed}}(x_t|x_{t-n+1} \dots x_{t-1}) = \frac{\text{count}(x_{t-n+1} \dots x_t) + \lambda}{\text{count}(x_{t-n+1} \dots x_{t-1}) + \lambda \cdot |V|},$$

where $|V|$ is the vocabulary size [9]. While effective in addressing the zero-probability issue, this method may not fully capture linguistic patterns or efficiently redistribute probability mass.

To further enhance the model's capability to reflect real-world linguistic dynamics, we incorporated Kneser-Ney smoothing [14]. This advanced method redistributes probability mass from observed events to unseen ones, better handling data sparsity and improving performance. Kneser-Ney smoothing integrates seamlessly with our trie-based architecture, as detailed in Section 4.5, where we introduce advanced rule-based probability estimation and computational optimizations for efficient implementation.

By employing these smoothing techniques, the model overcomes the limitations of unsmoothed probabilities, enabling a more robust representation of token probabilities within the n-gram framework.

N-grams are extensively used in predictive models like language modeling. For instance, in speech recognition and machine translation, n-grams help predict the likelihood of word sequences, optimizing tasks such as decoding and hypothesis ranking [22]. Nevertheless, challenges such as data sparsity and computational scalability remain significant. Recent advancements integrate n-grams with neural network models to leverage their respective strengths [25, 16].

To enable an efficient computation of n-gram statistics, we implemented a high-performance Rust package tailored for this purpose. This package computes token probabilities based on n-gram rules and context, further categorized into suffix, subgram, and all-rules evaluations, as inspired by Nguyen's original framework. Metrics such as variational distance and top-1 accuracy were assessed on validation batches multiple times per epoch, ensuring comprehensive evaluation [4]. These metrics were critical in evaluating the alignment between n-gram predictions and those of transformers, as discussed in Section 5.

The flexibility of the n-gram model lies in its ability to define context-token relationships through rules. This rule-based structure resembles how attention is drawn to different tokens in the context. The N-gram model computes the probability distribution for the next token, conditioned on the specified context and rule.

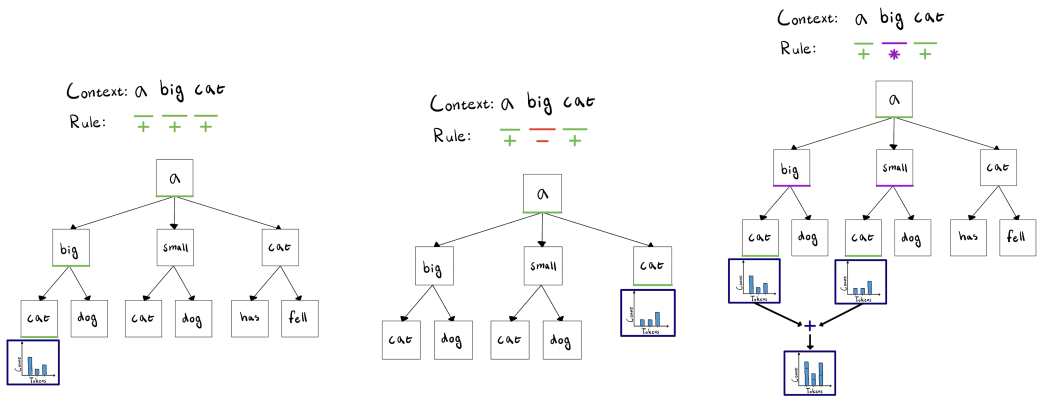
The rule symbols denote specific operations:

- $+$: *keep*s the token from the context
- $-$: *remove*s the token
- $*$: serves as a *wildcard* for any token

How these rules are applied to the context of the tries are illustrated in Figure 1. In particular the derivation of token counts from context-rules is illustrated. These counts can then be transformed into

the next token probabilities. For instance, given a context such as [a, big, cat] and a rule $[+, *, +]$, all paths in the trie with "a" as the first token and "cat" as the third token is matched. This particular example is illustrated in Figure 1c. The counts for the next tokens for each path is then added together to create a joint distribution for the next token prediction.

Section 4.4 further elaborates on the theoretical underpinnings of these rule-based next token predictions, while Section 5 evaluates their predictive fidelity against transformer outputs.



(a) Keep (b) Remove (c) Marginalize

Figure 1: Illustrating how the rules are applied with a context on a trie to give next token counts. The matching paths in the tries are marked, with green indicating an exact match and purple being a wildcard. 1a) shows an exact match of the entire context. 1b) shows an exact match after removing the second token. 1c shows a marginalization, matching all paths that has "a * cat". The resulting next token counts of marginalization is the joint distribution of all matching paths.
The tries are simplified for brevity and clarity. All nodes have next token counts and all trie paths have the same length.

One hypothesis proposed by Nguyen is that the introduction of Kneser-Ney smoothing enhances the generalization capability of n-gram predictions, potentially aligning them more closely with transformer distributions, particularly in datasets with complex structures. However, while this smoothing method appears to improve generalization, the extent to which it significantly impacts the alignment with transformer distributions has not been explicitly evaluated in our work.

By addressing the limitations of *unsmoothed* probabilities, this smoothing technique ensures that low-frequency events are assigned appropriate probabilities, facilitating a more realistic model of linguistic phenomena. Section 5 explores the comparative performance of smoothed and unsmoothed n-grams in terms of variational distance and top-1 accuracy, underscoring the necessity of smoothing for improving n-gram-based predictive models.

4.4 Methodology and Approach

Nguyen's approach to analyzing transformer-based language models centers on understanding if model predictions can be explained by statistical rules derived from N-gram statistics in training data. This method aims to evaluate whether transformer models rely more on local contextual patterns in the data rather than on a broader, global understanding, particularly in next-token prediction. The motivation behind Nguyen's work is to demystify the prediction process of transformer models by examining their behavior through statistical patterns found within training data, offering a clearer view of how much these models depend on literal dataset-driven structures as opposed to high-level contextual interpretation.

Nguyen develops a rule-based framework that attempts to emulate the attention mechanism of transformers by modifying token sequences. By retaining, discarding, or marginalizing tokens, the framework approximates how transformers might prioritize contextual elements. Comparing these rule-generated predictions with transformer outputs provides insights into transformer behavior.

In implementing this framework, Nguyen designs increasingly complex rulesets to better understand the nuances of model predictions. Suffix rules use consecutive tokens up to a specific length, subgram rules draw on token subsets with flexible attention to individual tokens, and the all-encompassing "all

rules" category permits a wide range of token combinations. The optimal ruleset is selected by finding the rule that most closely matches the transformer's predictions, minimizing the distance between the model output and the rule-generated prediction. Nguyen's analysis reveals a close association between the complexity of these rules and the variance in the model's predictions, showing that predictions with low variance are often well-approximated by simpler, more stable N-gram rules. This observation suggests that stable model predictions may stem from regular patterns in the dataset rather than more intricate contextual reasoning.

A key aspect of Nguyen's work is his exploration of curriculum learning dynamics within transformer models. He notes that during training, transformers appear to adopt a curriculum-like learning process, starting with simpler rules before moving on to more complex dependencies. In simple datasets like TinyStories, this progression aligns well with the model's training path. However, in our reimplementation, we found that this curriculum pattern did not hold when applied to more complex data such as StrangeTextbooks [20]. This divergence from Nguyen's findings suggests that while curriculum learning may apply in simpler, highly structured datasets, it may not be as appropriate for datasets that demand more nuanced contextual understanding. Section 5 continues exploring this topic.

In his implementation, Nguyen uses TinyStories [5] as the primary dataset, with Wikipedia data [24] serving as a supplementary source for testing scalability. TinyStories, a synthetically generated collection of simple children's stories, provides a relatively small (480 million tokens) and linguistically straightforward dataset that allows for efficient analysis of N-gram dependencies without excessive computational demands. Wikipedia, drawn from the MassiveText corpus, validates Nguyen's findings at a larger scale. Both datasets are processed into 2048-token sequences, shuffled to reduce fixed order effects and to help minimize training biases. Nguyen's experimental setup involves training two main transformer models: a 160-million-parameter model for the primary TinyStories experiments and a larger 1.4-billion-parameter model used in overfitting studies. Both models employ a cosine-decay learning rate with warmup and use a weighted Adam optimizer with regularization. Document boundaries are indicated by a <*BOS*> token, and chunking is employed to limit cross-document influence in N-gram statistics calculation.

While our implementation follows Nguyen's structure, several significant differences distinguish our approach. We deliberately did not implement Nguyen's overfitting detection method, as our focus did not include detecting or analyzing overfitting. This decision was influenced by two factors: first, the trends underlying Nguyen's method were not observed in the datasets we worked with; second, as this is a small-scale school project, it was neither practical nor necessary to replicate every aspect of the original study. Moreover, our findings on curriculum learning differ from Nguyen's conclusions; where he observed curriculum dynamics in TinyStories, we found that this pattern did not extend to the more complex datasets. This discrepancy suggests that curriculum learning may not be universally applicable across datasets with varying structural complexities.

In sum, our reimplementation of Nguyen's approach closely follows his methodology but introduces important variations in focus and findings. By omitting overfitting detection and reassessing curriculum learning dynamics, we adapted Nguyen's framework to address both the strengths and limitations of N-gram-based rule approximations. This project not only verifies the applicability of N-gram rules in simpler contexts but also tests the boundaries of this approach in more complex datasets, advancing our understanding of the balance between statistical regularity and deeper contextual dependence in transformer behavior.

4.5 N-Gram Implementation

The n-gram server began as a Python prototype but required a more efficient implementation to handle larger datasets. Consequently, we transitioned to Rust, a language better suited for handling large-scale datasets with minimal latency. Despite the substantial size of the n-gram data, we opted to keep all data in memory to maximize retrieval speed and minimize delays from disk I/O. This approach was facilitated by a setup involving a 128GB RAM server, designed to handle the entire TinyStories [5] dataset with efficient in-memory operations.

Memory optimization was a primary focus during the Rust implementation. We adopted a sorted vector map (rust-shed crate) [18] and a reference counting library (rclite) [13], reducing memory usage by avoiding node copies and using references instead. Initial map capacities were optimized based on average branching factors at each n-gram order, setting the first order to the vocabulary

size, the second to 512 (a balance for complex and simple datasets), and subsequent orders to 2. This approach improved memory efficiency, particularly for retrieving all children of a node, where vectors with binary search performed adequately compared to hashmaps. Caching was added for crucial functions to improve query times, leveraging the continuation of smaller rules in longer rules. The trie was refined to support parallel computations, maintaining efficiency as n-gram counts and query complexity increased.

Our trie also required advanced rule calculations for greater precision, to address rare and unseen n-grams, we implemented Modified Kneser-Ney [14] smoothing, a well-suited approach in natural language applications where low-frequency sequences provide valuable insights. Refinements in the smoothing process decreased speed and as of now are still questionable if they help in decreasing the variational distance.

The library is accessible with Python bindings, but can also run as an http server.

Future improvements can include distributed computing. As all the rule's first symbol is + keep, we can easily distribute the requests based on the first token, thus reducing the required RAM. The batch request for statistics will be faster as well as the queries are happening in parallel.

5 Experimental Evaluation and Results

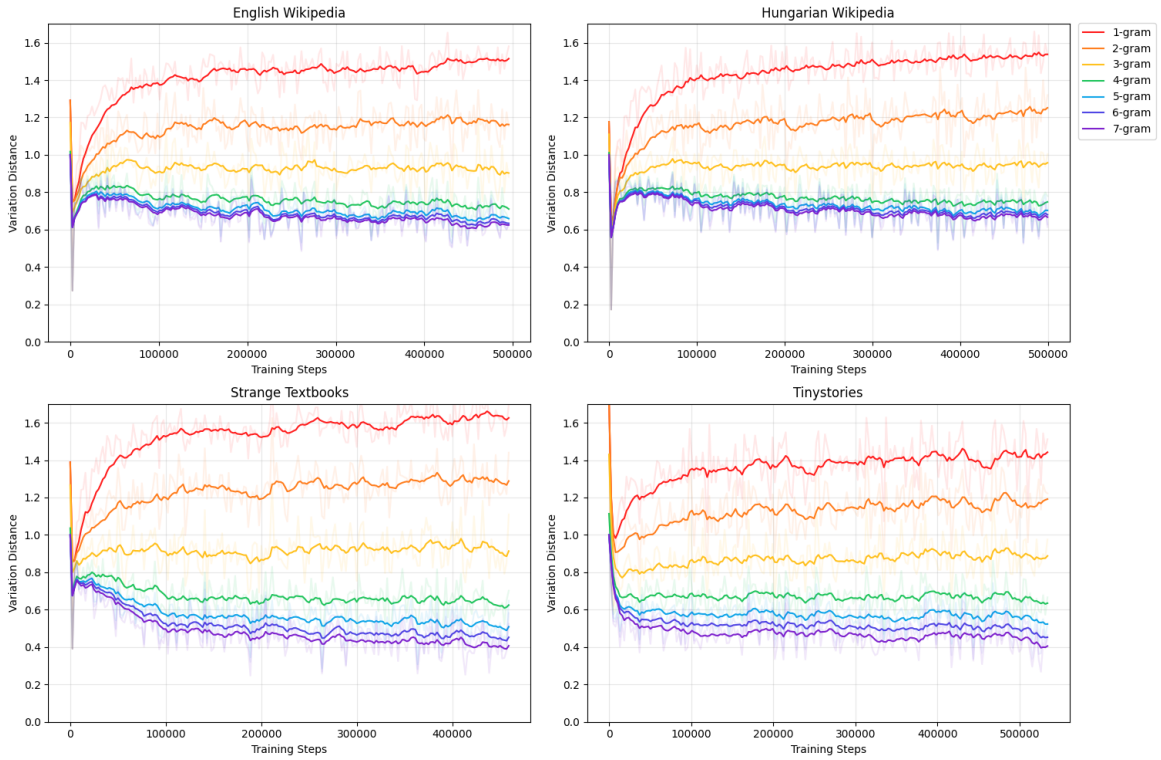


Figure 2: Variation Distance with unsmoothed ngram-statistics for different datasets, using all rules. Each color represent a different context length.

For all evaluated datasets, similar trends in variation distance, as observed by Nguyen in Figure 3 *left* of the original paper [21], are evident in our analysis (Figure 2). Specifically, for rules with a context length of 3 or less, the variation distance consistently increases as training progresses. Nguyen reported analogous behavior for context lengths of up to 4. However, our experimental setup significantly extends the training duration, running for 500,000 steps compared to Nguyen's 15,000, allowing us to uncover additional nuances. Notable differences across datasets emerge in this extended regime.

Zooming in on the first 15,000 training steps reveals a marked increase in variation distance for all context lengths within the Wikipedia datasets [6] (top panels). Furthermore, the Wikipedia datasets exhibit narrower gaps in variation distance between context lengths 5–7 compared to other datasets, suggesting greater uniformity in their contextual representation as training advances.

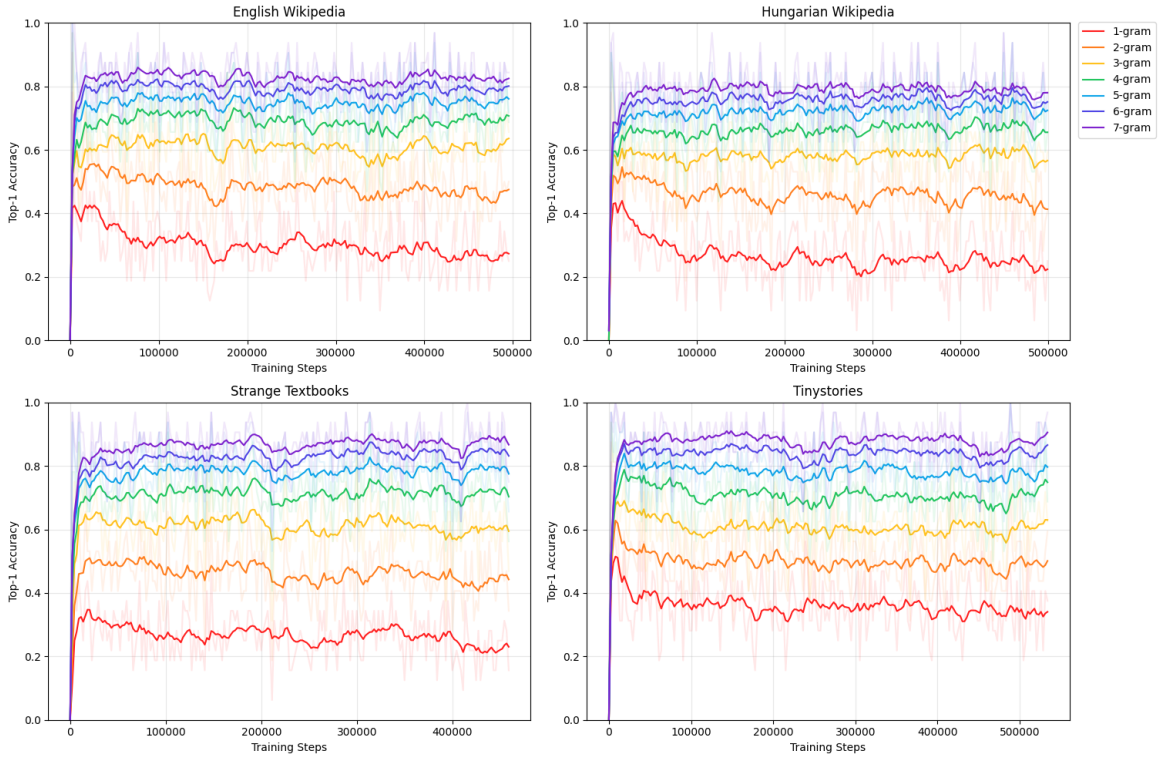


Figure 3: Top1 accuracies with unsmoothed ngram-statistics for different datasets, using all rules. Each color represent a different context length.

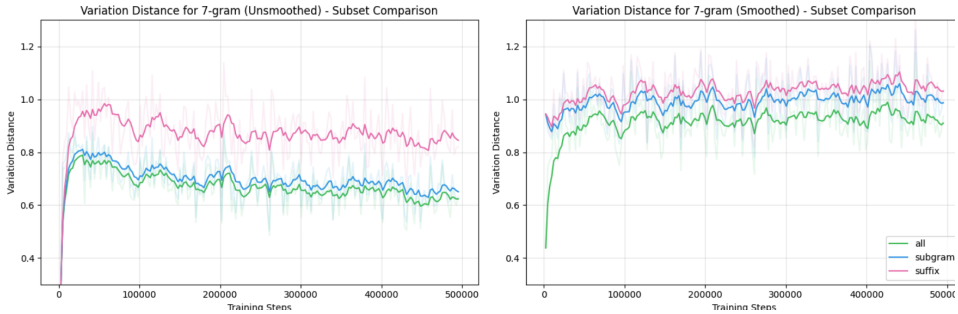


Figure 4: Subset comparison: Variation distance based on subset for unsmoothed and smoothed 7-gram rules.

Regarding the top-1 accuracy results depicted in Figure 3, longer context lengths universally achieve higher accuracy across all datasets. Notably, after the initial 15,000 steps the top1-accuracy is close to the maximum observed over the entire 500,000 steps span. A noteworthy deviation from Nguyen's findings is the behavior of shorter context lengths, particularly for contexts of length 1. In our experiments, these shorter contexts demonstrate declining accuracy after the initial 10,000 steps of training. This trend suggests that Nguyen might have observed similar behavior had the training been extended beyond 15,000 steps, underscoring the importance of prolonged experimentation in capturing long-term dynamics.

Figure 4 highlights the impact of smoothing and rule subsets on n-gram trie performance. Notably, n-gram predictions with smoothing diverge from those without smoothing, exhibiting differences that are partially expected but challenging to interpret. Additionally, while using all available rules generally maintains consistent patterns, specific subsets introduce variations that warrant further analysis to fully understand their implications.

6 Summary and Discussion

Our findings suggest that increasing dataset complexity (as shown in Figure 5) correlates with reduced differences in variation distance across higher-order n-grams. This observation aligns with the trends seen in the Wikipedia datasets when contrasted with the simpler, AI-generated datasets, TinyStories and Strange Textbooks.

In particular, the higher branching factors observed at the 4-, 5-, 6-, and 7-gram levels for TinyStories and Strange Textbooks may reflect the repetitive and templated nature of their content (e.g., phrases such as "once upon a time there was a cat/dog/tree"). Such repetition appears to enable us to to model their behavior predominantly through local context. Consequently, transformers trained on these simpler datasets can be more readily interpreted through local statistical patterns. However, this reliance on local modeling appears to diminish as dataset complexity grows, requiring the transformer to capture more intricate global contextual dependencies.

The variability within the datasets is illustrated in Figure A.2, which highlights that the Wikipedia datasets demonstrate greater variety per token. This is particularly evident in the number of nodes per layer, although all datasets share a low average branching factor for 7-grams, with roughly one token on average at that context length. A closer examination of the branching factors reveals consistently low values beyond layer 4, with averages falling below 2 for all datasets. Notably, TinyStories exhibits the highest expressibility among the datasets, with an average branching factor of 1.85 at layer 4, followed by StrangeTextbooks at 1.51, Wikipedia at 1.32, and Hungarian Wikipedia at 1.26. Interestingly, while TinyStories shows a relatively high expressibility in later layers, its branching factor for layer 1 is remarkably low at 129, compared to Hungarian Wikipedia, which leads with a branching factor of 910.

These trends suggest a clear relationship between branching factors and variation distance behavior. Low branching factors in the initial layers appear to correlate with a quicker decrease in variation distance for longer context lengths of 4 or more tokens. In the Wikipedia datasets, particularly the Hungarian subset, the average branching factors for layers 6 and 7 are as low as 1.06 and 1.04, respectively. This explains why the variation distances for 5-, 6-, and 7-grams are closely aligned. In contrast, TinyStories exhibits slightly higher branching factors of 1.25 and 1.17 for layers 6 and 7, contributing to a broader variation distance. For all datasets, the 4-gram variation distance (represented in green in Figure 2) decreases consistently, aligning with branching factors below 2 across layers.

Longer context lengths consistently result in higher top-1 accuracy across all datasets. However, the convergence of top-1 accuracy follows a similar trend to the variation distance: more complex datasets, such as Wikipedia and Hungarian, require longer to converge. The top-1 accuracy also indicates that these statistical patterns correlate with the transformer predictions, achieving approximately 80% accuracy or higher accross all datasets. While our results show higher accuracy compared to Nguyen's findings, they remain within a similar range: Nguyen reports 79% for TinyStories and 68% for Wikipedia.

The significantly inferior performance of smoothed probabilities (Figure 4) could further suggest that transformer predictions are largely derived from statistical patterns rather than generalizations to unseen data.

Our replication study confirms Nguyen's findings that n-gram rules with context lengths of 4 or more effectively explain transformer predictions, achieving accuracy levels comparable to those reported by Nguyen for diverse datasets. Additionally, our work extends these findings to more varied and complex datasets, highlighting their broader applicability. Future work could explore the generalizability of these results to larger models, other datasets and longer contexts.

7 Summary of Discussions with the Opposing Group

Both groups engaged in constructive discussions about Nguyen's findings, particularly regarding smoothing techniques and dataset trends. While both began with the TinyStories dataset, our group expanded to explore Tiny Textbooks and considered Wikipedia for broader analysis, reflecting a more comprehensive approach. Nguyen's use of "stupid backoff" smoothing sparked deeper discussions about the trade-offs between simplicity and accuracy.

Our custom training loop, optimized for rapid convergence, has shown promising results, reducing loss significantly in short test runs. In contrast, the opposing group employs a database-integrated pipeline that prioritizes scalability. Both approaches reflect different focuses, with our memory-based trie implementation offering faster lookups, while their SQL-based setup emphasizes structured data management.

Smoothing has been a shared challenge. We implemented Kneser-Ney smoothing to address rare n-grams, enhancing balance and performance, while the opposing group opted for a streamlined method, reflecting computational constraints. These differences underscore the complexity of achieving both efficiency and accuracy in n-gram models.

Collaboration between the groups has been constructive, with shared insights on training improvements and dataset exploration. Our focus on iterative testing and resource-efficient strategies contrasts with their emphasis on methodical reproducibility, highlighting complementary strengths in advancing the project.

References

[1] Abhishek Tamu, Arthur Zucker. Modeling llama (transformer), 2024.

[2] Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-Context Language Learning: Architectures and Algorithms, January 2024.

[3] Guido Bologna. A Simple Convolutional Neural Network with Rule Extraction. *Applied Sciences*, 9(12):2411, January 2019. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[4] Botond Lovász, Jonas Rosengren, Bjarni Haukur. n-gram trie library, 2024.

[5] Ronen Eldan and Yuanzhi Li. TinyStories: How Small Can Language Models Be and Still Speak Coherent English?, May 2023. arXiv:2305.07759 [cs].

[6] Wikimedia Foundation. Wikimedia downloads.

[7] M. Fukumi and N. Akamatsu. A new rule extraction method from neural networks. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 6, pages 4134–4138 vol.6, July 1999. ISSN: 1098-7576.

[8] Minoru Fukumi, Yasue Mitsukura, and Norio Akamatsu. Knowledge Incorporation and Rule Extraction in Neural Networks. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks — ICANN 2001*, pages 1248–1253, Berlin, Heidelberg, 2001. Springer.

[9] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Trans. Speech Lang. Process.*, 5(3), October 2008.

[10] Joshua Goodman. A Bit of Progress in Language Modeling, August 2001. arXiv:cs/0108005.

[11] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech 2020*, pages 5036–5040, 2020.

[12] Henrik Jacobsson. Rule Extraction from Recurrent Neural Networks: ATaxonomy and Review. *Neural Computation*, 17(6):1223–1263, June 2005.

[13] Khashayar Fereidani, Tom LeGodais, Nova, CosmicHorror. rclite, 2024.

[14] R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, May 1995. ISSN: 1520-6149.

[15] Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens, April 2024. arXiv:2401.17377 [cs].

[16] Luca Malagutti, Andrius Buinovskij, Anej Svete, Clara Meister, Afra Amini, and Ryan Cotterell. The role of n-gram smoothing in the age of neural networks, 2024.

[17] Clayton McMillan, Micahel C. Mozer, and Paul Smolensky. The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 13(0), 1991.

[18] Meta Platforms Inc. rust-shed, 2024.

[19] Nam Pham. tiny-textbooks (revision 14de7ba), 2023.

[20] Nam Pham. tiny-strange-textbooks (revision 6f304f1), 2024.

[21] Timothy Nguyen. Understanding Transformers via N-gram Statistics, June 2024. arXiv:2407.12034 [cs].

[22] T.R. Niesler and P.C. Woodland. A variable-length category-based n-gram language model. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 164–167 vol. 1, 1996.

[23] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. ProphetNet: Predicting Future N-gram for Sequence-to-SequencePre-training. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online, November 2020. Association for Computational Linguistics.

[24] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling Language Models: Methods, Analysis & Insights from Training Gopher, January 2022. arXiv:2112.11446 [cs].

[25] Brian Roark, Murat Saraclar, and Michael Collins. Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2):373–392, 2007.

[26] Aurko Roy, Rohan Anil, Guangda Lai, Benjamin Lee, Jeffrey Zhao, Shuyuan Zhang, Shibo Wang, Ye Zhang, Shen Wu, Rigel Swavely, Tao, Yu, Phuong Dao, Christopher Fifty, Zhifeng Chen, and Yonghui Wu. N-Grammer: Augmenting Transformers with latent n-grams, July 2022. arXiv:2207.06366.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[28] Andreas Waldis, Luca Mazzola, and Michael Kaufmann. Concept Extraction with Convolutional Neural Networks. In *Proceedings of the 7th International Conference on Data Science, Technology and Applications*, pages 118–129, October 2024.

A Appendix

A.1 Transformer Config

Parameter	Value
vocab_size	16,384
hidden_size	768
num_hidden_layers	12
num_attention_heads	12
intermediate_size	3,072
max_position_embeddings	2,048
Total Parameters	138,431,232

Table 1: Transformer Configuration

Table 1 outlines the configuration details for a Transformer model. Key parameters include the vocabulary size, the hidden layer size, and the number of attention heads and hidden layers, which influence the model's capacity and performance. The total number of parameters is approximately 138 million, which represents the complexity of the model. It uses 12 hidden layers, each with 12 attention heads, and supports a maximum position embedding length of 2048, enabling it to handle long sequences effectively.

A.2 Dataset Trie Statistics

Comprehensive characteristics of the fitted n-gram tries are displayed for each dataset. Figure 5 shows it visually while Table 2 shows absolute numbers. Average branching factor for layer n is calculated as $\frac{nodes\ layer\ n+1}{nodes\ layer\ n}$.

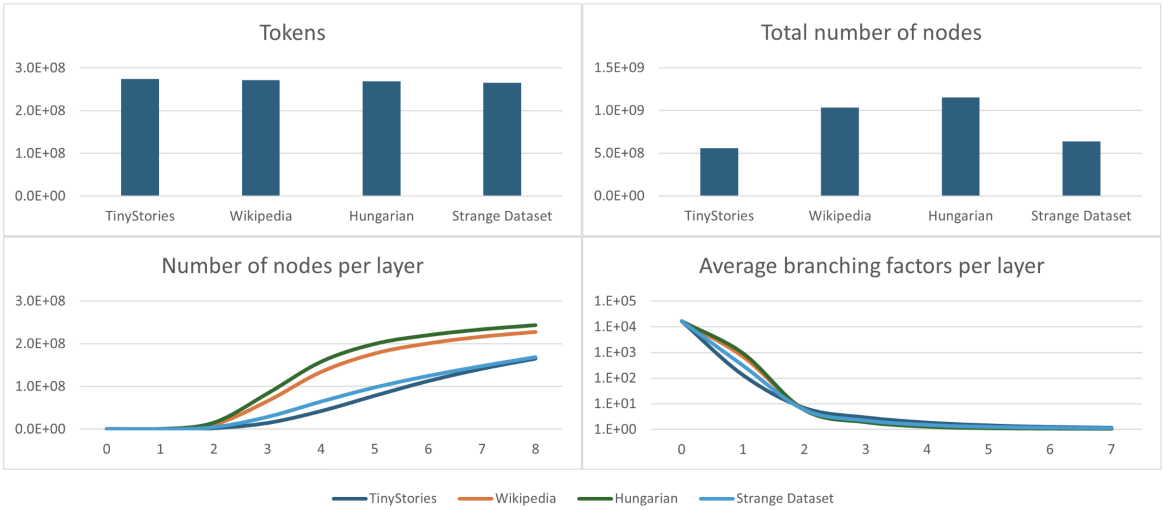


Figure 5: Data Analysis. Branching factors and nodes per layers for different datasets. Log-scale is used for branching factors.

Table 2: Branching Factors and Nodes per Layer for Various Datasets

Metric	TinyStories	Wikipedia	Hungarian	Strange
Total Nodes in Trie	558,054,306	1,035,654,436	1,154,669,279	639,679,835
Average Branching Factors per Layer				
Layer 0	16,384	16,384	16,384	16,384
Layer 1	129.59	674.25	910.71	304.81
Layer 2	6.82	5.98	5.59	5.81
Layer 3	2.93	2.04	1.89	2.24
Layer 4	1.85	1.32	1.26	1.51
Layer 5	1.44	1.13	1.10	1.27
Layer 6	1.25	1.08	1.06	1.18
Layer 7	1.17	1.05	1.04	1.14
Nodes per Layer				
Layer 0	1	1	1	1
Layer 1	16,384	16,384	16,384	16,384
Layer 2	2,123,263	11,046,856	14,921,145	4,994,051
Layer 3	14,490,738	66,014,723	83,447,753	28,994,087
Layer 4	42,434,830	134,490,569	158,078,116	65,054,420
Layer 5	78,489,699	177,626,035	199,948,117	98,150,779
Layer 6	113,072,832	201,265,744	220,422,894	125,133,092
Layer 7	141,835,059	217,103,047	234,010,356	148,256,979
Layer 8	165,591,500	228,091,077	243,824,513	169,080,042