Discovering Hidden Algebraic Structures via Transformers with Rank-Aware Beam GRPO

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent efforts have extended the capabilities of transformers in logical reasoning and symbolic computations. In this work, we investigate their capacity for non-linear latent pattern discovery in the context of functional decomposition, focusing on the challenging algebraic task of multivariate polynomial decomposition. This problem, with widespread applications in science and engineering, is proved to be NP-hard, and demands both precision and insight. Our contributions are threefold: First, we develop a synthetic data generation pipeline providing fine-grained control over problem complexity. Second, we train transformer models via supervised learning and evaluate them across four key dimensions involving scaling behavior and generalizability. Third, we propose Beam Grouped Relative Policy Optimization (BGRPO), a rank-aware reinforcement learning method suitable for hard algebraic problems. Finetuning with BGRPO improves accuracy while reducing beam width by up to half, resulting in approximately 75% lower inference compute. Additionally, our model demonstrates competitive performance in polynomial simplification, outperforming Mathematica in various cases.

1 Introduction

Transformers, initially developed for natural language processing [34], have shown remarkable versatility across diverse domains such as vision [7] and protein folding [17]. More recently, their applications in formal reasoning, symbolic mathematics and algorithmic tasks start to gain traction. Several works have showcased transformer-based architectures' ability to tackle highly structured problems, including theorem proving [27, 33], integration [20], matrix multiplication [12] and equation solving [8].

In this work, we investigate the transformer's capacity for non-linear latent pattern discovery in the context of functional decomposition, i.e. decomposing a complex function as the composition of simpler sub-functions. In contrast to step-by-step logical deduction, or pattern recognition in data analysis, functional decomposition poses significant new challenges to the transformer, because the forms of the sub-functions that we try to discover can be totally hidden or obscured in the final compact form of the original function. Furthermore, it requires extreme precision without any margin of error. Unlike more forgiving classification tasks, the decomposition problem admits only a sparse set of correct solutions: even minor deviations in signs or coefficients can render outputs completely invalid.

Beyond its theoretical interest, functional decomposition has ubiquitous applications in software engineering [32], systems biology [24], mechanical design [31], systems engineering [15] and digital logic design [1, 21], where capturing hidden substructures within high-dimensional functions leads to more tractable and efficient models. However, identifying a function's latent compositional structure

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

requires models to look past surface-level correlations, attending instead to deep algebraic symmetries and invariants.

A particularly rich case of functional decomposition arises in multivariate polynomial functions. The polynomial decomposition problem over a ring k seeks to decompose a given polynomial $f \in k[x_1, \ldots, x_n]$ into polynomials $g \in k[y_1, \ldots, y_m]$ and $h_1, \ldots, h_m \in k[x_1, \ldots, x_n]$ such that

$$f(x_1, \ldots, x_n) = g\big(h_1(x_1, \ldots, x_n),\ \ldots\ ,\ h_m(x_1, \ldots, x_n)\big). \tag{1}$$

It has wide-ranging applications from cryptography [26] to dynamical modeling [2], signal processing [4] and robotics [9, 23].

The multivariate polynomial decomposition problem has been proved to be NP-hard by Dickerson [5, 6], although efficient algorithms for various special cases are discussed in [14, 36, 37, 10, 11, 40]. To illustrate the difficulty of the problem for the models, let us consider the following expression

$$\begin{aligned} f =& 2a_1^3b_1^3 + 25a_1^2b_1^2 + 6a_1^2a_2b_2b_1^2 + 6a_1^2a_3b_3b_1^2 + 6a_1a_2^2b_2^2b_1 + 6a_1a_3^2b_3^2b_1 \\ &+ 96a_1b_1 + 50a_1a_2b_2b_1 + 50a_1a_3b_3b_1 + 12a_1a_2a_3b_2b_3b_1 + 2a_2^3b_2^3 + 2a_3^3b_3^3 \\ &+ 25a_2^2b_2^2 + 25a_3^2b_3^2 + 6a_2a_3^2b_2b_3^2 + 96a_2b_2 + 6a_2^2a_3b_2^2b_3 + 96a_3b_3 \\ &+ 50a_2a_3b_2b_3 + 12 \end{aligned}$$

It has a hidden $O(3)$-symmetry, which can be revealed by decomposing $f = g \circ h$, with $g(y) = y^2 + 2(4 + y)^3$ and $h = a_1b_1 + a_2b_2 + a_3b_3$. This is a highly nontrivial task to identify the inner function h directly from the expanded form of f, as its structure becomes completely obscured after polynomial substitution, expansion and simplification. Even in this relatively constrained case where g is univariate, discovering the decomposition requires recognizing non-linear latent patterns across dozens of terms. When g becomes multivariate, the complexity increases substantially, making the problem even more challenging.

To tackle the polynomial decomposition problem, we develop a systematic approach with four key components. First, we create a backward synthetic data generation pipeline that allows fine-grained control over polynomial complexity involving range of coefficients, degree, and number of variables. Second, we train lightweight transformer models on these synthetic datasets using supervised learning and analyze how performance scales across four axes (performance complexity scaling, architecture scaling, distribution adaptation, search strategy analysis). Third, we discover that both multi-sampling and greedy search methods struggle with the sparse solution space of the polynomial decomposition problem, and we implement a beam search strategy to effectively extract the models' capabilities. Finally, to address the computational intensity of beam search, we develop a rank-aware variant of the Grouped Relative Policy Optimization (GRPO) reinforcement learning algorithm, which encodes rank information directly in the reward function.

Our study makes the following contributions to neural approaches for polynomial decomposition. First, our backward data generation pipeline enables targeted training across varying levels of decomposition difficulty. Second, our comprehensive evaluation across four dimensions, for the first time, establishes robust baselines for transformers' performance on polynomial decomposition tasks. Third, using the rank-aware Beam Grouped Relative Policy Optimization (BGRPO), our models improve accuracy while reducing beam search width by up to 50%, resulting in 75% lower computational requirements during inference. Additionally, our model demonstrates competitive performance in polynomial simplification, outperforming Mathematica in various cases. This underscores the potential of neural models to complement and extend classical symbolic computation capabilities.

2 Method

2.1 Backward Synthetic Data Generation

We generate synthetic data for supervised learning using a backward approach, starting from the decomposed form. First, we generate the inner functions ($h_1, \ldots, h_m$ in Eq. (1)) and the outer function (g in Eq. (1)) with random monomial terms of bounded degree and random coefficients within a given range. Then, we obtain the composed function (f in Eq. (1)) via substitution, expansion, and term collection. See Appendix A for the detailed algorithm. For each generated instance, we create a training pair consisting of the expanded polynomial f as input and its decomposed components

$\{g, h_1, \ldots, h_{v_{\text{outer}}}\}$ as the target output. The model is trained to minimize the standard negative log-likelihood loss function.

Our synthetic data generation process provides fine-grained control over problem complexity through eight parameters: C_{inner} (coefficient range for inner polynomials), d_{inner} (maximum degree of inner polynomials), v_{inner} (number of variables in inner polynomials), t_{inner} (maximum number of terms in inner polynomials), and similarly C_{outer}, d_{outer}, v_{outer}, and t_{outer} for the outer polynomial.

2.2 Beam Search

Beam search is a breadth-first search algorithm that approximates optimal decoding by keeping track of the k most probable sequences at each step [13]. For each of the k current sequences, the algorithm considers the top-k token extensions per sequence. These k^2 candidate continuations are then ranked by the sum of log probabilities of all tokens in the sequence, and only the top-k sequences with the highest cumulative log probability are retained for the next step. In this paper, we refer to k as the beam width, and to the position (1st, 2nd, etc.) of an output in the final beam as its rank.

Our analysis across all model outputs identified a specific error pattern in polynomial decomposition: the model achieves approximately 90% accuracy for predicting non-sign tokens (operators, numbers, variables), but exhibits near-random performance for deciding between positive and negative signs. This creates a unique inference challenge where exploration needs to be constrained for high-confidence structural elements while simultaneously expanded for uncertain sign choices.

Beam search is particularly well-suited for this situation as it maintains the high-confidence structural backbone while systematically exploring variations in the uncertain components. Our experiments demonstrate that beam search significantly outperforms greedy decoding and random sampling for polynomial decomposition tasks. See Appendix C for a detailed error analysis and an explanation of beam search effectiveness for this task.

2.3 BGRPO : Reinforcement Learning Method Enhancing Beam Search Efficiency

The computational cost of beam search scales quadratically with beam width. There would be a significant computational advantage if we could improve the ranks of correct outputs. To address this, we introduce Beam Grouped Relative Policy Optimization (BGRPO), a reinforcement learning method that extends GRPO, uniquely taking into account rankings in the beam search, specifically designed for improving beam search inference efficiency.

Reinforcement learning enables models to explore solution spaces more effectively than supervised learning alone, enhancing the model's capabilities by addressing specific weaknesses through a reward mechanism. This approach encourages correct answers while discouraging incorrect ones based on an advantage function—the difference between a solution's reward and a baseline reward. Group Relative Policy Optimization (GRPO) [30] estimates this baseline for each question by sampling a group of outputs, and has shown promising results for reinforcement learning in language generation tasks due to its sample efficiency and stability [3]. We chose GRPO over traditional Proximal Policy Optimization (PPO) [29] because it eliminates the need for a separate value network or reward model, reducing training complexity while improving stability, and its group-wise baseline calculation naturally fits tasks with a clear binary reward structure like polynomial decomposition.

Our proposed Beam Grouped Relative Policy Optimization (BGRPO) extends this approach by using beam search rather than independent sampling for generating the group of outputs. While this significantly alters the distribution of outputs, making their average reward less suitable as a traditional baseline, it still provides valid training signals by reinforcing correct answers and penalizing incorrect ones. BGRPO is particularly effective for our task because beam search generates outputs with identical structure that differ only in the confusing elements (signs), creating a focused learning signal.

Additionally, BGRPO incorporates rank information directly into the reward function by applying an exponential decay factor based on the position in the beam. This incentivizes correct answers to appear at earlier positions in the beam search, effectively pushing correct solutions toward the top of the beam ranking.

Training Objective For a prompt x, let $\mathcal{B}(x) = \{y_1, \ldots, y_w\}$ be the set of beam search outputs with beam width w generated by the old policy $\pi_{\theta_{\text{old}}}$. Each output sequence y_i receives a reward r_i, where $r_i = 0$ for incorrect polynomial decomposition and $r_i = 1$ for correct decomposition. In BGRPO, we incorporate rank information by scaling the reward for correct decompositions using an exponential decay function $e^{-\text{rank}/w}$. We optimize the policy model π_θ for μ iterations by maximizing the following objective:

$$\mathcal{J}_{\text{BGRPO}}(\theta) = \frac{1}{w}\sum_{i=1}^{w}\left(\min\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}A_i, \text{clip}\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right), \tag{2}$$

where ε is the clipping parameter that constrains policy updates and β controls the KL divergence regularization term:

$$\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1. \tag{3}$$

Here, π_{ref} is the reference policy, which is the initial model before BGRPO training. The advantage function A_i is computed without normalization as $A_i = r_i - \text{mean}(\{r_1, r_2, \cdots, r_w\})$, following the approach in [22].

3 Experimental Setup

3.1 Evaluation Axes

To systematically analyze our models' capabilities for the polynomial decomposition problem, we consider four key evaluation dimensions.

Problem Complexity Scaling ($\mathcal{D}_1$). We analyze how the model performance varies with respect to changes in the complexity parameters for synthetic data generation. We vary the number of variables $v_{\text{inner}}, v_{\text{outer}}$, and the maximum degrees $d_{\text{inner}}, d_{\text{outer}}$ for both the inner and outer polynomials.

Architecture Scaling ($\mathcal{D}_2$). We investigate how model performance scales with key architectural hyperparameters of the transformer. In particular, we measure $\mathcal{P}(M(d, l, a))$, the performance of models with embedding dimension d, number of layers l, and number of attention heads a. Our goal is to characterize how these hyperparameters influence model capabilities.

Distribution Adaptation ($\mathcal{D}_3$). A practical challenge in applying transformers to symbolic computation is their sensitivity to the numerical ranges present in the training data. For example, models trained on specific coefficient ranges tend to struggle with polynomials outside these ranges. On the other hand, we found that models can rapidly adapt to new coefficient distributions with minimal additional training, suggesting that they manage to learn generalizable pattern recognition rather than merely memorizing specific numerical relationships.

To quantify the model's ability to transfer its polynomial decomposition skills to numerically distinct but structurally identical problems, we prepare the model $M^n_{C_1\to C_2}$. This model is initially trained on 1M polynomial decomposition examples with $C_{\text{outer}} = C_1$ and then fine-tuned with n examples with $C_{\text{outer}} = C_2$ where $C_1 \cap C_2 = \emptyset$. We measure the performance of model $M^n_{C_1\to C_2}$ on a test set of polynomial decomposition problems with $C_{\text{outer}} = C_2$:

$$\mathcal{G}(n) = \mathcal{P}\big(M^n_{C_1\to C_2},\ \text{test set with } C_{\text{outer}} = C_2\big) \tag{4}$$

Search Strategy Analysis ($\mathcal{D}_4$). We investigate how beam search enhances model performance on polynomial decomposition tasks, analyzing its effectiveness across different model architectures and levels of problem complexity.

3.2 Synthetic Dataset Setup

For the axis $\mathcal{D}_1$ of the problem complexity scaling, we first examine degree scaling by training a model on 2M polynomial decomposition examples with different inner and outer degrees as described in Table 1. We then evaluate this model on separate test datasets with the same configuration parameters, each corresponding to one of nine different $(d_{\text{inner}}, d_{\text{outer}})$ pairs to assess performance across varying problem complexities.

For the second part of the $\mathcal{D}_1$ axis, we train a model for each combination of v_{inner} and v_{outer} varying from 2 to 4 while fixing the other parameter at 3. For each combination, we use 1M examples to train the model.

For the axis $\mathcal{D}_2$ of architecture scaling, we train multiple models with varying architectural configurations, all using the same dataset of 2M examples with polynomial parameters as described in Table 1.

For the axis $\mathcal{D}_3$ of distribution adaptation, we train initial models on 1M examples with $C_{\text{outer}} = C_1 = [-5, 5]$ and then adapt them to examples with $C_{\text{outer}} = C_2 = [-10, -6] \cup [6, 10]$. Other parameters are the same across both datasets as described in Table 1.

For the second part of $\mathcal{D}_1$ (Variable Scaling) and $\mathcal{D}_2$, we set $t_{\text{inner}} = t_{\text{outer}} = 3$ to prevent expressions from becoming too long. We describe our tokenization in Appendix B.

Table 1: Synthetic Dataset Configuration Across Evaluation Axes

Evaluation Axis	Inner Coeff.	Outer Coeff.	Inner Degrees	Outer Degrees	Inner Vars	Outer Vars
$\mathcal{D}_1$ (Degree Scaling)	$[-20, 20]$	$[-20, 20]$	$\{2, 3, 4\}$	$\{2, 3, 4\}$	1	1
$\mathcal{D}_1$ (Variable Scaling)	$[-5, 5]$	$[-5, 5]$	3	3	$\{2, 3, 4\}$	$\{2, 3, 4\}$
$\mathcal{D}_2$ (Architecture)	$[-5, 5]$	$[-5, 5]$	3	3	3	3
$\mathcal{D}_3$ (Adaptation)	$[-20, 20]$	$C_1 = [-5, 5]$	$\{1, 2\}$	$\{1, 2, 3, 4\}$	1	1
	$[-20, 20]$	$C_2 = [-10, -6] \cup [6, 10]$	$\{1, 2\}$	$\{1, 2, 3, 4\}$	1	1

3.3 Architecture Configuration

We employ a decoder-only transformer architecture following standard design principles [34]. Table 2 summarizes our task-specific configurations across all experimental axes. For lightweight and effective training, we developed our own model and training pipeline based on `minGPT` [19].

Table 2: Transformer Model Configuration Across Experiments

Experiment	Context Window	Embedding Dim.	Layers	Heads
$\mathcal{D}_1$ (Degree Scaling)	256	512	6	8
$\mathcal{D}_1$ (Variable Scaling)	850	512	6	8
$\mathcal{D}_2$ (Architecture)	850	{256, 512, 768}	{4, 6}	8
$\mathcal{D}_2$ (Attention Heads)	850	512	6	{4, 8, 16}
$\mathcal{D}_3$ (Distribution)	256	512	4	8

Common settings: GELU activation, learned positional embeddings, multi-head attention with causal masking, MLP hidden dimension = $4\times$ embedding dimension.

3.4 Supervised Learning Details

We train our models using the Adam optimizer with an initial learning rate of 6×10^{-4}, incorporating a 10% warmup period followed by cosine decay. Each configuration initially trains on 1M instances, with additional 1M training examples added incrementally until performance saturation. We use a batch size of 200 throughout training. We train models with enough epochs until it saturates with the given dataset.

3.5 BGRPO Implementation

For the BGRPO reinforcement learning phase, we generate candidate solutions using beam search with a width of 32 and temperature of 1.0. We implement our approach using the GRPO functionality from the `trl` library [35]. The training process consists of 5 policy update iterations after sampling outputs for 8 distinct polynomial decomposition problems. We set the PPO clipping parameter ε to 0.2 and the KL divergence coefficient β to 0.01. The learning rate during BGRPO training is 1×10^{-5}. We train models from $\mathcal{D}_2$ on a dataset of 200 non-repeating problems, saving checkpoints every 5 iterations and selecting the best model based on performance with beam width 7.

4 Experimental Results

4.1 Problem Complexity Scaling ($\mathcal{D}_1$)

In the first part of $\mathcal{D}_1$, we examine how model performance varies with the degrees of inner and outer polynomials. The result is shown in Figure 1. We use greedy search for the inference. Regardless of the degrees of the polynomials, our model achieves a remarkable single-output accuracy. Notably, when using beam search with a width of 10, the model's accuracy reaches 100% for these configurations.

Our analysis reveals a pattern: performance remains invariant to increases in the outer polynomial's degree, while decreasing when the inner polynomial's degree increases. This demonstrates that the transformer's decomposition capability is primarily limited by the complexity of the inner polynomial rather than that of the outer polynomial.

In the second part of $\mathcal{D}_1$, we investigate how the performance scales with v_{inner} and v_{outer}, the number of variables in the inner and outer polynomials. Figures 2 and 3 present these results.

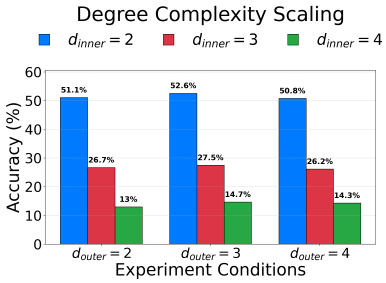

Figure 1: Performance across different $d_{\text{inner}}, d_{\text{outer}}$

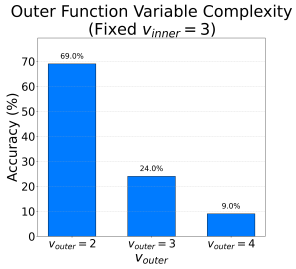

Figure 2: Performance across different v_{outer}

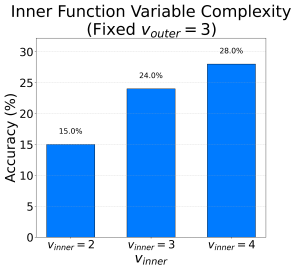

Figure 3: Performance across different v_{inner}

Given the challenging nature of multivariate polynomial decomposition, we evaluate the model's performance using beam search with a width of 30, considering a prediction correct if at least one of the 30 candidate outputs is correct decomposition.

Our results reveal two trends: performance decreases dramatically as v_{outer} increases, yet counter-intuitively improves as v_{inner} increases. This observation aligns with the following heuristic understanding: higher v_{outer} creates an information bottleneck, requiring the model to simultaneously resolve multiple interdependent inner functions. In contrast, higher v_{inner} provides more dimensions of input variation with additional structural indicators that can guide the decomposition process.

4.2 Architecture Scaling ($\mathcal{D}_2$)

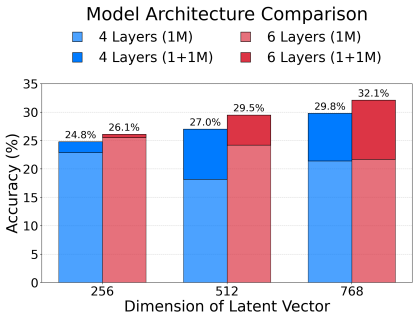

Figure 4: Accuracies on different number of layer and dimension.

In $\mathcal{D}_2$, we examine how model performance varies with architectural parameters: embedding dimension, number of layers, and number of attention heads. When varying the number of heads, we maintain a constant total embedding dimension, meaning that models with more heads have smaller per-head embedding dimensions. We use the dataset described in Section 3.2 and evaluate using beam search with a width of 30.

Figure 4 reveals the scaling behavior [18] of transformer architectures on polynomial decomposition. As model capacity increases through higher embedding dimensions and additional layers, performance consistently improves.

Notably, our results demonstrate the presence of a data-dependent scaling threshold. With limited training data (1M examples), larger models initially underperform their simpler counterparts, particularly evident in the 6-layer configurations with higher embedding dimensions. However, this pattern reverses completely with additional training data,

confirming that larger models possess superior capacity for mathematical pattern recognition when provided with sufficient examples to leverage their parametric advantage.

In $\mathcal{D}_2$, we also examine model performance with different numbers of attention heads. Our experiments reveal that increasing the number of attention heads while maintaining constant total embedding dimension leads to progressively deteriorating performance on polynomial decomposition tasks. Models with 4 heads achieved 32.0% accuracy, while those with 8 and 16 heads reached only 28.0% and 25.0% accuracy, respectively. This suggests that for our specific task of mathematical pattern recognition, fewer, more expressive attention heads with larger per-head dimensions provide better performance than numerous specialized heads with smaller dimensions.

4.3 Distribution Adaptation ($\mathcal{D}_3$)

We evaluate $\mathcal{G}(n)$ as defined in Eq. 4, which measures how quickly models adapt to new coefficient distributions as a function of adaptation sample size n. For this experiment, we train a model with 4 layers and 512 embedding dimension on the dataset described in Section 3.2. The initial training used 1M examples with outer polynomial coefficient range C_1, followed by fine-tuning on n examples with coefficient range C_2 for a single epoch. We report the variance in accuracy based on three independent trials.

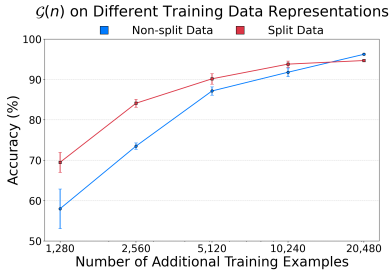

Figure 5: Performance recovery when adapting to a new coefficient distribution

Models trained exclusively on the first dataset achieve only 5.67% accuracy on the new distribution, despite reaching nearly 100% accuracy on the original distribution. Figure 5 illustrates how performance recovers during adaptation. Notably, despite using only $\approx 2\%$ of the original training data size, the model rapidly recovers its accuracy from single digits to over 90%. This rapid adaptation indicates successful transfer learning, suggesting that the model develops a general mathematical understanding of polynomial substructures rather than memorizing specific numerical relationships.

We further investigate whether alternative data representations could enhance this adaptation capability. We propose "split" representation of polynomials, where we randomly select terms from the expanded form and split their coefficients. For example:

$$\begin{aligned} f_{\text{non-split}}(a) &= -63 + 23a - 71a^2 - 11a^3 - 14a^4 - 12a^5 - 2a^6 \\ f_{\text{split}}(a) &= -63 + 23a - 4a^2 - 67a^2 - 8a^3 - 3a^3 - 7a^4 - 7a^4 - 12a^5 - a^6 - a^6 \end{aligned} \tag{5}$$

In Figure 5, the red line demonstrates $\mathcal{G}(n)$ of the model trained on data with both normal and split representation. Models trained on this mixed data including split representation demonstrate significantly faster adaptation, requiring only 70% of the additional training examples to reach equivalent performance on the new distribution.

This enhanced generalization likely stems from the model being forced to recognize mathematically equivalent but differently represented polynomials, compelling it to develop a deeper understanding of polynomial structure rather than memorizing specific patterns.

4.4 Search Strategy Analysis ($\mathcal{D}_4$)

We evaluate how search strategies impact model performance on polynomial decomposition tasks, with a particular focus on beam search efficiency. Figure 6 and 7 illustrate the accuracy achieved across different beam widths for polynomials with varying numbers of variables.

Our results reveal an unusually dramatic impact of beam search for polynomial decomposition compared to typical NLP tasks. For two-variable polynomials, accuracy improves from 11% with greedy search to 69% with a beam width of 30—a remarkable 6.3× improvement. This stands in stark contrast to standard neural machine translation applications, where beam search typically yields BLEU score improvements of only 2-4 points [16, 28]. Even more telling, most NMT systems show diminishing returns with beam widths beyond 5-10 [13].

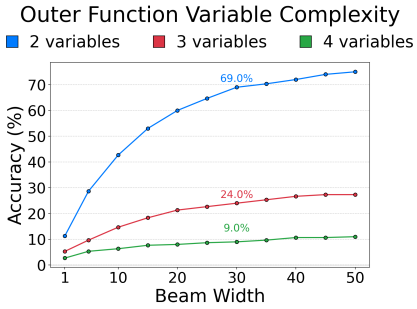

Figure 6: Beam width scaling with varying v_{outer} ($v_{\text{inner}} = 3$)

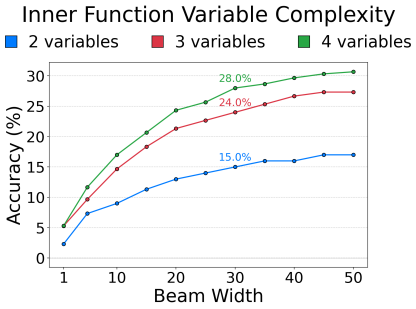

Figure 7: Beam width scaling with varying v_{inner} ($v_{\text{outer}} = 3$)

4.5 BGRPO Results

We evaluated BGRPO across models of varying sizes from our architecture scaling experiments($\mathcal{D}_2$), implementing versions both with and without rank signal. Fig 8 illustrates these results.

BGRPO consistently improved accuracy across all beam widths regardless of model size. Without rank signal, BGRPO gives average accuracy increases of 34.0%, 17.8%, and 12.4% for 6-layer models with dimension 256, 512, and 768 respectively. Including rank signal in BGRPO produces even more improvements, with average accuracy increases of 46.6%, 28.4%, and 30.2%.

These improvements translate to significant computational efficiency gains. For instance, the dimension-256 model initially achieved 26.1% accuracy with beam width 30. After applying BGRPO with rank signal, comparable accuracy (26.0%) was achieved with just beam width 16. This effectively halves the required beam width for equivalent performance. Since beam search computation scales quadratically with beam width, this improvement reduces beam search computation by approximately 75% while maintaining equivalent performance.

On average, BGRPO without rank signal reduced the required beam width by 31.3%, 14.9%, and 11.4% for 6-layer models with dimension 256, 512, and 768 respectively. When incorporating rank signal, BGRPO reduced required beam width even further, by 38.9%, 22.0%, and 26.5%.

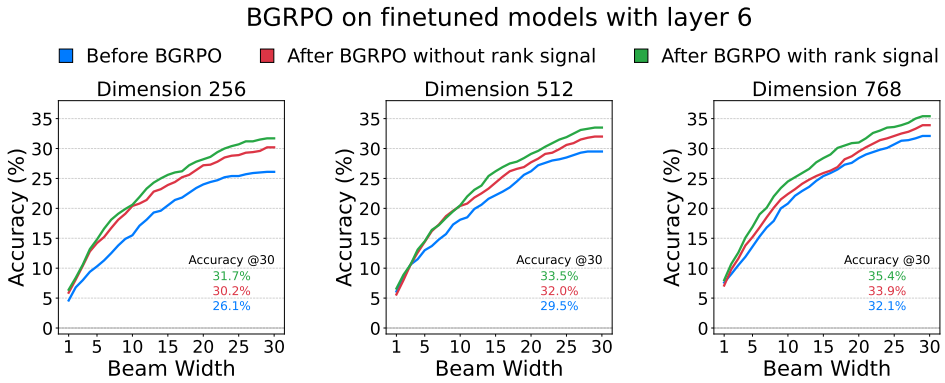

Figure 8: Accuracies on experiments with different dimension. Each experiment we have finetuned model with 2M data and models trained with BGRPO with and without rank signal on top of that.

4.6 Simplification Comparison with Mathematica

While polynomial simplification and polynomial decomposition represent two distinct mathematical objectives, simplification frequently arises as a consequence of decomposition, since decomposed forms generally exhibit reduced algebraic complexity compared to the original expression. In this subsection, we briefly explore the capabilities of our models for this related problem, and benchmark against the most powerful symbolic computation engine Mathematica. Despite our lightweight parameter budgets and the absence of any explicit simplification objective in our training, the models

were able to reduce the leaf count [39] of complex expressions, with performance on par with — and in two of five complexity regimes surpassing —Mathematica's state-of-the-art FullSimplify function (see Table 3, competitive performances are bolded).

Table 3: Average leaf count comparison (Beam width $= 30$)

Problem Complexity		**Leaf Count (mean)**		
v_O	v_S	Transformer	Mathematica	Δ
2	3	**27.28**	30.03	**-2.75**
3	3	22.85	**22.12**	**0.73**
4	3	22.52	20.00	2.52
3	2	17.27	**17.10**	**0.17**
3	4	**26.04**	27.56	**-1.52**

These findings highlight that transformers' inherent ability to uncover latent patterns rivals that of the most advanced symbolic computation methods.

5 Conclusion

Our investigation into transformers for polynomial decomposition uncovers key insights into how neural networks can infer hidden algebraic structures.

We find that model performance depends asymmetrically on polynomial complexity parameters ($\mathcal{D}_1$): inner polynomial degree plays a dominant role, while outer polynomial complexity has limited impact. Counterintuitively, increasing the number of inner variables improves accuracy by imposing structural constraints, whereas more outer variables create information bottlenecks.

From an architectural viewpoint ($\mathcal{D}_2$), we confirm that performance scales with model size. We observe that fewer but more expressive attention heads are especially effective for this task. In terms of distribution adaptation ($\mathcal{D}_3$), models transfer rapidly to new coefficient distributions, requiring as little as 2% of the original training data, indicating that they internalize generalizable principles rather than rely on memorization. Moreover, we can enhance this generalization capability through strategic dataset design.

Beam search analysis ($\mathcal{D}_4$) yields up to 6.3× improvement over greedy decoding due to the sparse, precise nature of mathematical solutions. Models finetuned with our rank-aware BGRPO reinforcement learning method achieve equivalent accuracy with up to 50% smaller beam widths, cutting inference computation by approximately 75%. Lastly, our model demonstrates competitive performance in polynomial simplification compared with symbolic computation tools in Mathematica.

Our work provides, for the first time, a systematic analysis of transformer capabilities for polynomial decomposition through carefully controlled experiments across four dimensions. Our methodologies can serve as a road map for exploring neural models in other domains that require non-local latent pattern discovery, such as functional decomposition problems ranging from systems engineering and mechanical design to digital logic design. While we developed BGRPO specifically for enhancing beam search in the polynomial decomposition problem, similar techniques may prove useful in other domains with sparse solution spaces where models can identify correct structures but struggle with specific details.

Limitations Our generalizability investigation was constrained to univariate polynomials with relatively narrow coefficient ranges and limited maximum degrees. Computational constraints restricted our architecture scaling experiments to relatively small models (maximum 6 layers, 768-dimensional embeddings); however, the consistent performance improvements without accuracy saturation suggest that further scaling would yield additional gains. Finally, our method's reliance on wide beam search creates computational overhead during inference despite BGRPO's improvements.

References

[1] Marian Andrzej Adamski, Andrei Karatkevich, Marek Wegrzyn, Mariusz Rawski, Tadeusz Łuba, Zbigniew Jachna, and Paweł Tomaszewicz. The influence of functional decomposition on modern digital design process. *Design of Embedded Control Systems*, pages 193–204, 2005.

[2] Thao Dang and Romain Testylier. Reachability analysis for polynomial dynamical systems using the bernstein expansion. *Reliab. Comput.*, 17(2):128–152, 2012.

[3] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

[4] Sefa Demirtas, Guolong Su, and Alan V Oppenheim. Sensitivity of polynomial composition and decomposition for signal processing applications. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 391–395. IEEE, 2012.

[5] Matthew T Dickerson. Polynomial decomposition algorithms for multivariate polynomials. Technical report, Cornell University, 1987.

[6] Matthew T Dickerson. General polynomial decomposition and the s-1-decomposition are np-hard. *International Journal of Foundations of Computer Science*, 4(02):147–156, 1993.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32):e2123433119, 2022.

[9] Alexander J Elias and John T Wen. Ik-geo: Unified robot inverse kinematics using subproblem decomposition. *Mechanism and Machine Theory*, 209:105971, 2025.

[10] Jean-Charles Faugère and Ludovic Perret. An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography. *Journal of Symbolic Computation*, 44(12): 1676–1689, 2009.

[11] Jean-Charles Faugère and Ludovic Perret. High order derivatives and decomposition of multivariate polynomials. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 207–214, 2009.

[12] Alhussein Fawzi, Khurram Kozhasov, Micah Goldblum, Jens Behrmann, Chaoning Zhang, Fabian Fuchs, Po-Sen Huang, Lala Li, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

[13] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, 2017. doi: 10.18653/v1/w17-3207. URL `http://dx.doi.org/10.18653/v1/W17-3207`.

[14] Joachim von zur Gathen, Jaime Gutierrez, and Rosario Rubio. Multivariate polynomial decomposition. *Applicable Algebra in Engineering, Communication and Computing*, 14(1):11–31, 2003.

[15] Isabella Hernandez, Bryan C Watson, Marc J Weissburg, and Bert Bras. Using functional decomposition to bridge the design gap between desired emergent multi-agent-system resilience and individual agent design. *Systems Engineering*, 27(5):911–930, 2024.

[16] Liang Huang, Kai Zhao, and Mingbo Ma. When to finish? optimal beam search for neural text generation (modulo beam size), 2018. URL `https://arxiv.org/abs/1809.00069`.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[18] Jared Kaplan, Sam McCandlish, Tom Henighan, and Tom B. Brown. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[19] Andrej Karpathy. mingpt. `https://github.com/karpathy/minGPT`, 2020.

[20] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:2006.02974*, 2020.

[21] Hsuan-Po Lin, Jie-Hong R Jiang, and Ruei-Rung Lee. To sat or not to sat: Ashenhurst decomposition in a large scale. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 32–37. IEEE, 2008.

[22] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL `https://arxiv.org/abs/2503.20783`.

[23] Dinesh Manocha and John F Canny. Real time inverse kinematics for general 6r manipulators. In *ICRA*, pages 383–389, 1992.

[24] Matteo Mori, Chuankai Cheng, Brian R Taylor, Hiroyuki Okano, and Terence Hwa. Functional decomposition of metabolism allows a system-level quantification of fluxes and protein allocation towards specific metabolic functions. *Nature Communications*, 14(1):4161, 2023.

[25] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[26] Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes is it easier than expected to design efficient asymmetric cryptosystems? In *International Conference on Information and Communications Security*, pages 369–380. Springer, 1997.

[27] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[28] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks, 2016. URL `https://arxiv.org/abs/1511.06732`.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

[30] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

[31] Jinjuan She, Elise Belanger, and Caroline Bartels. Evaluating the effectiveness of functional decomposition in early-stage design: development and application of problem space exploration metrics. *Research in Engineering Design*, 35(3):311–327, 2024.

[32] Ewan Tempero, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, Diana Kirk, Juho Leinonen, Asma Shakil, Robert Sheehan, James Tizard, Yu-Cheng Tu, et al. On the comprehensibility of functional decomposition: An empirical study. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*, pages 214–224, 2024.

[33] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

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

[35] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

[36] Joachim Von Zur Gathen. Functional decomposition ofpolynomials: the tame case. *Journal of Symbolic Computation*, 9(3):281–299, 1990.

[37] Joachim Von Zur Gathen. Functional decomposition of polynomials: the wild case. *Journal of Symbolic Computation*, 10(5):437–452, 1990.

[38] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL `https://arxiv.org/abs/2211.00593`.

[39] Wolfram Research, Inc. `ComplexityFunction` – Wolfram Language Documentation. `https://reference.wolfram.com/language/ref/ComplexityFunction.html`, 1996. Accessed 12 May 2025.

[40] Shangwei Zhao, Ruyong Feng, and Xiao-Shan Gao. On functional decomposition of multivariate polynomials with differentiation and homogenization. *Journal of Systems Science and Complexity*, 25(2):329–347, 2012.

A Backward Synthetic Data Generation Algorithm

Our backward synthetic data generation in subsection 2.1 can be described as follows.

Algorithm 1 Backward Generation of Synthetic Training Data

Require: Coefficient range C_{inner}, C_{outer}; maximal degrees d_{inner}, d_{outer}; variable counts v_{inner}, v_{outer}; term limits t_{inner}, t_{outer}.

1: Generate outer polynomial g with v_{outer} variables, coefficients $\in C_{\text{outer}}$, degree $= d_{\text{outer}}$, and no more than t_{outer} monomial terms.

2: Generate v_{outer} inner polynomials $h_1, \ldots, h_{v_{\text{outer}}}$, where each h_i has v_{inner} variables, coefficients $\in C_{\text{inner}}$, degree $= d_{\text{inner}}$, and no more than t_{inner} monomial terms.

3: $f \leftarrow g(h_1, \ldots, h_{v_{\text{outer}}})$, i.e. substitute $h_1, \ldots, h_{v_{\text{outer}}}$ into g, expand and collect the monomial terms.

4: **return** $(f,\ g,\ h_1, \ldots, h_{v_{\text{outer}}})$

B Tokenization

We encode polynomials using prefix notation, with separate tokens for operators, digits, and variables. Each number includes its sign, so we only use addition, multiplication, and power operators. Subtraction is represented as addition with a negative sign. Each input sequence consists of the tokenized expanded polynomial f followed by a question mark token '?'. The target output format depends on the number of outer variables: for $v_{\text{outer}} = 1$, the target output is simply the tokenized inner polynomial h; for $v_{\text{outer}} > 1$, the target output begins with the tokenized outer polynomial g followed by each tokenized inner polynomial $h_1, \ldots, h_{v_{\text{outer}}}$, with all polynomials separated by a delimiter token '&'.

Below is an example of a tokenized training input 'x' and target output 'y':

$$
\begin{aligned}
x &: + * \text{P } 9\ 0\ a + * \text{N } 3\ 1\ 9\ \hat{}\ a\ \text{P } 2 + * \text{N } 3\ 6\ \hat{}\ a\ \text{P } 3 * \text{N } 1\ \hat{}\ a\ \text{P } 4\ ?\ + \text{N } 5 + * \text{P } 1\ 8\ a\ \hat{}\ a\ \text{P } 2\ \square \ldots \\
y &: \square\square\square\

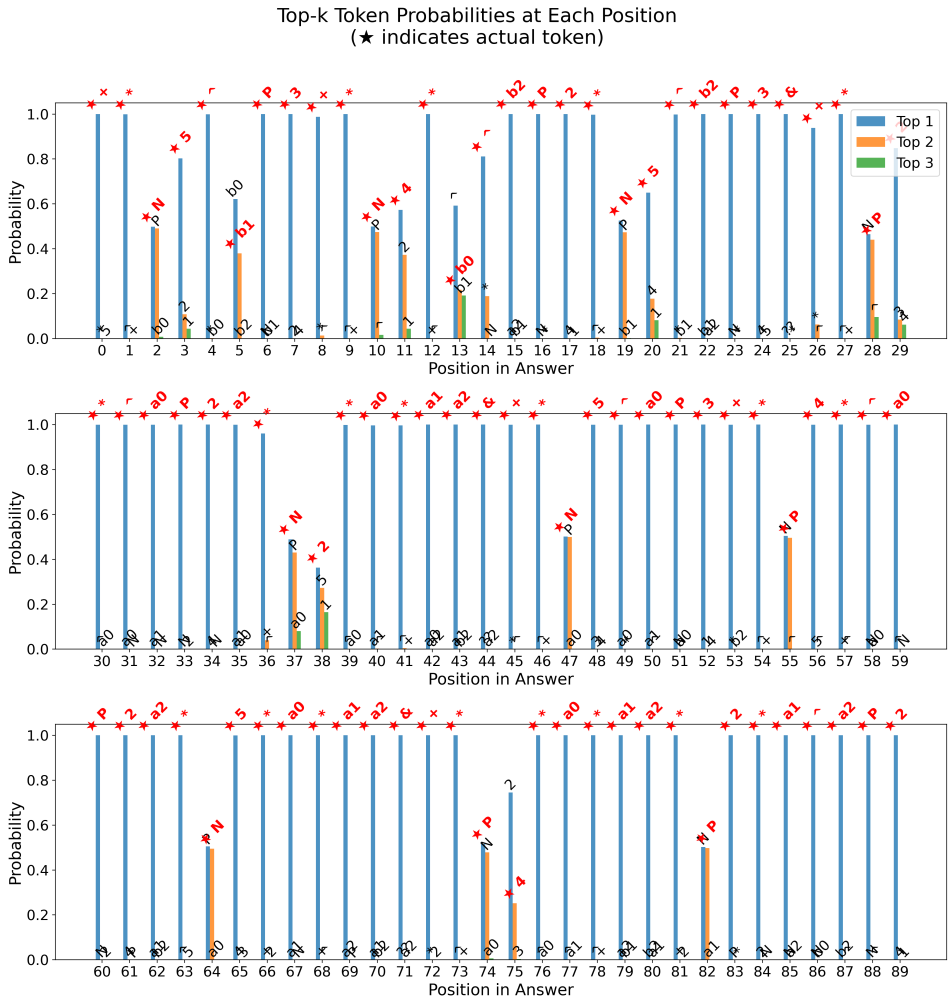

Figure 9: Top-3 probability for each token position in the answer sequence where

Answer: + * N 5 ^ b1 P 3 + * N 4 * b0 ^ b2 P 2 * N 5 ^ b2 P 3 & + * P 2 * ^ a0 P 2 a2 * N 2 * a0 * a1 a2 & + * N 5 ^ a0 P 3 + * P 4 * ^ a0 P 2 a2 * N 5 * a0 * a1 a2 & + * P 4 * a0 * a1 a2 * P 2 * a1 ^ a2 P 2

Question: + * P 6 2 5 ^ a0 P 9 + * N 1 5 0 0 * ^ a0 P 8 a2 + * P 1 8 7 5 * ^ a0 P 7 * a1 a2 + * P 1 2 0 0 * ^ a0 P 7 ^ a2 P 2 + * N 3 0 0 0 * ^ a0 P 6 * a1 ^ a2 P 2 + * P 1 8 7 5 * ^ a0 P 5 * ^ a1 P 2 ^ a2 P 2 + * N 3 2 0 * ^ a0 P 6 ^ a2 P 3 + * P 1 2 0 0 * ^ a0 P 5 * a1 ^ a2 P 3 + * N 1 6 2 8 * ^ a0 P 4 * ^ a1 P 2 ^ a2 P 3 + * P 4 3 3 * ^ a0 P 3 * ^ a1 P 3 ^ a2 P 3 + * N 1 2 8 * ^ a0 P 3 * ^ a1 P 2 ^ a2 P 4 + * N 3 5 2 * ^ a0 P 2 * ^ a1 P 3 ^ a2 P 4 + * N 3 2 * ^ a0 P 2 * ^ a1 P 2 ^ a2 P 5 + * N 2 0 8 * a0 * ^ a1 P 3 ^ a2 P 5 * N 4 0 * ^ a1 P 3 ^ a2 P 6 ?

C Example Ouput Logits and Effectiveness of the Beam Search

Figure 9 shows example top-3 probabilities for each token position in the answer sequence at temperature 1, using the layer-6, embedding dimension 512 model from our $\mathcal{D}_2$ experiments. Correct answers are highlighted in red. The visualization clearly illustrates that the model's primary source of confusion occurs in sign decisions, while it confidently predicts most of the other token types.

Table 4 quantifies this observation by showing the probability and accuracy statistics for different token types across our model architectures from $\mathcal{D}_2$. These statistics were computed using a test set of 1000 polynomial decomposition problems at temperature 1.

Table 4: Token Type Analysis Across Different Model Architectures

Token Type	Metric	4 Layers			6 Layers		
		256 dim	512 dim	768 dim	256 dim	512 dim	768 dim
Sign	Probability	0.489 ± 0.001	0.489 ± 0.001	0.493 ± 0.001	0.491 ± 0.001	0.490 ± 0.001	0.490 ± 0.001
	Accuracy	0.519 ± 0.006	0.531 ± 0.006	0.530 ± 0.006	0.522 ± 0.006	0.523 ± 0.006	0.521 ± 0.006
Operator	Probability	0.920 ± 0.002	0.915 ± 0.002	0.919 ± 0.002	0.927 ± 0.002	0.925 ± 0.002	0.925 ± 0.002
	Accuracy	0.937 ± 0.002	0.934 ± 0.002	0.935 ± 0.002	0.943 ± 0.002	0.941 ± 0.002	0.942 ± 0.002
Number	Probability	0.880 ± 0.002	0.870 ± 0.002	0.878 ± 0.002	0.890 ± 0.002	0.885 ± 0.002	0.884 ± 0.002
	Accuracy	0.901 ± 0.002	0.893 ± 0.003	0.897 ± 0.002	0.911 ± 0.002	0.905 ± 0.002	0.903 ± 0.002

Note: Values shown as mean $\pm$ standard error of the mean. The sign token probabilities are near-random, while operators and numbers show high confidence and accuracy.

As discussed in Section 2.2, our models achieve approximately 90% accuracy when predicting non-sign tokens, but exhibit near-random performance when choosing between positive and negative signs. This specific error pattern makes beam search particularly effective for our task.

The effectiveness of beam search stems from its ability to explore multiple sign configurations while preserving the high-confidence structural tokens. In probability terms, selecting a token with 0.1 probability instead of one with 0.9 probability is equivalent to making approximately 11 consecutive choices of a 0.45 probability token over a 0.55 probability token. Since our polynomial expressions typically contain fewer than 10 sign decisions, beam search with a width of approximately 30 can efficiently cover most viable sign permutations while maintaining the correct monomial structure identified with high confidence.

D Attention Score Analysis: Monomial Heads

Attention mechanism analysis has provided valuable insights into transformer model behaviors, with studies identifying specialized attention heads that serve specific functions. For example, [25] identified "Induction Heads" that play a crucial role in in-context learning, while [38] provided a comprehensive understanding of indirect object identification in GPT-2 Small.

In our analysis of attention patterns in polynomial decomposition models, we identified specialized attention heads that recognize the structure of polynomials, particularly focusing on monomial identification. We call these "Monomial Heads," and they appear consistently across all model sizes in our architecture scaling experiments ($\mathcal{D}_2$).

Monomial Heads manifest in two distinct patterns in our models. First, in layer 0, several attention heads consistently attend to tokens 1-5 positions behind the current position, as shown in the leftmost plot of Figure 10. Second, in layer 1, we observe specialized behavior where certain heads focus attention on specific tokens within each monomial of the input polynomial (middle plot), while others specifically attend to delimiter tokens in the decomposition output (rightmost plot).

We hypothesize that this represents a two-stage process: in the first layer, the model identifies key tokens that serve as indicators for each monomial by examining local context (1-5 tokens behind). In the second layer, tokens within each monomial attend to these indicator tokens to establish their monomial membership. While this pattern is most clear in the encoding of the input polynomial, the decomposition output shows evidence of boundary recognition, particularly at the transitions between inner functions marked by delimiter tokens.

E Computational Resources

This appendix provides detailed information about the computational resources used for each experiment in our study.

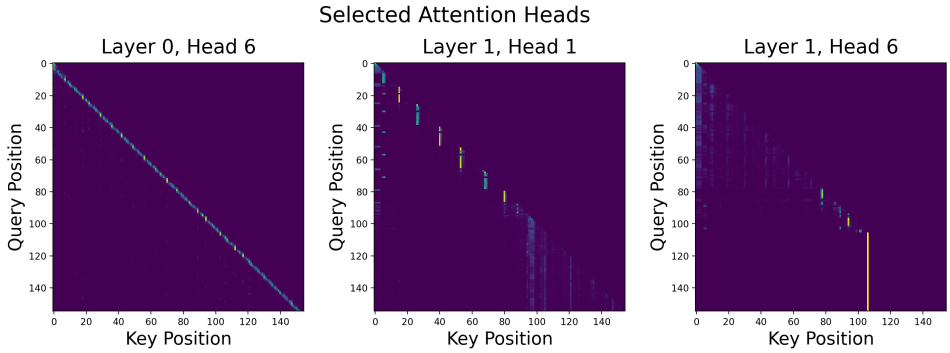

Figure 10: Attention score visualization of selected attention heads from our 6-layer transformer model with embedding dimension 768. The visualization shows attention patterns for a tokenized polynomial sequence and its decomposition.

Input polynomial: $+ * P\ 2\ 5\ 6 \wedge a0\ P\ 9 + * N\ 1\ 9\ 2 * \wedge a0\ P\ 8\ a1 + * P\ 4\ 8 * \wedge a0\ P\ 7 \wedge a1\ P\ 2 + * N\ 4 * \wedge a0\ P\ 6 \wedge a1\ P\ 3 + * N\ 6\ 4 * \wedge a0\ P\ 3 \wedge a1\ P\ 6 + * P\ 1\ 6 * \wedge a0\ P\ 2 \wedge a1\ P\ 7 * P\ 6\ 4 \wedge a1\ P\ 9\ ?$

Model's decomposition output: $+ * N\ 4 \wedge b0\ P\ 3 + * b0 \wedge b2\ P\ 2 * N\ 1 \wedge b2\ P\ 3\ \& + * N\ 4 \wedge a0\ P\ 3 * \wedge a0\ P\ 2\ a1\ \& + * N\ 3 \wedge a1\ P\ 3 + * N\ 2 * a1 \wedge a2\ P\ 2 * N\ 4 \wedge a2\ P\ 3\ \& * N\ 4 \wedge a1\ P\ 3$ The visualization reveals how different attention heads focus on specific structural elements when decomposing polynomials.

Table 5: Computational Resources by Experiment

Experiment	**Hardware**	**GPU-Hours**
Problem Complexity Scaling ($\mathcal{D}_1$)		
Degree Scaling (training)	NVIDIA A100	10
Variable Scaling (training)	NVIDIA A100	50
Beam Search Evaluation	NVIDIA A100	20
Architecture Scaling ($\mathcal{D}_2$)		
Layer-Dimension Variations (training)	NVIDIA A100	200
Attention Heads Variations (training)	NVIDIA A100	40
Beam Search Evaluation	NVIDIA A100	50
Distribution Adaptation ($\mathcal{D}_3$)		
Initial Distribution Training	NVIDIA A100	5
Distribution Adaptation	NVIDIA A100	2
BGRPO		
BGRPO Training	NVIDIA 4090	30
Beam Search Evaluation	NVIDIA 4090	10

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.

- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims of the abstract and introduction are justified by experimental evidence, provided in section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a separate paragraph in Section 5 on limitations of our research.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our work primarily presents an empirical study on transformers applied to the polynomial decomposition problem, and more generally, the functional decomposition problem, and is not theoretical.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We detail our method and full experimental settings: see Section 2, Section 3, Appendix A, and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed

instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and data will be made publicly available upon publication. We are currently preparing this repository to ensure it is well-documented and user-friendly.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify all training and test details in section 3, which describes our synthetic dataset setup, architecture configuration, supervised learning details, and BGRPO setup.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report error bars for our primary results in Figure 5, where we conducted multiple runs to establish confidence intervals. For the remaining experiments, computational resource constraints prevented us from performing multiple runs across all model configurations. Instead, we ensured reliability by using sufficiently large test sets to mitigate variance.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We indicate amount of GPU-Hour to train and evaluate each experiment in relevant subsection in Section E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We followed the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work focuses on transformers applied to the polynomial decomposition problem, which is foundational research not tied to immediate societal applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: Our research carries no significant risk of misuse, as our data is synthetically generated using Mathematica (see section 2.1) and models are trained for polynomial decomposition, a problem with no direct opportunities for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we credited them properly (see references).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not include crowdsourcing and research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing and research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method was developed independently without LLM involvement.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.