# Learning-Guided Flip-Graph Search for Low-Rank Polynomial Multiplication

**Dharunish Yugeswardeenoo** [1]

## Abstract

Polynomial multiplication is a fundamental bilinear operation whose bilinear algorithms correspond to rank-1 decompositions of a structure tensor, with scalar multiplication count equal to the number of rank-1 terms. We study algorithm discovery as learning-guided search over a tensor flip graph, whose nodes are exact decompositions and edges are local algebra-preserving transformations. We propose POLYDISCOVER, a learning-guided Monte Carlo Tree Search (MCTS) framework that uses a Transformer policy/value model to guide this search. Starting from the schoolbook FOIL decomposition, POLYDISCOVER navigates orbit flips, zero-sum additions, and recursive rank-reducing rewrites. Under the coefficient restriction {-1,0,1}, all candidate algorithms are verified by reconstruction of the polynomial tensor. For degrees 1–100, the discovered decompositions achieve fewer scalar multiplications than the published Weimerskirch–Paar counts in 84 cases and match them in the remaining 16 under the coefficient restriction over general rings. Structural analysis shows that the discovered algorithms form a recursive subset-sum basis where scalar multiplications are products of overlapping coefficient sums, and output coefficients are recovered by signed inclusion–exclusion.

## 1. Introduction

Recent work such as AlphaTensor has shown that reinforcement learning can search the space of tensor decompositions and discover nontrivial matrix multiplication algorithms (Fawzi et al., 2022). This raises a natural question: can learning-guided search discover efficient algorithms for other bilinear maps such as polynomial multiplication under arithmetic constraints over general rings?

[1]Central Bucks High School South, Warrington, PA, USA. Correspondence to: Dharunish Yugeswardeenoo <dharyugi@gmail.com>.

The $3^{rd}$ *AI for Math Workshop* at the $43^{rd}$ International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

Given two polynomials, the schoolbook method multiplies every pair of coefficients. Algebraically, this operation is a bilinear map and can therefore be encoded as a structure tensor. A multiplication algorithm is then a decomposition of this tensor into rank-1 terms. In this work, every coefficient in each rank-1 term is restricted to the alphabet $\{-1, 0, 1\}$, excluding rational interpolation constants and divisions. This makes the search space discrete, exact, valid over arbitrary rings, and implementation-friendly.

A naive learning formulation would ask a policy network to emit all rank-1 terms directly. However, most candidate decompositions are invalid, many syntactically different terms are algebraically equivalent, and exact tensor rank is computationally hard in general (Håstad, 1990). Instead, we formulate discovery as navigation over a tensor flip graph. The search begins from a known valid decomposition, the schoolbook decomposition, and moves through a graph whose edges are local transformations that preserve the represented tensor. A neural policy/value model scores candidate transformations, while Monte Carlo Tree Search (MCTS) supplies lookahead over short trajectories.

Karatsuba's algorithm showed that degree-1 polynomial multiplication can be performed with three scalar multiplications rather than four (Karatsuba & Ofman, 1963). Toom's method generalizes interpolation-based multiplication by evaluating operands at several points, multiplying the evaluations, and interpolating the result (Toom, 1963). FFT-based and Schönhage–Strassen-style methods achieve stronger asymptotic behavior using transform-domain arithmetic, roots of unity, modular transforms, or ring extensions. Our setting is narrower but more discrete. We search for finite-degree bilinear decompositions whose input and output coefficient vectors all lie in $\{-1, 0, 1\}$. We use the Weimerskirch–Paar generalized Karatsuba construction as the primary multiplication-count baseline because it targets low-multiplication symbolic formulas with small integer coefficients (Weimerskirch & Paar, 2006).

The classical parity split writes operands as $A_0(x^2) + xA_1(x^2)$ and $B_0(x^2) + xB_1(x^2)$, then combines $A_0B_0$, $A_1B_1$, and $(A_0 + A_1)(B_0 + B_1)$ by inclusion–exclusion. This is the standard Karatsuba-style odd–even decomposition described in modern arithmetic references (Brent & Zimmermann, 2010). Cryptographic overlap-free Karatsuba

variants also use parity splitting, but they primarily optimize XOR depth or circuit structure over $\mathbb{F}_2[x]$ rather than general-ring scalar-multiplication rank (Fan & Hasan, 2007; Fan, 2022).

Finding low-rank decompositions is central to algebraic complexity, but exact tensor rank is hard in general (Håstad, 1990). Kauers and Moosbauer introduced flip graphs as a structured search space over tensor decompositions of a fixed bilinear map, using random walks over local flip and reduction moves to discover improved schemes for several small matrix multiplication formats (Kauers & Moosbauer, 2022). Chen and Kauers extended the flip-graph framework to the polynomial multiplication tensor and proved that, over sufficiently large fields, the flip graph contains a bounded-length path from the standard representation to a representation of low rank. They also note that automated search over larger fields is complicated by the freedom to redistribute scalar factors across rank-one components (Chen & Kauers, 2025). The present work addresses this search obstruction in a restricted arithmetic model. By constraining all coefficient vectors to $\{-1, 0, 1\}$, we eliminate arbitrary field-scalar redistribution, leaving only finite sign and symmetry orbits that can be canonicalized. Within this discrete search space, we replace random walks with an orbit-compressed Transformer policy/value model inside MCTS, and extend the move set with recursive rank-reducing rewrites specialized to the polynomial multiplication tensor.

The learning architecture follows the policy/value and MCTS pattern of AlphaZero-style systems (Silver et al., 2018). It represents each state as a sequence of canonical orbit representatives, processes this sequence with a Transformer encoder (Vaswani et al., 2017), and uses a quantile value head inspired by distributional reinforcement learning (Dabney et al., 2018) to bias search toward rare high-reward rewrite trajectories.

Our contributions are: (i) we formulate exact polynomial multiplication over $\{-1, 0, 1\}$ as search over a tensor flip graph of equivalent decompositions, where local moves preserve correctness by construction (ii) we introduce an orbit-compressed Transformer-guided MCTS procedure that searches over algebra-preserving flips and rewrites rather than directly generating full decompositions (iii) we report verified degree-1 to 100 decompositions whose rank matches WP in 16 cases and improves WP in 84 cases and (iv) we identify a recursive subset-sum and signed inclusion–exclusion structure that explains the verified formulas and their multiplication-count recurrence over the tested range.

## 2. Methods

### 2.1. Polynomial multiplication tensor and search objective

Let $A(x) = \sum_{i=0}^{d} a_i x^i$ and $B(x) = \sum_{j=0}^{d} b_j x^j$. Their product $C(x) = A(x)B(x) = \sum_{k=0}^{2d} c_k x^k$ satisfies

$$c_k = \sum_{i+j=k} a_i b_j.$$

The corresponding polynomial multiplication tensor is

$$(\mathcal{T}_d)_{i,j,k} = \mathbf{1}[i + j = k], \quad \mathcal{T}_d \in \{0, 1\}^{(d+1)\times(d+1)\times(2d+1)}. \tag{1}$$

A rank-1 term $u \otimes v \otimes w$ represents the operation

$$\left(\sum_i u_i a_i\right)\left(\sum_j v_j b_j\right)$$

followed by adding the product to output coefficients according to $w$. A valid algorithm with $R$ scalar multiplications is therefore an exact decomposition

$$\begin{aligned} \mathcal{T}_d &= \sum_{r=1}^{R} u_r \otimes v_r \otimes w_r, \\ u_r, v_r &\in \{-1, 0, 1\}^{d+1}, \\ w_r &\in \{-1, 0, 1\}^{2d+1}. \end{aligned} \tag{2}$$

The search objective is to minimize $R$ subject to exact equality in Equation (2) and the alphabet constraint $\{-1, 0, 1\}$.

### 2.2. Tensor flip graph and initial decomposition

The initial state is the FOIL decomposition

$$\{e_i \otimes e_j \otimes e_{i+j} : 0 \le i, j \le d\},$$

which has $(d + 1)^2$ rank-1 terms. We define a tensor flip graph whose nodes are decompositions equivalent to $\mathcal{T}_d$ and whose edges are local symbolic transformations that preserve the represented tensor. PolyDiscover searches this graph for shorter equivalent decompositions. Candidate solutions are accepted only after exact reconstruction of $\mathcal{T}_d$ and entry-by-entry equality checking.

### 2.3. Orbit-based state representation

Many rank-1 terms represent the same algebraic object up to symmetry. For square polynomial multiplication, we canonicalize each term by enumerating optional input swaps, simultaneous reversal of coefficient order, and the sign symmetries

$$(u, v, w), \quad (u, -v, -w), \quad (-u, v, -w), \quad (-u, -v, w). \tag{3}$$

The canonical representative is the lexicographically smallest element in this orbit. A state is encoded as a sequence $S = \{(\bar{t}_1, \rho_1), \ldots, (\bar{t}_H, \rho_H)\}$, where $\bar{t}_h$ is an orbit representative and $\rho_h$ is its normalized multiplicity. Input-swap symmetry is used only for equal-size inputs and the odd recurrence handles unequal parity lengths by zero-padding the shorter odd part into the larger middle product.

### 2.4. Local symbolic moves

**Orbit flips.** If two terms share two factors, for example $(u, v, w_1)$ and $(u, v, w_2)$, then

$$u \otimes v \otimes w_1 + u \otimes v \otimes w_2 = u \otimes v \otimes (w_1 + w_2). \quad (4)$$

The search may transfer a value $\delta \in \{-1, +1\}$ between matching coordinates of $w_1$ and $w_2$, as long as both vectors remain in $\{-1, 0, 1\}$. Analogous flips apply when the shared pair is $(u, w)$ or $(v, w)$.

**Recursive rewrites.** Rewrites are the main rank-reduction operator. The search scans the current state for FOIL sub-blocks and replaces them with lower-rank kernels. A kernel for length $L$ is built recursively by splitting $L = m + k$, composing kernels for $m$ and $k$, and sanitizing any intermediate coefficient outside $\{-1, 0, 1\}$ by splitting it into legal $\pm 1$ components.

**Zero-sum additions.** Zero-sum additions are rank-increasing exploration moves. For any admissible rank-1 term $\tau = u \otimes v \otimes w$ with $u, v, w \in \{-1, 0, 1\}^*$, the search may add $\tau$ and an inverse term $-\tau$:

$$\tau + (-\tau) = 0. \quad (5)$$

The represented tensor is unchanged, but the added terms may create new shared-factor pairs, allowing orbit flips to redistribute coefficients and expose FOIL-like sub-blocks or cancellation patterns. Zero-sum additions therefore serve as temporary rank-increasing escape steps that preserve correctness while opening paths to future rank-reducing moves.

### 2.5. Policy/value model and MCTS

Each orbit row is linearly projected to a $d_{\text{model}}$-dimensional embedding and processed by a Transformer encoder. Candidate actions are encoded as $[\texttt{tflag}, \text{flatten}(\text{rep}_a), \text{flatten}(\text{rep}_b)]$, where `tflag` identifies the move type and $\text{rep}_a, \text{rep}_b$ are canonical orbit representatives associated with the move. The policy head scores every candidate action conditioned on the state context. The value head predicts 32 quantiles, and the top quartile is averaged to obtain an optimistic value estimate. This top-quartile aggregation is a risk-seeking search heuristic which biases MCTS toward rare high-payoff moves. Lookahead is important because local progress can be misleading.

**Algorithm 1** Self-play discovery for degree $d$

Initialise $S_0$ as the FOIL decomposition of $\mathcal{T}_d$
**for** $t = 0, 1, \ldots, T-1$ **do**
  Generate legal symbolic moves from $S_t$
  Run MCTS using the Transformer policy/value model
  Store $(S_t, \pi_t)$, where $\pi_t$ is the root visit distribution
  Sample and apply an action to obtain $S_{t+1}$
**end for**
Verify exact equality $\sum_{r \in S_T} u_r \otimes v_r \otimes w_r = \mathcal{T}_d$
Train policy on $\pi_t$ and value on $1 - |S_T|/|S_0|$

Useful trajectories may require neutral or temporarily rank-increasing moves before a large rewrite becomes possible. For each state, MCTS expands legal actions generated by the symbolic move generator. Selection uses

$$\text{UCB}(s, a) = Q(s, a) + C_{\text{PUCT}}\, P(s, a)\, \frac{\sqrt{N(s)}}{1 + N(s, a)}, \quad (6)$$

where $P$ is the learned action prior, $Q$ is the running value, and $N$ denotes visit counts.

### 2.6. Learning objective and self-play procedure

Self-play produces trajectories $S_0 \to S_1 \to \cdots \to S_T$, where $S_0$ is FOIL and $|S_T|$ is the final rank. At each state, MCTS returns a visit distribution over legal actions. The policy is trained to predict this distribution and the value head is trained to predict the final rank-reduction score $V(S_T) = 1 - |S_T|/|S_0|$. Algorithm 1 summarizes the procedure.

Hyperparameters used in all discovery runs are listed in Table 1.

## 3. Results

### 3.1. Verified rank improvements across degrees 1–100

We evaluate discovery quality by verified rank: the number of rank-1 terms in an exact decomposition of the polynomial multiplication tensor. For each degree $d \in \{1, \ldots, 100\}$, PolyDiscover starts from the schoolbook FOIL decomposition and records the lowest-rank exact decomposition found by learning-guided flip-graph search.

PolyDiscover finds verified schemes for every tested degree. Across degrees 1–100, the discovered algorithms match WP in 16 cases and improve WP in 84 cases, giving strictly smaller verified rank under the same $\{-1, 0, 1\}$ general-ring arithmetic model. The full degree-by-degree comparison is given in Appendix A, Table 2, and the corresponding savings are visualized in Figure 1. The discovered scheme is never worse than WP across the tested range.

*Table 1.* Configuration used for discovery runs.

| Component | Value |
|---|---|
| Seed | 0 |
| State encoder | Transformer encoder |
| $d_{\text{model}}$ | 256 |
| Layers / heads | 4 / 8 |
| Feedforward width | $4d_{\text{model}}$ |
| Dropout / activation | 0.1 / GELU |
| Value head | 32 quantiles, top 25% average |
| MCTS simulations / move | 100 |
| $C_{\text{PUCT}}$ | 2 |
| Max trajectory depth | 64 |
| Max candidate moves | 32 |
| Self-play iterations | 1,000 |
| Optimizer | Adam |
| Learning rate | $10^{-5}$ |
| Weight decay | $10^{-4}$ |
| Gradient clip | 0.5 |
| Batch size | 64 |

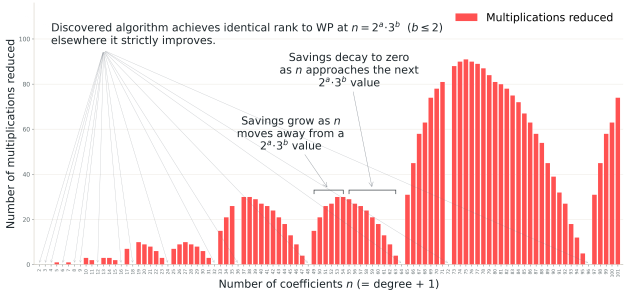


*Figure 1.* Per-degree multiplication savings of the discovered scheme over the Weimerskirch–Paar (WP) baseline. For each polynomial degree $d \in \{1, \ldots, 100\}$, the bar shows $\Delta = \text{WP} - \text{Scheme}$, the number of scalar multiplications saved by the discovered scheme relative to WP under the same $\{-1, 0, 1\}$ coefficient restriction. Positive values mean the discovered scheme uses strictly fewer scalar multiplications; zero values mark matching cases.

## 4. Discussion

The algebraic structure of the decompositions discovered by PolyDiscover reveals a compact *subset-sum* interpretation: scalar multiplications are products of overlapping coefficient sums, and output coefficients are recovered by signed inclusion–exclusion.

### 4.1. Subset-sum decompositions

For a subset $S \subseteq \{0, \ldots, d\}$, define

$$Q(S) = \left(\sum_{i \in S} a_i\right)\left(\sum_{j \in S} b_j\right). \tag{7}$$

Expanding gives

$$Q(S) = \sum_{i \in S}\sum_{j \in S} a_i b_j. \tag{8}$$

Thus one multiplication $Q(S)$ contains many coefficient-pair products at once.

**Definition 4.1.** A symmetric subset-sum scheme for degree $d$ consists of subsets

$$S_0, S_1, \ldots, S_{r-1} \subseteq \{0, \ldots, d\}$$

and decoding coefficients

$$\lambda_{k,\ell} \in \{-1, 0, 1\}, \qquad 0 \le k \le 2d, \quad 0 \le \ell < r,$$

such that

$$c_k = \sum_{\ell=0}^{r-1} \lambda_{k,\ell} Q(S_\ell). \tag{9}$$

The role of the decoding coefficients is to cancel the extra coefficient-pair interactions introduced by the large subset products.

**Proposition 4.2.** *Let $\chi_\ell(i)$ be the indicator of $i \in S_\ell$. A subset-sum scheme is correct if and only if, for every $p, q \in \{0, \ldots, d\}$ and every $k \in \{0, \ldots, 2d\}$,*

$$\sum_{\ell=0}^{r-1} \lambda_{k,\ell}\chi_\ell(p)\chi_\ell(q) = \mathbf{1}[p + q = k]\,. \tag{10}$$

*Proof.* Expanding the decoded coefficient gives

$$\sum_{\ell=0}^{r-1} \lambda_{k,\ell} Q(S_\ell) = \sum_{\ell=0}^{r-1} \lambda_{k,\ell} \sum_{p=0}^{d}\sum_{q=0}^{d} \chi_\ell(p)\chi_\ell(q) a_p b_q \tag{11}$$

$$= \sum_{p=0}^{d}\sum_{q=0}^{d} \left(\sum_{\ell=0}^{r-1} \lambda_{k,\ell}\chi_\ell(p)\chi_\ell(q)\right) a_p b_q. \tag{12}$$

This equals $c_k = \sum_{p+q=k} a_p b_q$ exactly when the inner coefficient is 1 for pairs satisfying $p+q = k$ and 0 otherwise. □

Equation (10) gives a compact certificate for every reported formula: the subsets $S_\ell$ and the integer decoding matrix $(\lambda_{k,\ell})$ can be checked by exact arithmetic.

### 4.2. Relation to the Weimerskirch–Paar baseline

The Weimerskirch–Paar generalized Karatsuba construction can also be viewed as a subset-sum method, but with a restricted subset family. For $N = d + 1$ input coefficients, its one-step form uses singleton products

$$D_i = a_i b_i, \qquad 0 \le i < N, \tag{13}$$

and pair-sum products

$$D_{s,t} = (a_s + a_t)(b_s + b_t), \qquad 0 \le s < t < N. \tag{14}$$

Since

$$D_{s,t} = a_s b_s + a_s b_t + a_t b_s + a_t b_t, \tag{15}$$

the cross terms are recovered by

$$a_s b_t + a_t b_s = D_{s,t} - D_s - D_t. \tag{16}$$

The one-step multiplication count is therefore

$$r_{\mathrm{WP},1}(N) = N + \binom{N}{2} = \frac{N(N+1)}{2}. \tag{17}$$

The discovered schemes differ by using larger overlapping subsets, not only singletons and pairs. A single product $Q(S)$ with $|S| > 2$ mixes many coefficient interactions. Signed combinations of such products can isolate the same anti-diagonal information that WP obtains from multiple pair products. In this sense, WP uses a local pair basis, while the discovered schemes use a larger overlapping basis.

### 4.3. Recursive rank pattern

Let $R(n)$ denote the number of scalar multiplications used by the discovered scheme for multiplying two polynomials with $n$ coefficients. The verified decompositions in our tested range follow the recurrence

$$R(1) = 1, \tag{18}$$
$$R(2m) = 3R(m), \tag{19}$$
$$R(2m+1) = 2R(m+1) + R(m) - 1. \tag{20}$$

The even case is the usual Karatsuba-style parity split. Write

$$A(x) = A_0(x^2) + xA_1(x^2), \tag{21}$$
$$B(x) = B_0(x^2) + xB_1(x^2). \tag{22}$$

Then

$$A(x)B(x) = P_0(x^2) + x(P_\Sigma - P_0 - P_1)(x^2) + x^2 P_1(x^2), \tag{23}$$

where

$$P_0 = A_0 B_0, \tag{24}$$
$$P_1 = A_1 B_1, \tag{25}$$
$$P_\Sigma = (A_0 + A_1)(B_0 + B_1). \tag{26}$$

Thus a length-$2m$ multiplication reduces to three length-$m$ multiplications.

The odd case is where the discovered structure saves one multiplication. For $n = 2m + 1$, the even-indexed part has length $m + 1$, while the odd-indexed part has length $m$. A direct uneven Karatsuba split would require

$$2R(m+1) + R(m)$$

multiplications. The discovered formulas fuse one shared corner product between the two length-$(m + 1)$ recursive calls, giving the observed recurrence

$$R(2m+1) = 2R(m+1) + R(m) - 1.$$

The matching cases against WP occur at smooth recursive split sizes of the form $n = 2^a 3^b$ with $b \le 2$, where the published WP construction already matches the same low-rank subset-sum structure. Away from these smooth lengths, the discovered recursive subset basis gives a smaller multiplication basis, producing the positive savings seen in Figure 1.

## 5. Limitations and Future Work

The reported results are verified upper bounds under an exact discrete arithmetic model, not proofs of tensor-rank optimality. The coefficient alphabet restriction $\{-1, 0, 1\}$ is useful for exactness and implementability, but it excludes algorithms that rely on rational interpolation constants, roots of unity, or field extensions. The current study focuses on equal-degree univariate polynomial multiplication for degrees 1–100. Future work could be on generalization to unequal input lengths, and relaxing the coefficient restriction. Another direction is hardware-aware discovery: rather than optimizing only scalar multiplication count, the reward could include depth, memory traffic, parallelism, latency, or measured energy on a target device.

## Software and Data

All verified decompositions reported in this paper are released as supplementary material for review. The full degree-by-degree comparison underlying Figure 1 and Table 2 is included in Section A.

## Impact Statement

This work aims to advance machine-learning-assisted algorithm discovery for exact symbolic computation. Potential positive impacts include improved finite-degree polynomial multiplication algorithms and a better understanding of how learning-guided search can navigate constrained algebraic spaces. We do not identify any direct foreseeable societal harms from this work. The main risks are indirect: improved arithmetic algorithms may be used in downstream systems with both beneficial and harmful applications, as is generally true for advances in computational efficiency.

## References


Brent, R. P. and Zimmermann, P. *Modern Computer Arithmetic*. Cambridge University Press, 2010. doi: 10.1017/CBO9780511921698.

Chen, S. and Kauers, M. Flip graphs for polynomial multiplication, 2025. URL https://arxiv.org/abs/2502.06264.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 2892–2901, 2018.

Fan, H. Low-delay polynomial multipliers based on overlap-free Karatsuba formulae. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(12):4831–4842, 2022.

Fan, H. and Hasan, M. A. A new approach to subquadratic space complexity parallel multipliers for extended binary fields. *IEEE Transactions on Computers*, 56(2):224–233, 2007. doi: 10.1109/TC.2007.30.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., Ruiz, F. J. R., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., and Kohli, P. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610: 47–53, 2022. doi: 10.1038/s41586-022-05172-4.

Håstad, J. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990. doi: 10.1016/0196-6774(90)90014-6.

Karatsuba, A. A. and Ofman, Y. P. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7: 595–596, 1963.

Kauers, M. and Moosbauer, J. Flip graphs for matrix multiplication, 2022. URL https://arxiv.org/abs/2212.01175.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Toom, A. L. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Weimerskirch, A. and Paar, C. Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Paper 2006/224, 2006. URL https://eprint.iacr.org/2006/224.

## A. Full Degree-by-Degree Rank Table

*Table 2.* **Full degree-by-degree rank comparison of the discovered scheme against the Weimerskirch–Paar baseline.** $d$ is polynomial degree, $n = d + 1$ is the number of input coefficients, WP is the published Weimerskirch–Paar rank, Scheme is the verified rank of the discovered decomposition, and $\Delta = \mathrm{WP} - \mathrm{Scheme}$. Positive $\Delta$ indicates that the discovered scheme uses fewer scalar multiplications. The table comprises four panels of 25 rows each, read in column-major order across degrees 1–100.

| $d$ | $n$ | WP | Scheme | $\Delta$ | $d$ | $n$ | WP | Scheme | $\Delta$ | $d$ | $n$ | WP | Scheme | $\Delta$ | $d$ | $n$ | WP | Scheme | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 3 | 0 | 26 | 27 | 210 | 200 | 10 | 51 | 52 | 594 | 567 | 27 | 76 | 77 | 1185 | 1096 | 89 |
| 2 | 3 | 6 | 6 | 0 | 27 | 28 | 216 | 207 | 9 | 52 | 53 | 618 | 588 | 30 | 77 | 78 | 1197 | 1110 | 87 |
| 3 | 4 | 9 | 9 | 0 | 28 | 29 | 228 | 220 | 8 | 53 | 54 | 630 | 600 | 30 | 78 | 79 | 1209 | 1125 | 84 |
| 4 | 5 | 15 | 14 | 1 | 29 | 30 | 234 | 228 | 6 | 54 | 55 | 642 | 613 | 29 | 79 | 80 | 1215 | 1134 | 81 |
| 5 | 6 | 18 | 18 | 0 | 30 | 31 | 240 | 237 | 3 | 55 | 56 | 648 | 621 | 27 | 80 | 81 | 1263 | 1183 | 80 |
| 6 | 7 | 24 | 23 | 1 | 31 | 32 | 243 | 243 | 0 | 56 | 57 | 672 | 646 | 26 | 81 | 82 | 1287 | 1209 | 78 |
| 7 | 8 | 27 | 27 | 0 | 32 | 33 | 291 | 276 | 15 | 57 | 58 | 684 | 660 | 24 | 82 | 83 | 1311 | 1236 | 75 |
| 8 | 9 | 36 | 36 | 0 | 33 | 34 | 315 | 294 | 21 | 58 | 59 | 696 | 675 | 21 | 83 | 84 | 1323 | 1251 | 72 |
| 9 | 10 | 45 | 42 | 3 | 34 | 35 | 339 | 313 | 26 | 59 | 60 | 702 | 684 | 18 | 84 | 85 | 1347 | 1280 | 67 |
| 10 | 11 | 51 | 49 | 2 | 35 | 36 | 324 | 324 | 0 | 60 | 61 | 714 | 701 | 13 | 85 | 86 | 1359 | 1296 | 63 |
| 11 | 12 | 54 | 54 | 0 | 36 | 37 | 375 | 345 | 30 | 61 | 62 | 720 | 711 | 9 | 86 | 87 | 1371 | 1313 | 58 |
| 12 | 13 | 66 | 63 | 3 | 37 | 38 | 387 | 357 | 30 | 62 | 63 | 726 | 722 | 4 | 87 | 88 | 1413 | 1323 | 90 |
| 13 | 14 | 72 | 69 | 3 | 38 | 39 | 399 | 370 | 29 | 63 | 64 | 729 | 729 | 0 | 88 | 89 | 1446 | 1356 | 90 |
| 14 | 15 | 78 | 76 | 2 | 39 | 40 | 405 | 378 | 27 | 64 | 65 | 825 | 794 | 31 | 89 | 90 | 1464 | 1374 | 90 |
| 15 | 16 | 81 | 81 | 0 | 40 | 41 | 429 | 403 | 26 | 65 | 66 | 873 | 828 | 45 | 90 | 91 | 1425 | 1393 | 32 |
| 16 | 17 | 105 | 98 | 7 | 41 | 42 | 441 | 417 | 24 | 66 | 67 | 921 | 863 | 58 | 91 | 92 | 1431 | 1404 | 27 |
| 17 | 18 | 108 | 108 | 0 | 42 | 43 | 453 | 432 | 21 | 67 | 68 | 945 | 882 | 63 | 92 | 93 | 1443 | 1425 | 18 |
| 18 | 19 | 129 | 119 | 10 | 43 | 44 | 459 | 441 | 18 | 68 | 69 | 993 | 919 | 74 | 93 | 94 | 1449 | 1437 | 12 |
| 19 | 20 | 135 | 126 | 9 | 44 | 45 | 471 | 458 | 13 | 69 | 70 | 1017 | 939 | 78 | 94 | 95 | 1455 | 1450 | 5 |
| 20 | 21 | 147 | 139 | 8 | 45 | 46 | 477 | 468 | 9 | 70 | 71 | 1041 | 960 | 81 | 95 | 96 | 1458 | 1458 | 0 |
| 21 | 22 | 153 | 147 | 6 | 46 | 47 | 483 | 479 | 4 | 71 | 72 | 972 | 972 | 0 | 96 | 97 | 1554 | 1523 | 31 |
| 22 | 23 | 159 | 156 | 3 | 47 | 48 | 486 | 486 | 0 | 72 | 73 | 1101 | 1013 | 88 | 97 | 98 | 1602 | 1557 | 45 |
| 23 | 24 | 162 | 162 | 0 | 48 | 49 | 534 | 519 | 15 | 73 | 74 | 1125 | 1035 | 90 | 98 | 99 | 1650 | 1592 | 58 |
| 24 | 25 | 186 | 179 | 7 | 49 | 50 | 558 | 537 | 21 | 74 | 75 | 1149 | 1058 | 91 | 99 | 100 | 1674 | 1611 | 63 |
| 25 | 26 | 198 | 189 | 9 | 50 | 51 | 582 | 556 | 26 | 75 | 76 | 1161 | 1071 | 90 | 100 | 101 | 1722 | 1648 | 74 |