Enhancing Graph Embedding Expressivity through Cycle Counting and Graph Products

Maximilian Seeliger
maximilian.seeliger@tuwien.ac.at

November 2023

Contents

Abstract

This paper presents a novel approach to enhance graph embedding expressivity by integrating cycle counting with graph product operations, including Cartesian, Strong, and Tensor products. The study focuses on applying these techniques to a variety of small graphs and analyzing their effectiveness in improving the distinctiveness of graph embeddings. The research identifies specific combinations of graph products and factor graphs that significantly reduce collisions, i.e., instances where different graphs yield identical embeddings. The findings highlight a convergence trend, indicating an optimal range for factor graph size that maximizes the effectiveness of graph embeddings. The study demonstrates that certain combinations, notably the Strong product with path graphs, are highly effective in reducing collisions by about 90% to 99%. Additionally, the research compares these methods with established baselines, such as the Weisfeiler-Lehman method, revealing that the proposed approach can surpass this traditional technique in certain configurations. This work provides valuable insights for optimizing graph embedding techniques and suggests new avenues for further exploration in the field of graph embeddings.

1 Introduction

Graph embeddings have become a pivotal tool in understanding and analyzing the complex structures of graphs. The process involves mapping graph structures to a vector space, where the relationships between vectors reflect the properties and relationships of the underlying graph entities. Cycle counting, an approach that emphasizes the enumeration of cycles within a graph, has shown promise in capturing the intricate topology of graphs. However, enhancing the expressivity of such embeddings remains a challenge. This study introduces the use of graph products - Cartesian, Strong, and Tensor - as a means to augment the expressivity of cycle-based graph embeddings.

Graph products are operations that combine two graphs to produce a new graph, which encapsulates characteristics of both parent graphs. These products are crucial in exploring complex graph interactions and are instrumental in unveiling deeper structural properties and relationships.

1.1 Cartesian Product

The Cartesian product of two graphs G and H, denoted as $G \Box H$, forms a graph where each vertex represents a pair of vertices, one from G and one from H. An edge is formed between two such vertices if the corresponding vertices in G are the same and their counterparts in H are adjacent, or if their counterparts in G are adjacent and those in H are the same. This product tends to preserve the original graph structures, leading to an embedding that reflects both individual and combined properties.

1.2 Strong Product

The Strong product, denoted as $G \boxtimes H$, is a blend of the Cartesian and Tensor products. It connects two vertices if they are connected in the Cartesian product, or if their corresponding vertices are adjacent in both G and H. This product creates a denser graph than the Cartesian product, potentially offering a richer and more intricate embedding.

1.3 Tensor Product

Also known as the Kronecker product, the Tensor product $G \otimes H$ forms a graph where vertices are connected if their corresponding vertices in G and H are both connected. This product tends to create a graph that is significantly denser than the original graphs, often leading to a more complex embedding that can capture deeper inter-graph relationships.

1.4 Visualization

To better illustrate the impact of these graph products on graph structure, let's generate visualizations for each product using simple graphs. For this purpose, we can use two basic graphs, such as a path graph with 3 nodes (P_3) and a complete graph with 3 nodes (K_3) (see figure 1), and visualize their Cartesian, Strong, and Tensor products (see figure 2).

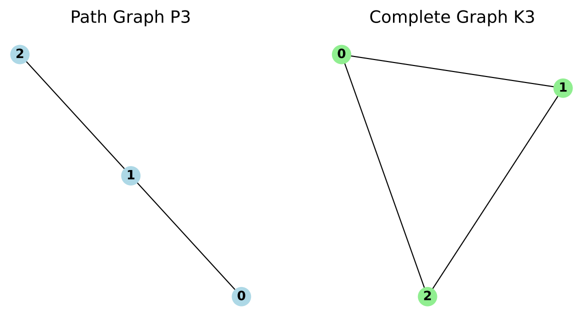

Figure 1: Original Graphs: Path Graph P_3 and Complete Graph K_3

2 Methodology

In this article, all the results and visualizations, including the diverse range of plots and graphical representations, have been generated using Python with

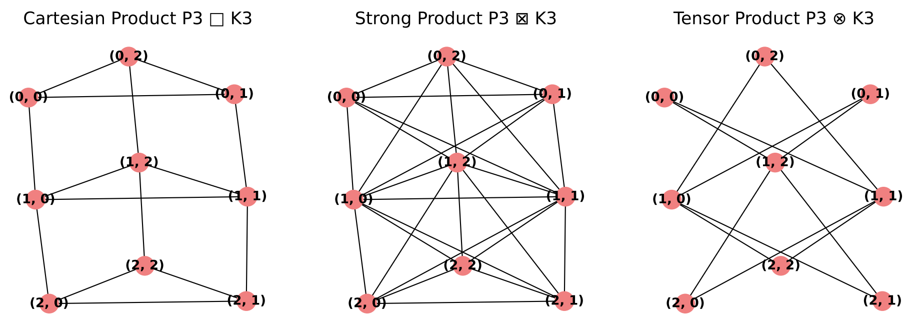

Figure 2: Graph Products of P_3 and K_3: Cartesian Product $P_3 \Box K_3$, Strong Product $P_3 \boxtimes K_3$, and Tensor Product $P_3 \otimes K_3$

NetworkX[1], Matplotlib[2], Scikit-Learn[3], Numpy[4] and Pandas[5]. The code is openly hosted on GitHub, under the repository **graph-gumba**[6]. The repository encompasses all the essential scripts, algorithms, and computational methods employed throughout our research. This ensures transparency, reproducibility, and ease of access for those interested in examining, verifying, or building upon our work.

2.1 Graph Selection

Our study involved the use of all connected and non-empty graphs denoted by $G = (V, E)$ where the number of vertices $|V|$ is less than or equal to 7. This comes to a total of 995 graphs. These graphs were sourced from the NetworkX graph atlas as referenced in Read et. al. [1]. The dataset we utilized covers a wide range of graph structures. However, it is important to note that the dataset is unstratified and exhibits a notable bias towards graphs with a greater number of nodes, as illustrated in figure 3.

2.2 Factor Graphs

In our study, we employed the concept of factor graphs to enhance the structural intricacy of a primary graph through the application of graph products. This was explored utilizing three distinct variants of factor graphs: the complete graph, the path graph and the star graph as depicted in figure 4. For these structural models, we selected every other count of nodes, starting from 3 nodes up to 15 nodes.

[1]`https://networkx.org/`
[2]`https://matplotlib.org/`
[3]`https://scikit-learn.org/`
[4]`https://numpy.org/`
[5]`https://pandas.pydata.org/`
[6]`https://github.com/max-seeli/graph-gumbo`

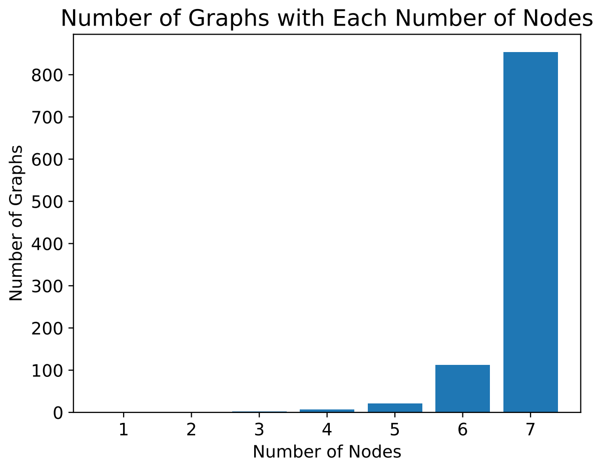

Figure 3: The distribution of graphs by their node count

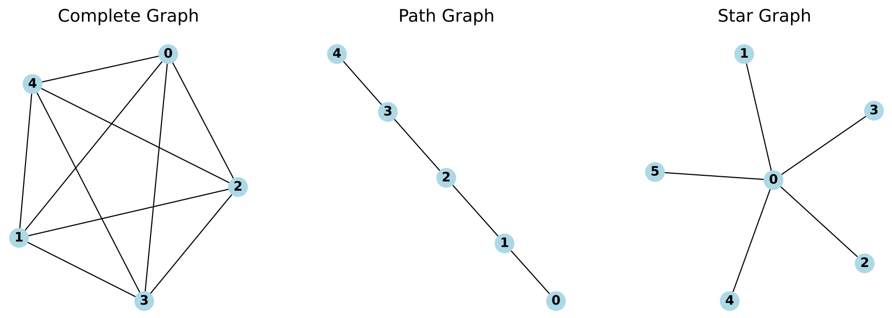

Figure 4: Used Factor Graphs: Complete Graph (K_5), Path Graph (P_5) and Star Graph (S_5)

2.3 Graph Products

To enhance the expressivity of our embeddings, we applied three types of graph products: Cartesian, Strong, and Tensor (as described in section 1). This was done for all combinations of graphs from our dataset with the factor graphs.

2.4 Graph Embedding through Cycle Counting

The core of our methodology involved embedding graphs based on cycle counting. This technique captures the cyclic structures within graphs, offering insights into their topological characteristics. For every graph $G = (V, E)$, all available cycles $\mathcal{C}(G) = \{C | C \text{ is a cycle in } G\}$ are aggregated and using a fixed vector space, the number of cycles for each length is evaluated. At every index of the vector space, the number of cycles with the corresponding length is stored, forming the embedding:

$$\text{Embed(G)} = \mathbf{v}$$

Where:

- $\textbf{Embed}(G)$ is the embedding function applied to G.
- $\mathbf{v} = [v_1, v_2, v_3, \ldots, v_n]$, with $v_i = |\{c \mid c \in \mathcal{C}(G) \wedge |c| = i\}|$

We can look at the example in figure 5. This graph would result in the following embedding vector:

$$\mathbf{v} = [0, 0, 2, 1, 1]$$

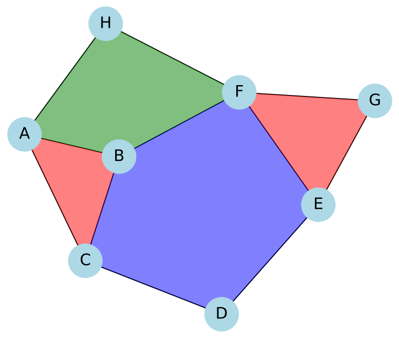

Figure 5: Example graph with cycles of different sizes

3 Experiment

3.1 Baseline

To assess the effectiveness of our proposed methodology, we established two comparative baselines.

The first baseline was developed through the application of the cycle counting method to embed our graph dataset. This process served as a preliminary benchmark, enabling us to gauge the potential advancements achievable through the implementation of graph products.

For the second baseline, we utilized the algorithm associated with the Weisfeiler-Lehman isomorphism test. This approach facilitated the generation of hash-embeddings for the graphs. Employing this well-established method allowed us to evaluate the efficacy of our proposed approach in relation to existing, recognized techniques in the field. (More on the Weisfeiler-Lehman embedding in Appendix A)

3.2 Application of Graph Products and Embeddings

After the preliminary baseline embeddings were established, we employed the Cartesian, Strong, and Tensor products. These were meticulously applied across every combination of the graph dataset and the factor graphs. Further, we have again embedded each of these combinations using the cycle counting method. This comprehensive and systematic application ensures an equal preprocessing, facilitating an in-depth and multifaceted analysis of the data under consideration.

3.3 Comparative Analysis

The final stage of our experiment involved a comparative analysis of the embeddings, pre and post the application of graph products. This was achieved by looking at every combination of graph product method and factor graph individually. Then, every possible unordered pair of graphs is checked for equality. Whenever two graph embeddings are identified as identical, this indicates that the chosen method (i.e. graph product and factor graph) cannot effectively distinguish between these two specific graphs. This comparison was crucial in determining the impact and effectiveness of graph products in graph embedding expressivity.

4 Results

In the following, we will benchmark the performance of the transformation by looking at the number of pairs of graphs from our dataset that have the same embedding even though they are non-isomorphic.

For our two baselines, we were able to find the following:

- For the original graphs with the cycle counting embedding, we found a total of **14239** pairs of graphs that were indiscernible.
- The traditional Weisfeiler-Lehman only has **20** pairs that result in the same embedding.

4.1 Evaluation of Absolute Graph Product Performance

When first calculating the product graph with a specific factor graph and only then embedding the result, we achieved the following results:

	Graph Products			
Factor	Cartesian	Strong	Tensor	
K_3	1096	11795	255	Complete
K_5	1040	9086	108	
K_7	1039	9086	96	
K_{13}	1039	9086	**95**	
P_3	1407	787	7311	Path
P_5	1404	33	549	
P_7	1404	11	121	
P_{13}	1404	**5**	49	
S_3	191	1987	7263	Star
S_5	**185**	2570	7224	
S_7	185	1390	7224	
S_{13}	185	1443	7224	

Table 1: Number of non-discernible pairs of graphs per method.

4.2 Embedding Dimensionality Reduction

To further visualize the effect, the graph product does have on the embedding space, we performed a dimensionality reduction using the *T-distributed Stochastic Neighbor Embedding (t-SNE)*. We applied this reduction on the embeddings of the original graphs (see figure 6) as well as on all the combinations of graph products with three selected factor graphs (see figure 7). We chose the K_7, the P_7 and the S_7 to each represent their class of graphs with a size that seems to achieve near maximal performance according to table 1.

5 Discussion

This study embarked on a novel exploration of graph embeddings, leveraging the combined utility of cycle counting and graph product operations. Our findings provide insightful revelations into the capabilities and limitations of this approach.

5.1 Interpretation of the Results

Examining Table 1 reveals a standout observation: the combination of the Strong product with the path graph P_13 emerges as exceptionally effective, resulting in only 5 instances where different graphs shared identical embeddings—a scenario we refer to as 'collisions.'

Moreover, our analysis identified certain combinations of graph products and structures of factor graphs that demonstrated superior performance in minimizing collisions. These notably effective combinations include:

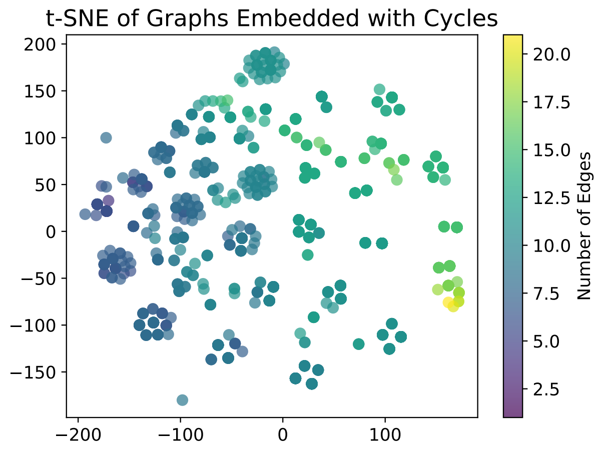

Figure 6: t-SNE dimensionality reduction of the embeddings of the original graphs

- Cartesian product paired with star graphs
- Strong product in conjunction with path graphs
- Tensor product combined with either complete graphs or path graphs

These combinations stood out, showing a remarkable reduction in collisions—about 90% to 99% less compared to other tested pairs. This significant decrease underscores their effectiveness in creating distinct embeddings for different graphs.

This trend is further corroborated by the results seen in the t-SNE dimensionality reduction plots (refer to figure 7). In these visual representations, the above-mentioned combinations display a more dispersed and uniform distribution in the embedding space. This is indicative of a more effective separation of graph characteristics, as opposed to other combinations where we observed a tendency for embeddings to cluster together, suggesting less distinctiveness.

In summary, these findings highlight the importance of selecting the appropriate combination of graph product and factor graph, as this choice greatly influences the distinctiveness and effectiveness of graph embeddings. For domain specific applications, this could also mean to adjust the chosen combination to the general structure of the primary graphs.

5.2 Comparison with Baseline

In the comparison with both of our established baselines, we can observe, that the first basesline, where the primary graphs were embedded using the cycle counting method was superseded by every single combination of graph product with factor graph. More interestingly, the traditional Weisfeiler-Lehman baseline with 20 collisions was surpassed by the best performing combination of the

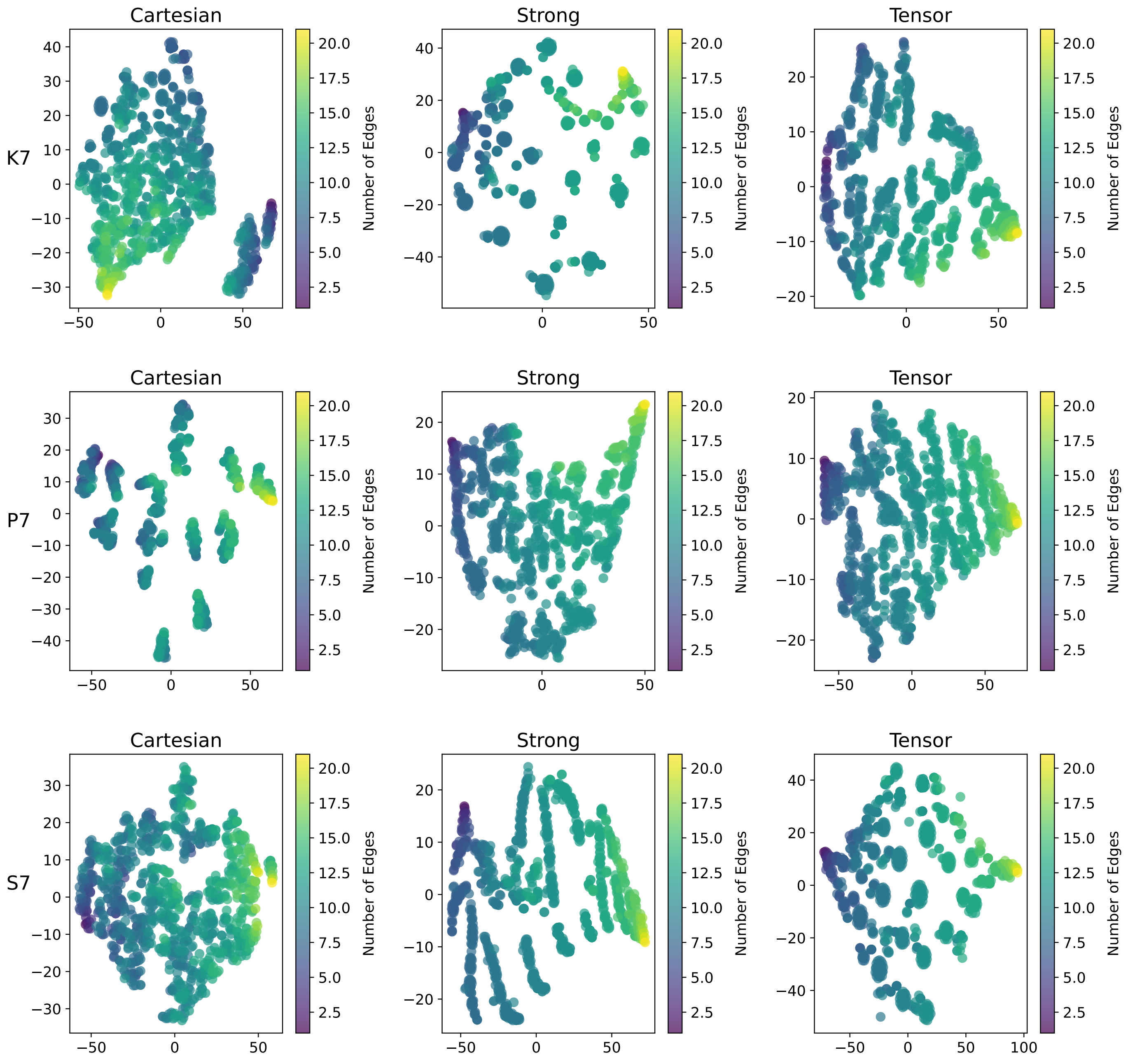

Figure 7: t-SNE dimensionality reduction of the embeddings of the transformed graphs using graph products

strong product with path graphs. Here we were able to achieve a minimum number of collisions counting only 5.

Our analysis reveals some compelling insights when comparing the outcomes of our novel approach with the established baselines. In the initial baseline, where we used the cycle counting method to embed the original graphs, it was evident that all combinations of graph products and factor graphs outperformed this approach. This result highlights the enhanced effectiveness of integrating graph products into the embedding process.

Even more notably, when compared to the traditional Weisfeiler-Lehman method, which resulted in 20 instances where different graphs were indistinguishable, our approach showed a marked improvement. The most successful combination, involving the strong product applied to path graphs, significantly reduced the number of collisions to just 5. This finding is particularly striking as it suggests that our method not only competes with but can surpass established techniques in certain configurations. It underscores the potential of using specific graph product and factor graph pairings to achieve more distinctive and expressive graph embeddings.

5.3 Convergence

Our analysis reveals a notable trend: as we increase the size of the factor graphs, the effectiveness of cycle counting embeddings in differentiating graph structures improves. This trend is evident when comparing the results of different path graphs with the strong product. For instance, the path graph P_3 combined with the strong product resulted in 787 indistinguishable graph pairs, whereas P_5 significantly reduced this number to 33, and P_7 further lowered it to just 11 collisions.

This pattern suggests that enhancing the complexity and size of the factor graphs positively impacts the distinctiveness of the graph products. However, it's important to note that this improvement in differentiation is primarily observed in the initial increases in factor graph size. Beyond a certain point—typically when the factor graph's size approaches or slightly exceeds the number of nodes in the original graph—the benefits plateau. In other words, increasing the factor graph size beyond this threshold yields minimal or no additional advantages in distinguishing between non-isomorphic graphs.

This observation points to a convergence in the effectiveness of graph product transformations for enhancing graph embedding expressivity, highlighting a limit to the benefits gained from merely increasing factor graph size.

6 Conclusion

This study introduced an innovative method to enhance graph embedding expressivity by integrating cycle counting with graph product operations, such as Cartesian, Strong, and Tensor products. Our research focused on applying these techniques to a variety of small graphs and analyzing their effectiveness.

Key findings from our study include the identification of specific combinations of graph products and factor graphs that significantly reduce the occurrence of collisions — cases where different graphs yield identical embeddings. Notably, the Strong product with path graphs, the Cartesian product with star graphs, and the Tensor product with complete or path graphs were highly effective, reducing collisions by about 90% to 99%.

Our analysis also revealed a convergence trend, where increasing the size of factor graphs initially improves the distinctiveness of graph embeddings, but this benefit plateaus beyond a certain size. This finding suggests an optimal range for factor graph size that maximizes the effectiveness of graph embeddings.

In summary, this research offers valuable insights into optimizing graph embedding techniques, highlighting the importance of choosing appropriate graph products and factor graphs. These insights can be particularly beneficial for domain-specific applications and pave the way for further exploration in the field of graph embeddings.

References

[1] R. Read and R. Wilson, *An Atlas of Graphs*, ser. Oxford science publications. Clarendon Press, 1998. [Online]. Available: https://books.google.at/books?id=QOPuAAAAMAAJ

Appendix A Weisfeiler-Lehman Embeddings

Initially, we contemplated the possibility of enhancing the Weisfeiler-Lehman embedding method using our graph product approach. However, our experiments yielded an unexpected outcome: the application of graph products had no noticeable effect on the performance of the Weisfeiler-Lehman method in terms of distinguishing between different graphs.

In our trials, each combination of factor graphs and graph products resulted in exactly 20 pairs of graphs that were indistinguishable through their embeddings. This number precisely matched the count of indiscernible pairs we found when applying the Weisfeiler-Lehman method to the original graphs, as mentioned in our baseline analysis (see section 3.1).

From these observations, we draw a clear conclusion: the application of graph products, as proposed in our study, does not influence the effectiveness of the Weisfeiler-Lehman embedding method. This finding suggests that the Weisfeiler-Lehman technique is inherently robust to the alterations made by graph products, maintaining its discriminative power regardless of such transformations.