TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

Artificial Intelligence's Impact on Data Structures

Plaaksha Chaudhry[1], Tanya Verma[2], Dr Gaurvi Shukla[3], Dr Shalini Lamba[4]

[1,2] Scholar, National Post Graduate College, Lucknow

[3,4] Assistant Professor, Department of Computer Science, National Post Graduate College, Lucknow

chaudhryplaaksha@gmail.com[1], tanyavr870@gmail.com[2], gaurvi16@gmail.com[3], drshalinilamba@gmail.com[4]

KEYWORDS

Machine Learning for Data Structures, Learned Index Structures, Hybrid System Design, Predictive Memory Management, Adaptive Algorithms

ABSTRACT

To counter the growing number of modern workload complexities, system designers are increasingly incorporating machine learning (ML) to optimize software performance. Traditional system components like data structures, caches, memory allocators, garbage collectors, and database optimizers employ pre-computed heuristics or analytical models that guarantee worst-case performance but neglect patterns observed in actual workloads. Data-driven methods can boost average case performance by predicting demands, critical resources, and access patterns based on historical execution traces and observations during runtime. These applications include learning-based auto-tuning databases for adapting to varying workloads, ML-assisted memory management for improving locality and reducing fragmentation issues, learning-based data structures with predictive caching for reducing latency and cache misses. While end-to-end systems based on ML learning are still being researched owing to limitations with inference time complexity, retraining expenses, and robustness to distributional shifts, a complementary learning approach utilizing small ML models along with algorithms is preferable to ensure feasible performance improvements. Future research topics on learning-enhanced system design will be discussed in the following sections.

1. Introduction

Artificial Intelligence, or AI for short, refers to a method of developing machines to perform tasks requiring human intelligence, including pattern recognition, decision-making, and problem-solving. The principal aim of AI technology is to perform these complex tasks more effectively and precisely than humans, or even better than humans can do naturally. Using software to program machines to adapt to their surroundings and change their behaviours accordingly makes AI possible to develop long-term objectives.

Data structures are very helpful for effectively arranging and controlling data, and AI has altered the way in which data structures operate. Conventional data structures such as arrays, linked lists, trees, and graphs can now be optimized by applying ML or DL algorithms for the purpose of enhancing efficiency, reliability, and malleability. This paper examines the impact of AI, which affects basic data structures, and thus, the capability of intelligent algorithms will make computing more efficient and intelligent.

2. Literature Review

During the last decade, researchers have explored how machine learning and AI can be used to enhance traditional data structures. Initial work focused on the area of learning indexing, where neural networks predict critical locations and model the underlying distribution of data. This approach has shown improved search efficiency when set up against traditional indexing methods such as trees of b-variety, especially under workloads with predictable patterns. Later work extends these initial ideas to include

Corresponding Author: Plaaksha Chaudhry, Scholar, National Post Graduate College, Lucknow, India
Email: chaudhryplaaksha@gmail.com

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

adaptable and updatable learning indexes, which can accommodate evolving data and provide flexible performance in changing conditions. Sorting and searching algorithms have also seen their share of AI contributions. Methods like reinforcement learning optimize sorting networks for small or frequently queried datasets and obtain faster executions and reduced numbers of computational steps. In similar fashion, machine learning-based hashing and caching techniques demonstrate a reduction in collision rates and cache misses, especially in systems that have to deal with enormous datasets. These methods illustrate how AI can contribute to gains in real-world computing applications in terms of system responsiveness and temporal complexity.

AI-based approaches like smart garbage collection and prediction tools have additionally influenced memory management. These approaches can lower fragmentation, improve cache locality, and make adjustments to resource allocation based on workload patterns by studying real-time memory patterns. However, computationally costly overhead, retraining costs, and uncertainties during unexpected data patterns still exist as challenges. To strike an optimal balance between performance, trustworthiness, and predictability, various studies propose the use of AI-predictive/heuristic combined approaches with standard algorithms on these realizations to grasp the role of AI-based methodological improvements for enhancing the performance and design of modern data structures.

2.1. Automation and Optimization of Data Structures

Data structure optimisation involves selecting or designing the most appropriate structures for a specific task while maintaining a balance between execution speed and memory consumption. Traditionally, this optimisation process has relied on human expertise and manual adjustments. Although effective to some extent, such methods are often time-consuming and prone to errors.

With the advancement of modern computational methods, this process can now be automated. For example, reinforcement learning techniques can determine the most suitable data structure for a given workload at runtime by analysing previous access patterns. In practice, such a system can predict whether a hash table or a binary search tree will deliver better performance in a query-intensive environment. In addition, genetic algorithms can explore different configurations of data structures and refine them iteratively to reduce computational costs. By automating these decisions, the dependency on manual intervention is reduced, and data structures are consistently optimized for their intended applications, resulting in more efficient system performance.

2.2. Reducing time complexity in searching and sorting.

Searching and sorting form the backbone of operations in data structures, and their performance is generally judged by time complexity—for example, O(n) in the case of linear search or O(log n) for binary search. While these classical approaches are effective, they face limitations when applied to large or complex datasets. To address this, advanced computational methods introduce intelligent techniques that improve both execution speed and scalability.

One prominent solution involves the use of neural networks to predict search patterns. In heavily queried databases, for instance, a trained model can prioritise or rank specific records, effectively reducing the scope of the search. This targeted approach is particularly advantageous in large-scale systems where traditional searches become inefficient or impractical.

Sorting operations can also benefit from intelligent strategies. By estimating the degree of disorder within a dataset, adaptive methods can be applied to select the most suitable sorting technique. A hybrid system, for example, might dynamically alternate between comparison-based algorithms such as quicksort and distribution-based methods, depending on the characteristics of the input data. This

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

flexibility allows sorting to be carried out with greater efficiency than conventional fixed algorithms, particularly in diverse real-world applications.

2.3. Improving memory management through predictive algorithms.

The efficient use of memory is a fundamental aspect of data structure performance, particularly in environments with strict resource constraints such as embedded systems or large-scale distributed platforms. To meet these demands, advanced computational methods make use of predictive techniques that anticipate how data will be accessed and manage storage in a way that minimizes waste while maximizing efficiency.

In cache management, for instance, models such as recurrent neural networks (RNNs) can predict which data items are most likely to be requested in the near future. This allows the system to pre-fetch information and significantly reduce cache misses. Similarly, in dynamic memory allocation, analysing patterns of memory usage enables smarter allocation and release of resources, which helps to reduce fragmentation. A practical application of this can be seen in garbage collection within programming languages like Java and Python, where predictive models improve the identification of unused objects and release memory seamlessly, without disrupting execution.

Together, these innovations make data structures more compact, reduce unnecessary memory overhead, and enhance the overall performance of computing systems.

2.4. Adapting dynamic structures for real-time applications.

Applications that operate in real time—such as self-driving cars, online gaming environments, and high-frequency trading systems—require data structures capable of handling rapidly changing conditions and continuous data flow. Traditional static structures are often inadequate in these scenarios, whereas adaptive, self-adjusting structures provide the flexibility needed to maintain performance under pressure.

Take online recommendation platforms as an example: the underlying structures, whether graphs or tries, can be reorganized in response to user activity to deliver instant, low-latency suggestions. Likewise, online learning methods allow data structures to evolve as new information is introduced, automatically adjusting their organization to sustain efficiency. In graph-based contexts like social network analysis, frequently accessed nodes can be prioritized dynamically, reducing traversal times and improving responsiveness.

These adaptive qualities make intelligent data structures particularly well-suited to situations where both speed and flexibility are essential for success.

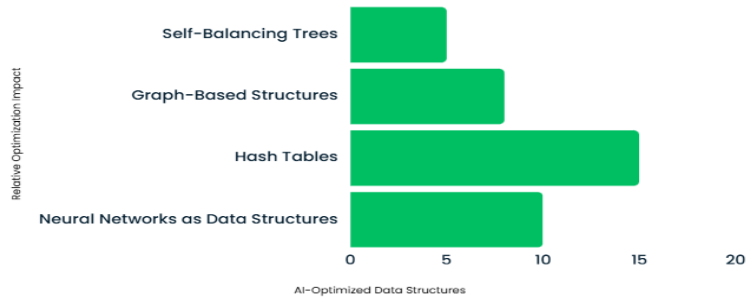

Figure 1: AI-based optimisation techniques enhance the adaptability and efficiency of traditional data structures by introducing learning-driven decision mechanisms

3. AI-Optimised Data Structures

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

3.1. Self-Balancing Trees

In order to maintain height balance and ensure logarithmic time complexity for insertion, deletion, and search operations, traditional self-balancing trees like AVL trees, Red-Black trees, and B-trees rely on predefined rotation and rebalancing procedures. Despite their effectiveness, these methods cannot always produce the best results when dealing with extremely dynamic or unpredictable access patterns. These trees are able to adaptively choose rebalancing solutions based on observed runtime behaviour thanks to AI approaches, especially reinforcement learning. AI-enhanced self-balancing trees can lower restructure overhead and boost overall operational efficiency in real-time applications by learning appropriate rotation rules and avoiding imbalance conditions.

3.2. Graph-Based Structures

Graph data structures are frequently used to model complicated relationships in applications such as network analysis, autonomous machinery, and navigation systems. Traditional shortest-path algorithms, such as Dijkstra's and A*, seek optimal routes using set cost functions and deterministic heuristics. Although these techniques function well in static environments, their efficiency can decline when system circumstances change regularly. To address this limitation, artificial intelligence technologies are increasingly integrated into graph-based algorithms. By learning from previous traversal data and historical feedback, machine learning models enable dynamic adjustment of edge weights and algorithmic values. This adaptability provides more efficient and accurate path selection in unreliable and dynamic situations. Such learning-driven optimisation is particularly helpful for autonomous systems, where real-time flexibility and constant modification are essential.

3.3. Hash Tables

Hash tables are extensively employed in contemporary computing systems due to their ability to support efficient data storage and retrieval with average constant-time complexity. However, their overall performance is highly dependent on collision frequency, which is influenced by the choice of hash functions and the effectiveness of collision resolution strategies. In large-scale and data-intensive environments, traditional static approaches often struggle to adapt to evolving access patterns, leading to increased collision overhead and reduced efficiency. Artificial intelligence–based optimisation techniques address these limitations by enabling dynamic analysis of data distribution and access behaviour. By learning from historical usage patterns, machine learning models can adjust hash functions and collision handling mechanisms in real time. This predictive and adaptive capability reduces clustering, improves memory utilisation, and enhances system scalability and performance.

3.4. Neural Networks as Data Structures

Deep learning models such as convolutional and recurrent neural networks are primarily designed for pattern recognition, capturing spatial and temporal relationships in large-scale data. While they process structured inputs through layered transformations, they are not traditionally considered data structures, which are engineered for efficient data storage and retrieval with defined performance guarantees. A related development is learned index structures, where neural networks approximate traditional indexing mechanisms by modelling data distributions. Despite that, such approaches typically rely on simple feedforward models rather than CNNs or RNNs, which remain focused on feature extraction rather than explicit data organisation.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

Data Structure	**Limitation in Conventional Approach**	**AI-Based Optimisation Method**	**Key Advantage**
Self-Balancing Trees	Performance overhead due to rigid rebalancing rules	Reinforcement learning–guided balancing strategies	Improved efficiency through adaptive tree balancing
Graph-Based Structures	Dependence on static heuristics and fixed cost functions	Machine learning–driven heuristic optimisation	Enhanced accuracy in shortest-path computation
Hash Tables	Performance degradation caused by frequent collisions	Predictive collision resolution using learning models	Faster and more reliable data access
Neural Networks as Data Structures	Limited flexibility in traditional data representations	Deep learning architectures such as CNNs and RNNs	Scalable and adaptive information representation

4. AI's Impact on Search and Sorting Algorithms

4.1. AI-driven optimisations in binary search and hash-based search techniques.

Binary search provides $O(\log n)$time complexity for lookup operations on sorted arrays, while hash tables achieve average-case $O(1)$performance when collision resolution is effective. Both techniques are traditionally designed under assumptions of uniform data distributions or worst-case analysis. Learned index structures replace conventional indexing mechanisms with machine learning models that approximate the cumulative distribution function of sorted keys. These models predict an approximate key location, which is subsequently refined using a localised search procedure, thereby reducing memory accesses for non-uniform data distribution. Similarly, learning-based hashing techniques employ data-dependent hash functions trained on observed key distributions, resulting in reduced collision rates and improved lookup efficiency for predictable workloads.

4.2. Evolution of sorting algorithms, such as AI-enhanced quicksort and merge sort for large datasets.

Quicksort and merge sort both achieve an average-case time complexity of $O(n\log n)$ however, quicksort is frequently favoured in practical implementations due to its superior cache behaviour and smaller constant factors. Recent studies have examined the use of machine learning techniques to optimise specific components of these algorithms rather than to replace them entirely. For example, reinforcement learning has been applied to identify efficient sorting networks for small, fixed-size inputs, as illustrated by the AlphaDev approach, which reports reductions in instruction count and execution time. These optimised networks can be employed in the base cases of recursive sorting algorithms, leading to performance improvements for certain data distributions. In contrast, learned sorting methods based on position prediction using data distribution models have demonstrated limited practical benefit due to prediction inaccuracies. As a result, classical sorting algorithms remain the dominant choice for general-purpose sorting as of early 2026.

4.3. Predictive analytics in sorting for efficient real-time data processing.

Real-time data processing often involves streaming or partially ordered inputs, where early sorting decisions can influence overall performance. Predictive models have been explored to assist such decisions by identifying patterns in incoming data. For example, learned models may guide pivot selection in quicksort by leveraging statistical properties of data streams or preliminary partitioning results. Neural networks trained on historical stream data can also estimate approximate ordering in dynamically changing environments, producing partially sorted states that require minimal refinement.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

These techniques have been applied in domains such as sensor data processing, real-time fraud detection, and recommendation systems that require efficient ranking.

Despite these potential benefits, the computational overhead associated with model inference may offset performance gains in highly variable or unpredictable streams. Empirical studies indicate that learning-assisted approaches are most effective when data exhibits stable or repeatable patterns, complementing traditional external sorting methods. Consequently, classical algorithms remain preferable in terms of general applicability, worst-case performance guarantees, and implementation simplicity. Hybrid systems that combine conventional algorithms with selective learning-based optimisations represent a promising direction for future research.

5. AI in Dynamic Memory Management

5.1. Machine learning models predicting memory allocation needs.

Traditional memory allocation strategies often experience fragmentation and suboptimal utilisation under dynamic workloads, as they rely on fixed heuristic policies such as first-fit and best-fit that cannot adapt to changing allocation behaviour. To address these limitations, recent research has explored the application of machine learning techniques that leverage runtime observations and historical allocation data to inform memory management decisions.

Models trained on allocation traces or calling-context information can estimate object lifetimes and access characteristics, enabling the construction of specialised heap organisations based on predicted lifetime classes. Such approaches have been shown to reduce fragmentation and improve cache locality, particularly in large-page memory configurations. In addition, predictive modelling of data access frequency has become an important aspect of memory system design in hybrid memory architectures combining DRAM and non-volatile memory, where informed data placement can significantly reduce access latency.

Machine learning methods, including regression and reinforcement learning, have also been applied to predict task-specific memory demand in cloud and high-performance computing workloads, helping to limit over-provisioning and improve resource efficiency. While these techniques demonstrate clear benefits for workload-specific scenarios, effective management of highly irregular and unpredictable allocation patterns remains an open research challenge as of early 2026.

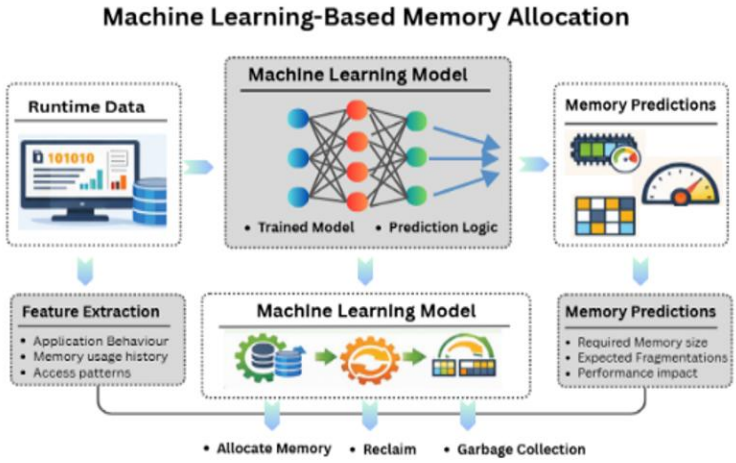

Figure 2: Machine learning–based prediction of memory usage guides proactive and efficient memory allocation in modern systems.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

5.2. Automated garbage collection in modern programming languages using AI.

In managed programming languages such as Python and Java, garbage collection (GC) is responsible for reclaiming memory occupied by unreachable objects. Beyond memory reclamation, GC design involves critical decisions regarding collection timing and strategy, which directly influence application throughput and pause times. Recent research has explored *learned garbage collection* techniques that employ machine learning, particularly reinforcement learning, to adaptively determine collection initiation while optimising user-defined objectives such as latency or throughput.

Early approaches based on methods such as tabular Q-learning demonstrate the feasibility of adapting GC behaviour to application-specific allocation patterns, enabling reduced collection frequency and more informed parameter tuning, including heap sizing and trigger thresholds. Machine learning models have also been investigated for predicting collection events based on runtime observations and historical execution data.

Despite these promising results, fully learning-driven garbage collectors remain largely experimental. As of early 2026, production-grade collectors—such as G1 and ZGC in the Java Virtual Machine—continue to rely primarily on carefully engineered heuristics rather than learned control policies, reflecting the challenges of robustness, predictability, and deployment at scale.

5.3. Efficient memory handling in AI applications such as cloud computing and big data processing.

AI-intensive workloads, including model training and inference, impose substantial memory demands due to large input datasets, parameter matrices, and intermediate activation states. In cloud-based environments, machine learning techniques have been applied to predict memory usage patterns and enable dynamic memory resizing for virtual machines and containers, thereby supporting more efficient resource allocation.

In GPU-accelerated deep learning systems, predictive models are increasingly used to manage limited device memory through mechanisms such as unified memory management and controlled swapping, helping to balance performance and capacity constraints. Similarly, in-memory computing platforms employ learning-based scheduling strategies to optimise data placement across tiered memory hierarchies, such as DRAM and persistent memory, within large-scale data processing frameworks.

In heterogeneous systems, hybrid memory management approaches combine runtime profiling with machine learning to guide memory allocation across different memory technologies. These techniques improve scalability for large language models and data analytics pipelines by adapting to workload-specific characteristics. Overall, workload-aware memory management driven by machine learning has demonstrated improved utilisation and cost efficiency, particularly in distributed systems and AI-centric computing environments. Traditional methods continue to provide strong baselines for general use, with hybrid systems representing the most practical advancements as of January 2026.

6. Real-World Applications

6.1. AI-powered databases optimising query processing (e.g., Google's BigTable, Amazon DynamoDB).

Contemporary database systems are increasingly exploring the use of machine learning to improve both resource efficiency and query execution performance. A prominent example is the development of learned index structures, which model data distributions to estimate the likely location of keys. In certain

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

workloads, these approaches can complement or partially replace traditional index structures, such as B-trees, resulting in faster access for sorted data.

Machine learning has also been adopted within query optimisers to support more informed decision-making when selecting execution plans, particularly for complex join operations or frequently executed queries. By improving estimates of cardinality and workload behaviour, these techniques help optimise performance in large-scale and distributed database systems.

By contrast, highly scalable distributed databases such as Google Bigtable and Amazon DynamoDB continue to rely primarily on well-established distributed design principles and deterministic indexing mechanisms to achieve reliability and performance. In these systems, machine learning is typically applied at higher layers for analytics, monitoring, and predictive management rather than for core storage and indexing functions.

6.2. AI-driven caching mechanisms improving data retrieval speeds.

Latency reduction for frequently accessed content is largely achieved through caching mechanisms. Traditional policies, such as Least Recently Used (LRU), can be augmented with machine learning models that predict future access patterns, enabling more informed decisions regarding prefetching and eviction. Unlike adaptive methods that continuously adjust to real-time workload variations, these models rely on historical query data to estimate demand. Such approaches are particularly effective in remote caches and multi-tier storage architectures, where they reduce cache misses and improve retrieval performance in cloud-based applications. While the overhead of model inference is typically low, the benefits are most pronounced in environments with predictable or moderately variable access patterns, such as streaming services and model-serving platforms.

6.3. AI's role in cybersecurity, fraud detection, and real-time decision-making systems.

Machine learning has demonstrated significant potential in cybersecurity and fraud detection due to its ability to process large volumes of data rapidly and identify anomalies or threats in real time. Techniques such as supervised classifiers, ensemble methods, and unsupervised detection algorithms offer advantages over traditional rule-based systems by analysing transaction streams or network activity to detect suspicious behaviour.

Recent advancements, including transformer-based architectures and graph neural networks, are particularly effective at identifying subtle temporal patterns and coordinated attacks. Explainable AI approaches further enhance adoption by presenting alerts in an interpretable manner, supporting trust and compliance.

The application of these methods has been shown to reduce false positives and improve response times across domains such as financial services, e-commerce, and critical infrastructure. Nevertheless, challenges remain in ensuring robustness against adversarial attacks and maintaining performance when processing high-velocity data streams.

7. Challenges and Future Prospects

7.1. Computational complexity of AI-optimised data structures.

AI-driven data structures introduce additional costs that do not arise in traditional designs, mainly due to model inference and the need for occasional retraining. Classical data structures offer predictable performance with well-defined asymptotic guarantees. In contrast, learning-based approaches rely on executing a model—such as a neural network—to predict key locations or adjust hashing behaviour. Although these predictions are typically computed in constant time, their execution involves higher overhead than simple arithmetic or pointer operations.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

As data distributions evolve, maintaining model accuracy may require periodic updates or retraining, which can increase amortised costs compared to conventional methods. Memory usage is another important consideration, as storing model parameters alongside application data can significantly increase storage requirements, particularly in environments with limited resources.

For these reasons, AI-optimised data structures are most effective in settings with relatively stable data distributions or workloads dominated by read operations, where the benefits of accurate prediction outweigh the costs of inference and maintenance. Determining how best to balance performance gains with computational and memory overheads remains an active area of research.

7.2. Trade-offs between accuracy and efficiency.

Learned models can be sensitive to distribution shifts, which may lead to degraded performance under unfavourable or adversarial input conditions. Although such models are effective at capturing regularities in specific data distributions, their performance may decline when deployed outside the conditions observed during training. For instance, a learned index optimised for uniformly distributed keys may perform worse than a traditional B-tree when exposed to highly skewed workloads.

Many learning-based techniques trade strict correctness guarantees for improved average-case performance by providing probabilistic or approximate results. While this trade-off can be acceptable in performance-oriented applications, it is often unsuitable for safety-critical systems that require deterministic behaviour and strong worst-case guarantees. In addition, the practical complexity associated with model training, tuning, and deployment can exceed that of classical data structures, which benefit from well-understood behaviour and long-standing reliability.

To address these limitations, recent research has focused on hybrid approaches that combine learning-based components with conventional algorithms. Such designs aim to preserve formal guarantees while leveraging learned predictions where they are accurate, and reverting to traditional methods when model confidence or performance degrades.

Aspect	**Higher Accuracy**	**Higher Efficiency**
Model Complexity	Uses deeper or more expressive models that capture fine-grained patterns	Relies on simpler or compressed models with fewer parameters
Computation Cost	Requires more computation during inference and training	Minimizes computation to reduce latency and energy use
Memory Overhead	Needs additional memory to store model parameters and intermediate states	Keeps memory usage low, suitable for resource-constrained systems
Prediction Quality	Produces more precise predictions, improving average-case performance	Accepts approximate predictions that are “good enough” in practice
Response Time	May introduce small delays due to model execution	Enables faster responses and real-time decision making
Robustness to Change	Often sensitive to workload shifts and may require retraining	More stable under changing conditions with less maintenance
Typical Use Cases	Offline optimization, analytics, or stable workloads	Online systems, latency-critical paths, and embedded environments

7.3. Future AI-driven innovations in data organisation and management.

Hybrid approaches that combine lightweight learning components with classical data structures are increasingly seen as a practical path forward. Recent progress in model compression, quantisation, and hardware support for inference has reduced the cost of integrating machine learning into low-level system components, making such designs more feasible than in the past.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

At the system level, research has explored self-tuning databases that adapt indexing, caching, and query optimisationstrategies based on observed workloads. Large pretrained models have also shown promise in assisting data management tasks by capturing general data patterns across different environments. In addition, techniques based on reinforcement learning and neural algorithmic reasoning may enable the automated discovery of algorithms tailored to specific problem classes. Federated and privacy-preserving learning methods further support optimisation across distributed systems without requiring direct data sharing.

Overall, both research and industry are moving toward solutions that improve performance and resource efficiency while reducing configuration and operational complexity. Although challenges remain in terms of robustness, generality, and predictable behaviour, the continued integration of machine learning with traditional systems design is likely to play an important role in the future of data organisation and management.

8. Conclusion

This paper examined how artificial intelligence is influencing the design and behaviour of modern data structures. Learning-based techniques can improve adaptability and performance in tasks such as searching, sorting, and memory management, particularly when workloads exhibit stable patterns. However, traditional data structures continue to be indispensable because of their predictable behaviour, low overhead, and proven reliability. Overall, hybrid approaches that combine classical techniques with selective use of machine learning appear to offer the most practical and effective direction for future data organisation and system designing.

References

[1]. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[2]. S.Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2021.
[3]. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf
[4]. Y. LeCun, Y. Bengio, and G. Hinton,“Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015.
[5]. T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, “The case for learned index structures,” in Proc. ACM SIGMOD Int. Conf. on Management of Data, 2018, pp. 489–504.
[6]. J. Ding et al., “ALEX: An updatable adaptive learned index,” in Proc. ACM SIGMOD Int. Conf. on Management of Data, 2020, pp. 969–984.
A. Kipf et al., “RadixSpline: A single-pass learned index,” Proc. VLDB Endowment, vol. 13, no. 12, pp. 2777–2790, 2020.
[7]. Mitzenmacher, “A model for learned Bloom filters and optimizing by sandwiching,” in Advances in Neural Information Processing Systems (NeurIPS), 2018. https://papers.nips.cc/paper/2018/hash/0ec21f0ae55b2f5c33b9e4f4d3b5c6b0-Abstract.html
[8]. Mankowitz et al., “Discovering faster sorting algorithms using reinforcement learning,” Nature, vol. 618, pp. 257–263, 2023.
[9]. G. Graefe, “Implementing sorting in database systems,” ACM Computing Surveys, vol. 38, no. 3, 2006.
[10]. Y. Mao, J. Oak, and M. Stonebraker, “Learning-based sorting,” arXiv preprint, 2020.
A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in Proc. Int. Conf. on Very Large Data Bases (VLDB), 1999, pp. 518–529.

TEJAS Journal of Technologies and Humanitarian Science
ISSN-2583-5599
Vol.04, I.04 (2025)
https://www.tejasjournals.com/
https://doi.org/10.63920/tjths.44010

[11]. A. Wang, Z. Ding, and Y. Zhang, "Learning-based hashing for efficient similarity search," IEEE Trans. Knowledge and Data Engineering, vol. 33, no. 7, pp. 2945–2958, Jul. 2021.
[12]. E. D. Berger, B. G. Zorn, and K. S. McKinley, "Reconsidering custom memory allocation," in Proc. ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2002.
[13]. X. Yang, J. Chen, and Y. Bao, "Predicting object lifetimes to improve memory management," in Proc. IEEE/ACM Int. Symp. on Microarchitecture (MICRO), 2017. https://doi.org/10.1145/582419.582421
[14]. K. Ma, X. Li, and S. Wang, "Machine learning–based adaptive garbage collection," in Proc. IEEE Int. Conf. on Software Maintenance and Evolution (ICSME), 2019, pp. 550–560.
[15]. J. Ren, S. Nguyen, and D. Gregg, "Learning-based memory management for managed runtimes," IEEE Trans. Computers, vol. 69, no. 11, pp. 1656–1669, Nov. 2020. https://doi.org/10.1109/TC.2020.3008134
[16]. A. Marcus et al., "Neo: A learned query optimizer," Proc. VLDB Endowment, vol. 12, no. 11, pp. 1705–1718, 2019. https://www.vldb.org/pvldb/vol12/p1705-marcus.pdf
[17]. F. Chang et al., "Bigtable: A distributed storage system for structured data," ACM Trans. Computer Systems, vol. 26, no. 2, 2008. https://doi.org/10.1145/1365815.1365816
[18]. M. Zaharia et al. "Learning-based cache admission policies," in Proc. ACM Symp. on Cloud Computing (SoCC), 2017.
[19]. J. Hashemi et al., "Learning cache replacement with METADATA," in Proc. ACM SIGMETRICS, 2018.
[20]. P. García-Teodoro et al., "Anomaly-based network intrusion detection," Computers & Security, vol. 28, no. 1–2, pp. 18–28, 2009. https://doi.org/10.1016/j.cose.2008.08.003
[21]. H. Sarker, "CyberLearning: Effectiveness analysis of machine learning security modeling," arXiv preprint, 2021.
[22]. L. Marino et al., "Self-supervised and interpretable anomaly detection using transformers," arXiv preprint, 2022.
[23]. Y. Liu et al., "Financial fraud detection using graph neural networks," Expert Systems with Applications, 2024. https://doi.org/10.1016/j.eswa.2023.121014
[24]. Chen et al., "Self-driving database management systems," in Proc. Conf. on Innovative Data Systems Research (CIDR), 2017. http://cidrdb.org/cidr2017/papers/p15-chen-cidr17.pdf
[25]. Konečný et al., "Federated learning: Strategies for improving communication efficiency," in NeurIPS Workshop, 2016.