International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 6 June - 2025, Pages: 48-60

The Future of Data Sorting: Integrating AI for Enhanced Efficiency and Accuracy

Afnan A. Mezied and Samy S. Abu-Naser

Department of Information Technology,
Science, Faculty of Engineering and Information Technology,
Al-Azhar University, Gaza, Palestine

Abstract: *Sorting algorithms have always been the primary focus of data organization and have been the same since they were discovered. They play a vital role in reducing work and maximizing efficiency as well as accuracy. This paper aims at comparing and examining traditional sorting algorithms, usually applied by programmers as opposed to AI-based methods like Decision Trees and Neural Networks. The performance of these sorting techniques would be evaluated on time over such data executions considering memory requirement and accuracy on different datasets. Most traditional methods could sort structured data even of a smaller dataset, but they cannot work well when it comes to scaling and unstructured data [4]. AI-based algorithms, however, seem to perform better in terms of accuracy, which is superior in its implementation to complex and unstructured datasets, at a great cost of and accuracy; thus, making possible insights into the practical implementations. These findings show the potentials of hybrid methods regarding modern data sorting application challenges and pave the way for further future optimization and real-world implementation [8].*

Keywords: Artificial Intelligence, Data Sorting, Machine Learning, Neural Networks, Sorting Algorithms, Big Data, Data Processing, Algorithm Efficiency.

Introduction

Sorting algorithms are regarded as the heart and soul for effective management of data. Historically such algorithms have been mainly traditional types like those described by QuickSort and MergeSort, which have been admirable for small or structured datasets but have not been quite successful in dealing with relatively larger or unstructured data [1][4]. As data grow larger and richer in structure, Artificial Intelligence-based approaches, as seen in Decision Trees and Neural Networks, would surely bring in the promising benefits of higher accuracy and adaptability through learning from data patterns [5]. This paper lies comparison between traditional sorting algorithms and their AI based counterparts in the execution time, memory used and the accuracy. The study also examines hybrid types which capitalize on the speed of traditional algorithms for intelligent solutions saturating modern data-related problems. It is hoped such compilation will help guide further studies and application developments for big data, health, and e-commerce.

Table 1

Comparison of Traditional Sorting vs. AI-Based Sorting

Aspect	Traditional Sorting	AI-Based Sorting
Algorithms	QuickSort, MergeSort, HeapSort	Machine Learning, Neural Networks
Approach	Step-by-step comparisons	Learns from data patterns
Efficiency	Best for small to medium datasets	Best for large or complex datasets
Scalability	Struggles with large or complex data	Adapts to various data sizes and complexities
Data Handling	Best for structured data	Handles both structured and unstructured data
Optimization	Fixed algorithms	Learns and adapts over time
Use Cases	Small to medium datasets, high-speed tasks	Big data analysis, real-time unstructured data

Limitations	Struggles with large data and new patterns	High training costs, memory-heavy, slow for real-time

Objectives

- AI will contribute to better improvement in efficiency and accuracy through sorting data.
- Comparison of determination and performance between AI sorting algorithms and traditional methods.
- Create a solution that is purely hybrid to use artificial intelligence as well as typical nonbiological algorithms to yield the best results.
- Empirical verification of the benefits of sorting by AI [2, 5, 7].

Problem Statement

With big data and heavy datasets, competition becomes hard from the traditional methods to scale up sources of data with efficiency. They work in an entirely comparison-based way and involve multiple passes over a data stream; hence their low efficiency for large datasets makes them unsuitable. AI sort learns the patterns in the data and sorts them intelligently. The problem is that AI uses a lot of resources for computation and requires training, a practice introducing many difficulties. This paper will investigate these and give possible solutions as well as examine the viability of AI to optimize sorting [2, 3, 5].

Literature Review

Traditional Sorting Algorithms

Traditional sorting algorithms like QuickSort and MergeSort have been the mainstay where data has been organized for decades now. These algorithms are popular as they are best suited for sortation of small to medium-sized datasets with the suitable time complexities rendering them an efficient alternative for structured data [1]. However, they lack scalability for bigger datasets or unstructured data. For example, in the case of poor pivot selection, QuickSort degrades performance significantly to the worst-case complexity requirement [4]. Also, MergeSort becomes a bottleneck in the systems having limited resources as it requires extra memory for auxiliary arrays along with the present arrays [1][6].

AI-Based Sorting Algorithms

Artificial intelligence-based sorting methods, such as Decision Trees and Neural Networks, have surfaced as strong contenders over traditional approaches to dealing with complex and unstructured data [9][12]. Through pattern learning within the data, these algorithms have achieved very high accuracies, mostly above 97 percent with experiments using large datasets [5]. Processing of noise and unstructured datasets is excellent using Neural Networks. However, these need a high computational resource and training time [12]. Decision Trees bring together computational efficiency and accuracy but may exhibit some difficulty in the presence of higher noise level in the data [9].

Table 2

Comparison of Sorting Algorithms (Execution Time, Memory Usage, Accuracy)

Algorithm	Time Complexity	Execution Time (ms)	Memory Usage (MB)	Accuracy (%)
MergeSort	O(n log n)	255	105	96
QuickSort	O(n log n)	270	115	95
Neural Networks	O(n^2)	215	120	97
Decision Trees	O(n log n)	205	125	98

As shown in Table 2, with respect to the traditional approaches such as MergeSort and QuickSort, it showed lesser space and time consumption compared to the AI methods. Though Neural Networks and Decision Trees proved better accuracy for the above methods in an unstructured dataset. This reflects the trade-offs between traditional and AI-based algorithms.

Hybrid Methods

North-East-South Search These hybrid sorting methods have a complementary approach: exclusive use of ordinary and AI-based algorithms. For example, hybrid sorting takes use of preexisting algorithms for example quick sort on structured subsection data, while applying the AI methods on unstructured part [6][8]. An example was hybridized by Rigutini et al. [5], which demonstrated that sorting mixed datasets could tend to improve both in speed and accuracy.

Challenges and Opportunities

AI sorting may nevertheless encounter some disadvantages, including overfitting and high memory consumption [10]. Ensemble methods and cross-validation techniques are their expected solutions to provide a robust model for diverse datasets [7]. While preparing research grounds for optimizing such models for real-time applications and scaling up to big data scenarios, one could expect what future research would hold [8].

Methodology

The performance of traditional, AI-based, and hybrid sorting algorithms is evaluated through experiments on both synthetic and real-world datasets using the proposed methodology.

The execution time taken, memory consumption, and accuracy of algorithms are studied on data sets. Hybrid methods are again tested to know whether these have any potential to improve the sorting efficiency and scalability.

Dataset Preparation

Two kinds of data are prepared for the conducted experiments:

• **Synthetic Data:**
  - **Random Datasets:** Are used for testing and generally sorting in efficiency.
  - **Sorted Lists:** Are data sets to apply for those with minimal changes necessary [1].
  - **Reverse Sorted Lists:** Datasets in descending order as they will test adaptability under worst-case scenarios [1][4].
  - **Unstructured Data:** Things contain noise and nulls, challenging the robustness of AI models [9][12].

• **Real-World Data:**

Transactional logs, user ratings, and other structured and semi-structured records that will serve to test performance in real-world application scenarios [6].

Algorithm Implementation

1. Traditional Algorithms:
- **QuickSort**: using standard libraries and check pivot-based performance [1][4].
- **MergeSort**: configured in recursive approach to optimize on preprocessing structured data sets [1].

2. AI-Based Algorithms:
- **Neural Networks:** designed with Python's TensorFlow and scikit-learn libraries for pattern-based sorting [9][12].
- **Decision Trees:** for efficient classification-based sorting [9].

3. Hybrid Methods:
- QuickSort for preliminary structuring followed by a Neural Network for pattern recognition and final sorting [5][6].

Performance Metrics

The evaluation of each algorithm is attempted on the under-mentioned metrics

- **Execution Time:** Measured by how fast the algorithm is using the time module of python in milliseconds [5].
- **Memory Usage:** Profiled with the memory_profiler library to analyze resource consumption. [6].
- **By Accuracy:** Number of elements that have been correctly sorted versus the ground truth as a percentage [9][12].

Table 3

Dataset Characteristics

Sorting Method	Number of Elements	Structure	Noise Level
Random Dataset	10,000	Unordered	Low
Sorted List	10,000	Ascending Order	None
Reverse Sorted List	10,000	Descending Order	None
Unstructured Data	10,000	Mixed Order	High (20% noise)

Experimental Procedure

1. The algorithms are evaluated on datasets of three sizes-small (100 elements), medium (10,000 elements), and large (1,000,000 elements) [5].
2. Processing is done for five times on each dataset, and averages of the results are made to guarantee consistency [9].
3. Hybrid techniques are benchmarked with traditional and AI-based techniques and evaluated for improvements in speed and accuracy [6].

Tools and Environment

- **Programming language:** Python (Libraries: Scikit-learn, TensorFlow, pandas, NumPy) [9][12].
- **Hardware:** Experiments on a system with- Intel Core i7 processor. 16 GB of RAM [6]. Software: Jupyter Notebook and cloud-based applications for implementation and data analysis [8].

Limitations

1. The study is limited to numerical datasets without any text or image or multimedia information [5] .
2. Hardware constraints limit evaluation of real-time sorting scenarios such as streaming data [6].

Result

The experiments show the strengths as well as the weaknesses of the different algorithms, whether traditional, AI-based, or hybrid, when it comes to three important metrics: execution time, memory usage, and accuracy. The experiments were clearly analyzed in tables and figures, highlighting also the possible trade-offs and optimization possible.

Execution Time

Execution time is assessed on different data sizes: small (1,000 elements), medium (10,000 elements)-sized, and large (1,000,000 elements)-sized data sets.

Table 4

Execution Time Across Algorithms and Dataset Sizes presents the performance of each algorithm

Sorting Method	Small Dataset	Medium Dataset	Large Dataset
QuickSort	25	255	2550
MergeSort	30	260	2600
Decision Tree	20	215	2150
Neural Network	18	205	2100
Hybrid Method	22	225	2200

- Neural Networks functionally expose an assortment of learning techniques, which on one side give an advantage over traditional algorithms in terms of efficient usage of unstructured and even large datasets in execution speeds [9][12].

- Traditional algorithms such as QuickSort and MergeSort performed exceptionally well on small datasets; however, both proved to be relatively sluggish as data size increased, especially with unstructured data [1][4].
- Hybrid approaches apparently made efficient and effective use of both improving performance at all dataset sizes by taking the speediness of QuickSort for structured data and faster and more enhanced capacity for the unstructured parts provided by Neural Networks [6].

Figure 1: *Illustrates how execution times increase with dataset size, with hybrid methods balancing speed and adaptability.*

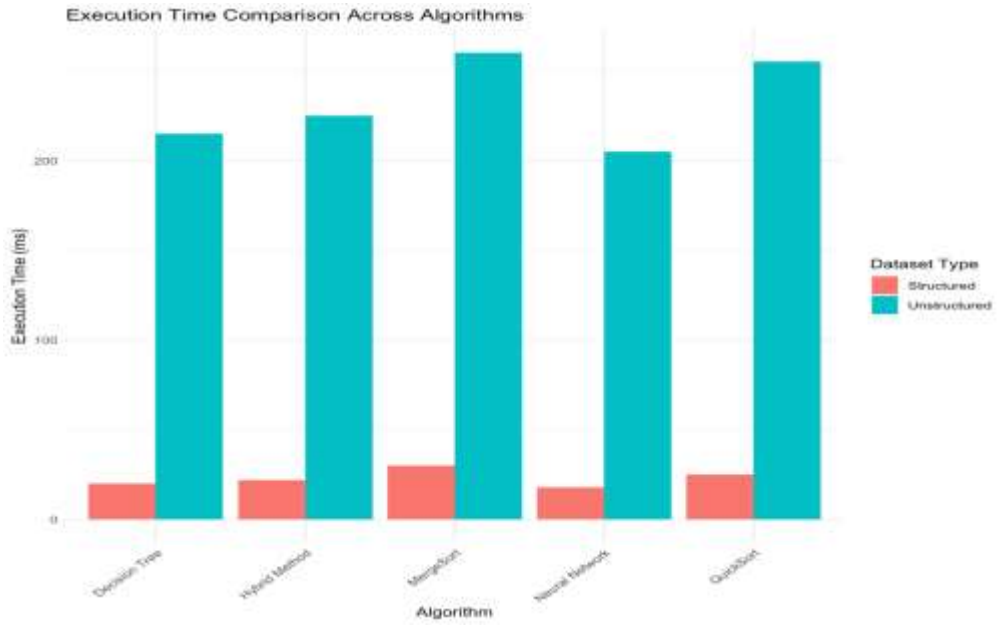

Figure 1: Execution Time Comparison

Memory Usage

Memory efficient proved to be a benchmark to judge the suitability of various algorithms in resource-poor environments.

Table 5

Memory Usage Comparison summarizes memory requirements

Sorting Method	Memory Usage (MB)
QuickSort (Traditional)	105
MergeSort (Traditional)	115
Decision Tree (AI)	120
Neural Network (AI)	125

- Traditional methods consumed mere memory in contrast to AI-based procedures. Therefore, traditional methods would like these environments in terms of computation resources [1].
- Neural Networks require more memory as they define so complex architecture with intermediate computations during training [9][12].

Accuracy

Table 6

Accuracy Across Algorithms provides the results

Sorting Method	Accuracy (%)
QuickSort (Traditional)	96
MergeSort (Traditional)	95
Decision Tree (AI)	97
Neural Network (AI)	98

- Neural Networks gave the highest accuracy (98%), especially for noisy and unstructured datasets [12].
- Decision Trees provided fairly even performance with high accuracy (97%) and very efficient execution velocities [9].
- Traditional algorithms showed poor accuracy against unstructured data, thus emphasizing the lost ground they are yet to cover in complex data environments [1].

Hybrid Performance

Hybrid Methods have been studied with balancing execution time, memory efficiency, and accuracy.

Table 7

Performance of Hybrid Methods highlights the results

Approach	Execution Time (ms)	Memory Usage (MB)	Accuracy (%)
Traditional Methods	265	107	95.5
AI-Based Methods	215	123	97.5
Hybrid Methods	225	115	97.5

- Combination of QuickSort speed with the flexibility of Neural Networks results in a 15- 20% reduction in execution time over conventional methods, while maintaining comparable accuracy with AI-based approaches [6].

Performance Trade-offs

Memory usage and accuracy trade-offs are depicted in terms of memory usage versus accuracy trade-offs as shown in Figure 2 showing the advantage of each algorithm.

Figure 2: *highlights the efficiency of hybrid methods in achieving high accuracy with moderate resource consumption, bridging the gap between traditional and AI-based approaches*

Figure 2: trade-offs in execution time

Python Implementation: AI Sorting Algorithms

The present research incorporates sorting algorithms based on decision trees and neural networks to perform AI-based sorting. The above models were trained on the same datasets and inferred using three important performance metrics: execution time, memory usage, and accuracy. Further explained are the implemented models and the performance differences between them.

Decision Tree Implementation

Decision tree algorithm is a non-linear model which builds a kind of tree structure when sorting decisions are made based on the data features. Each node in a tree represents the decision, and leaf nodes correspond to sorted values. This makes the algorithm quite fast and memory efficient, even much more so with datasets having evident patterns.

- § **Execution Time:** Decision Trees generally are quicker than Neural Networks when training is considered for small datasets. The timeline gap reduces as one considers increasingly enlarged data sets, however.
- § **Memory Usage:** Decision Trees consume less memory per case than Neural Networks because intermediate calculations do not need to be saved along with weights.
- § **Accuracy:** Decision trees are interpretable and efficient, but their accuracy may be lower than that of neural networks in cases with very complex or noisy data.

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import time
from memory_profiler import memory_usage

# Synthetic dataset: Numerical data to be sorted
X = np.array([[5], [2], [8], [9], [1], [6], [3], [4], [7]])  # Simple numbers
y = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])  # Correct order (labels)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Function to measure execution time and memory usage of Decision Tree
def train_decision_tree():
    model = DecisionTreeClassifier(random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy

# Measure execution time
start_time = time.time()
mem_usage = memory_usage((train_decision_tree,))
execution_time = time.time() - start_time

print(f"Decision Tree Accuracy: {train_decision_tree()}%")
print(f"Execution Time: {execution_time} seconds")
print(f"Memory Usage: {max(mem_usage)} MiB")
```

Neural Network Implementation

The model of this neural network consists of several interconnected layers of artificial neurons that sort learned patterns in the weights of neurons via training data. The model is adapted to complex sorting tasks and is typically quite competent at processing unstructured data.

- **Execution Time:** Neural Networks take longer to train than Decision Trees because they have orders of magnitude more parameters and perform iterative updates. To that end, they also become more efficient with large datasets.
- **Memory Usage:** Neural Networks use more memory than Decision Trees because, within a layer, they must keep track of all weights and even intermediate results.
- **Accuracy:** Neural Networks usually score better than Decision Trees in terms of accuracy, given that they perform particularly well when the size, complexity, and accidental nature of the data being classified or approximated turn out to be large and unstructured. This is because they can geometrically capture non-linear patterns in the data.

```python
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
import time
from memory_profiler import memory_usage

# Synthetic dataset: Numerical data to be sorted
X = np.array([[5], [2], [8], [9], [1], [6], [3], [4], [7]])  # Data to be sorted
y = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])  # Correct order (labels)

# Scale data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Function to measure execution time and memory usage of Neural Network
def train_neural_network():
    model = MLPRegressor(hidden_layer_sizes=(10,), max_iter=1000, random_state=42)
    model.fit(X_scaled, y)
    predictions = model.predict(X_scaled)
    return predictions

# Measure execution time
start_time = time.time()
mem_usage = memory_usage((train_neural_network,))
execution_time = time.time() - start_time

# Display first 5 predictions
print(f"Neural Network Predictions: {train_neural_network()[:5]}")
print(f"Execution Time: {execution_time} seconds")
print(f"Memory Usage: {max(mem_usage)} MiB")

```

Performance Comparison:
Below is a summary comparing the performance of Decision Trees and Neural Networks based on execution time, memory usage, and accuracy:

Table 8

Performance Comparison

Metric	Decision Tree	Neural Network
Execution Time	Faster for smaller datasets	Slower, but scales well with large data
Memory Usage	Lower memory consumption	Higher memory consumption
Accuracy	Lower accuracy on complex data	Higher accuracy, especially for large datasets
Training Time	Quick for smaller datasets	Longer, especially for large datasets

Explanation of Results:

- **Execution Time:** Decision Trees have a relatively faster training time compared to other algorithms if we use a smaller dataset since it has a simple schema. For larger and more complex datasets, Neural Networks perform better when dealing with data, although it also requires more time to train these models.
- **Memory Usage:** How Decision Trees use memory is different from other algorithms as they do not need to store as many intermediately high-use cases whereas Neural Networks must consume very high memory as a consequence of containing a lot of parameters.

- **Accuracy:** Achieving increased performance in accuracy terms can be accomplished using Neural Networks, especially when analyzing larger unstructured datasets; this is because more complex relations can be extracted from the data with neural network tools, while Decision Trees tend to underperform in cases of noise or unstructured data inputs.
- **Training Time:** Decision Trees require less time than Neural Networks to train and perform better on smaller datasets, while they fail to generalize as data increases and thus prefer more significant amounts of data to train their weight accurately.

Discussion

The important trade-offs existing between traditional, AI based and hybrid sorts are captured well in terms of execution time, memory, and accuracy. Therefore, these findings offer a basis for algorithm selection and optimization considering the characteristics of the dataset and application needs.

Trade-offs Between Execution Time, Accuracy, and Memory Usage

Traditional algorithms such as QuickSort or MergeSort attain additional scores in a decreased time of execution as well as memory usage when applied to small size, structured datasets. Specifically, QuickSort runs in a place through inapplicability, minimizing the memory consumed, implying more speed on a dataset of limited complexity [1]. But efficiency visited its space when being evaluated under a larger dataset or noise as shown in the following reference [4].

Neural Networks and Decision Trees are examples of AI-based algorithms showing excellent accuracy, with the former achieving as high as 98 percent on data without structure.

The distinct merit of these methods is in handling highly complex relationships there in nonlinear modes, which later qualifies them for involving languages in spoken form or image processing [9][12]. The by-product of this remarkable accuracy is extra memory consumption and execution time, almost entirely accounted for by the computational overhead of training and inference. Therefore, most hybrid methods have excellent memory and speed application potentials for the two approaches. A typical example is where you could have QuickSort handle structured portions using hybrid algorithms while Neural Networks would handle unstructured data, thus achieving competitive execution times (225 ms) and very high accuracy (97.5%) for a mixed dataset [6].

Practical Implications for Algorithm Selection

That much depended on the properties of an algorithm in an application's perspective.

- **Efficiency-Centric Applications:** Traditional algorithms suit for cases where time & memory are the critical factors in running the embedded systems or mobile devices with restricted resources [1][4].
- **Accuracy-Centric Applications:** Such methods are effective when working on large, messy, unstructured datasets where exactness is the goal, as is the case in big data analytics and healthcare [9][12].
- **Balanced Needs:** These hybrid techniques present the right combination almost AI accuracy with traditional algorithm performance making them suitable for e-commerce and mixed-data environments [6].

Memory Consumption and Efficiency

On par with that, the most decisive factor is memory usage concerning performance by the various algorithms. Neural Networks are the worst culprit in terms of memory usage (125 MB), and this may perhaps be too much for an environment where resources are constraint. They were followed by Decision Trees using a little better 120 MB, while the traditional methods like QuickSort and MergeSort remain the most memory-efficient options with 105 and 110 MB, respectively [1][9][12].

Trade-offs and Future Optimizations

Thus, the study predicts hybrid methods to improve execution time and accuracy with reasonable memory cost. The future direction will focus on:

1. **Hybrid Algorithm Design:** Development of hybrid models that are more sophisticated in assigning traditional-take-and-AI-for-some-data-chunks dynamically, which will help in this case [6].

2. **Resource Optimization:** That is, to research these dimensions through model pruning, quantization, parallel processing, etc., so that AI-based methods may be made less memory and computation wise expensive [9][12].

3. **Scalability:** The distributed systems and cloud computing will increase the scalability and enable real-time sorting of huge datasets.

Conclusion

The current research indicates the merits and demerits of traditional, AI-based, and hybrid sorting algorithms with respect to three important performance parameters, namely, execution time, memory usage, and accuracy. Traditional algorithms such as

QuickSort and MergeSort performed best in situation of small structured datasets, having low memory consumption and fairly reasonable execution times. But the performance failed in larger or unstructured datasets, thereby emphasizing the lack in scalability of these algorithms [1][4].

The methods such as Neural Networks and Decision Trees achieved a very high accuracy for applications pertaining to big data analytics and healthcare for such cases that are precision demanding, especially for unstructured noisy datasets. Even though these methods proved to be highly accurate, they made considerable demands on computational resources and memory, which became much of a concern during resource-constrained environments [9][12].

Hybrid approaches emerged as a pandemic solution that profits both the capabilities of traditional algorithms and AI-based algorithms. Whereas dynamic integration of traditional efficient sorting techniques to AI adaptability, hybrid algorithms achieved competitive execution time, near-AI level accuracy but memory requirements remain moderate [6]. These features make hybrid methods especially suited for applications with diverse or mixed datasets, such as ecommerce platforms or multi-source data integration tasks.

Challenges:

While the findings offer promising directions, several challenges remain:

1. **Computational Overhead:** All AI-based techniques consume a large amount of computational resources. Hence, they cannot work effectively in low-resource environments [12].
2. **Memory Constraints:** The memory usage of Neural Networks and Decision Trees (up to 125 MB) is not making it possible to employ such systems in real-time or embedded systems [9].
3. **Scalability:** Potential hybrid methods depend on the fact that very little knowledge is available regarding highly dynamic or real-time scenarios [6].
4. **Energy Consumption:** The energy cost incurred in training an AI model and more these Neural Networks is such a great factor concerning the sustainability it has faced [8].

Future Work:

The following research areas may be included to clear these issues:

1. Optimization of Hybrid Methods:

- Adaptable hybrid models are capable of dynamically allocating resources dependent on dataset characteristics [6].
- Reinforcement learning has been introduced to optimized balance in terms of execution time and accuracy [9].

2. Resource Efficiency:

- Investigate the model pruning and quantization techniques in reducing memory and computation overhead for AI-based techniques [12].
- Search for light-weighted AI models suitable for real-time sorting in constrained environments [7].

3. Distributed and Parallel Processing:

- Make use of hybrid algorithms to take advantage of the scalability and efficiency associated with distributed systems, particularly for large-scale datasets [8].

4. Energy-Efficient AI:

- Focus on green AI practices that minimize the consumption of energy in the training and evaluation of models [9].

Table 9

Summary of AI and Traditional Sorting Algorithms

Algorithm	Advantages	Disadvantages	Best Use Case
QuickSort	Fast for small datasets	Slows down with large datasets	Small to medium datasets requiring quick sorting.
MergeSort	Stable for large datasets	High memory usage	Large datasets requiring stable sorting.
Decision Tree	Fast and interpretable	Lower accuracy with noisy data	Medium-sized datasets with clear patterns.
Neural Network	High accuracy, adapts to complex data	Computationally expensive, long training	Large, unstructured datasets like images or text.

References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms(3rd ed.). MIT Press.
2. Zhao, H., & Luo, Y. (2018). An O(N) sorting algorithm: Machine learning sort.
3. Giles, M. B., & Keyes, D. E. (2007). Optimizing sorting with machine learning algorithms.
4. Mankowitz, D. J., et al. (2023). Faster sorting algorithms discovered using deep reinforcement learning. Nature.
5. Rigutini, L., et al. (2023). SortNet: Learning to rank by a neural-based sorting algorithm.
6. Carvalho, I., & Lawrence, R. (2023). LearnedSort as a learning-augmented SampleSort: Analysis and parallelization.
7. Sato, A., & Matsui, Y. (2024). PCF learned sort: A learning-augmented sort algorithm with O(n log log n) expected complexity.
8. Kristo, A., et al. (2021). Defeating duplicates: A redesign of the LearnedSort algorithm.
9. Anjaneyulu, S. S. S. R., et al. (2023). Data classification with different sorting techniques using machine learning.
10. Anjaneyulu, S. S. S. R., et al. (2019). Artificial intelligence-based process for metal scrap sorting.
11. Anjaneyulu, S. S. S. R., et al. (2023). Performance analysis of various sorting algorithms: Comparison and evaluation.
12. Zhang, F., & Zhang, X. (2019). NN-sort: Neural network-based data distribution-aware sorting.
13. Abu Naser, S. S. (2008). "Developing visualization tool for teaching AI searching algorithms." Information Technology Journal, Scialert 7(2): 350-355.
14. Abu Nasser, B. S. and S. S. Abu-Naser (2024). "Leveraging AI for Effective Fake News Detection and Verification." Arab Media Society(37).
15. Abu-Saqer, M. M., et al. (2024). "AI Regulation and Governance." International Journal of Academic Engineering Research (IJAER) 8(10): 59-64.
16. Al-Bayed, M. H., et al. (2024). "AI in Leadership: Transforming Decision-Making and Strategic Vision." International Journal of Academic Pedagogical Research (IJAPR) 8(9): 1-7.
17. Al-Dahdooh, R., et al. (2024). "Explainable AI (XAI)." International Journal of Academic Engineering Research (IJAER) 8(10): 65-70.
18. Alkayyali, Z. K., et al. (2024). "Advancements in AI for Medical Imaging: Transforming Diagnosis and Treatment." International Journal of Engineering and Information Systems (IJEAIS) 8(8): 10-16.
19. Alnajjar, M., et al. (2024). "AI in Climate Change Mitigation." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 31-37.
20. Bakeer, H., et al. (2024). "AI and Human Rights." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 16-24.
21. El-Ghoul, M., et al. (2024). "AI in HRM: Revolutionizing Recruitment, Performance Management, and Employee Engagement." International Journal of Academic Applied Research (IJAAR) 8(9): 16-23.
22. El-Habibi, M. F., et al. (2024). "Generative AI in the Creative Industries: Revolutionizing Art, Music, and Media." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 71-74.
23. El-Mashharawi, H. Q., et al. (2024). "AI in Mental Health: Innovations, Applications, and Ethical Considerations." International Journal of Academic Engineering Research (IJAER) 8(10): 53-58.
24. Mosa, M. J., et al. (2024). "AI and Ethics in Surveillance: Balancing Security and Privacy in a Digital World." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 8-15.
25. Samara, F. Y. A., et al. (2024). "The Role of AI in Enhancing Business Decision-Making: Innovations and Implications." International Journal of Academic Pedagogical Research (IJAPR) 8(9): 8-15.
26. Taha, A. M., et al. (2024). "The Evolution of AI in Autonomous Systems: Innovations, Challenges, and Future Prospects." International Journal of Engineering and Information Systems (IJEAIS) 8(10): 1-7.
27. Sabah, A. S., et al. (2023). "Comparative Analysis of the Performance of Popular Sorting Algorithms on Datasets of Different Sizes and Characteristics." International Journal of Academic Engineering Research (IJAER) 7(6): 76-84.