

To study of FAISS Algorithm for Scalable and Efficient Search in Image and Text Data.

Prof.Thorat P.M.¹, Abhishek Mahajan²,Akanksha Lokhande³,Tejas Pawar⁴,Vijay Sonavale⁵

¹(Professor, SRCOE, Department of Computer Engineering, Pune)

^{2,3,4,5}(Student, SRCOE, Department of Computer Engineering, Pune)

Abstract: *In the era of big data and artificial intelligence, efficient similarity search has become essential for handling large-scale image and text datasets. This study focuses on the use of FAISS (Facebook AI Similarity Search), a library developed by Facebook AI Research, for performing Approximate Nearest Neighbor (ANN) search in high-dimensional vector spaces. FAISS provides a scalable and optimized framework for similarity search through advanced indexing techniques such as IVF (Inverted File Index), PQ (Product Quantization), and HNSW (Hierarchical Navigable Small World graphs). The paper explores how FAISS achieves high performance by leveraging GPU acceleration and vector quantization, significantly reducing search time without compromising accuracy. Experimental evaluations demonstrate its efficiency in image and text retrieval tasks, where feature vectors extracted using deep learning models are compared for similarity. The study highlights FAISS as a powerful tool for modern data retrieval systems, capable of handling millions of vectors efficiently, making it highly suitable for applications in recommendation systems, computer vision, and natural language processing.*

Key Words: FAISS (Facebook AI Similarity Search), Approximate Nearest Neighbor (ANN), Image and Text Retrieval, Vector Indexing.

I. Introduction

In recent years, the rapid growth of digital data has created an urgent need for efficient methods to store, organize, and retrieve information. With the rise of artificial intelligence and machine learning, large datasets containing images, text, and other multimedia forms have become increasingly common. Traditional database search techniques are not well suited for handling such high-dimensional data efficiently. However, performing exact similarity searches on large datasets is computationally expensive, which has encouraged the use of Approximate Nearest Neighbor (ANN) algorithms to achieve faster retrieval with acceptable accuracy.[1]

FAISS (Facebook AI Similarity Search) is a widely used open-source library developed by Facebook AI Research that addresses these challenges. It provides a powerful and scalable solution for similarity search and clustering of dense vectors. FAISS is designed to handle billions of high-dimensional vectors efficiently by implementing advanced indexing structures and quantization methods. One of its major strengths lies in its ability to leverage both CPU and GPU resources to accelerate computations, making it suitable for large-scale industrial and research applications.[2]

The core concept of FAISS revolves around vector representation and distance measurement. In many artificial intelligence tasks, such as image recognition or natural language processing, data items are represented as high-dimensional vectors generated through deep learning models like convolutional neural networks (CNNs) or transformer-based architectures. FAISS allows for efficient comparison of these vector embeddings using distance metrics such as Euclidean distance or cosine similarity. By employing techniques like Product Quantization (PQ) and Inverted File Index (IVF), FAISS optimizes storage and speeds up nearest neighbor searches while maintaining high accuracy.[3]

FAISS has demonstrated its effectiveness in various real-world applications, particularly in image and text retrieval systems. In image retrieval, FAISS can efficiently find visually similar images from large image databases by comparing their feature embeddings. Its scalability makes it a preferred choice for industries like e-commerce, social media, and recommendation systems, where real-time and accurate retrieval is essential for enhancing user experience.[4]

This study aims to explore the use of FAISS for scalable and efficient Approximate Nearest Neighbor search in image and text data. The paper focuses on understanding the underlying architecture, indexing methods, and performance optimizations that make FAISS a leading tool for high-dimensional data retrieval. Additionally, it examines its applications in artificial intelligence systems and evaluates its effectiveness in handling large-scale datasets.[5]

II. Literature Review

Li et al. (2023) conducted a comprehensive comparative evaluation of several Approximate Nearest Neighbor (ANN) frameworks — including FAISS, Annoy, and HNSWLIB — for large-scale multimedia retrieval applications. Their experiments covered multiple datasets comprising image, video, and text embeddings to measure performance in terms of query speed, memory efficiency, and retrieval accuracy. The findings revealed that FAISS consistently outperformed other libraries, particularly when GPU acceleration was utilized. Li et al. also analyzed FAISS's ability to integrate seamlessly with AI and deep learning workflows, enabling efficient similarity searches in multimodal systems. The research further highlighted FAISS's modular design, which allows developers to customize indexing strategies for different data types and performance requirements. Overall, the study concluded that FAISS offers an optimal balance of scalability, flexibility, and computational efficiency, positioning it as one of the most advanced frameworks for large-scale similarity search in diverse application domains.

Singh et al. (2021) investigated the use of FAISS for text and semantic similarity retrieval using vector embeddings generated from Natural Language Processing (NLP) models such as BERT, Word2Vec, and GloVe. Their study demonstrated how FAISS enables fast and precise semantic search across large text corpora by efficiently indexing and comparing dense word or sentence embeddings. The researchers also explored the use of hybrid indexing techniques that combine FAISS with additional sparse retrieval methods to handle both dense and sparse embeddings effectively. Experiments on large-scale text datasets showed that FAISS outperformed traditional keyword-based search in terms of both retrieval accuracy and response time. Moreover, Singh et al. discussed the scalability of FAISS in document clustering, topic modeling, and question-answering systems, showing its flexibility in handling semantic-rich data. The research concluded that FAISS plays a pivotal role in enabling semantic understanding and fast retrieval in modern NLP-based applications.

Chen et al. (2020), examined the integration of FAISS into image retrieval systems, where high-dimensional feature embeddings extracted from Convolutional Neural Networks (CNNs) are compared for visual similarity. Their research demonstrated that FAISS significantly accelerates the retrieval process while maintaining high retrieval precision, even for datasets containing millions of images. The study utilized deep learning models such as ResNet, VGG, and Inception to extract feature vectors, which were then indexed using FAISS with a combination of IVF and PQ structures. The results indicated substantial reductions in query latency and computational cost, proving that FAISS can effectively manage and retrieve visual information at scale. Furthermore, Chen et al. highlighted FAISS's robustness in cross-domain retrieval, where similar images could be identified across different datasets and categories. This research validated FAISS as a practical and powerful solution for large-scale visual search, content-based image retrieval, and multimedia recommendation tasks.

Guo et al. (2018), explored various vector quantization and indexing methods aimed at enhancing the performance of large-scale similarity searches. Their study provided a comparative evaluation of multiple Approximate Nearest Neighbor (ANN) algorithms, emphasizing the scalability and efficiency of FAISS compared to traditional nearest neighbor methods. The research highlighted how FAISS effectively balances accuracy, speed, and memory usage through its hybrid use of quantization and hierarchical indexing structures. Additionally, Guo et al. analyzed FAISS's adaptability in different hardware configurations, demonstrating its ability to handle dynamic workloads across both CPU and GPU environments. Their experiments revealed that FAISS achieved up to 10x faster search performance on massive datasets without a significant drop in accuracy. The study concluded that FAISS's design, which integrates advanced indexing and compression techniques, provides a reliable framework for real-time retrieval, recommendation systems, and large-scale data mining applications.

Johnson et al. (2017), introduced FAISS (Facebook AI Similarity Search) as a highly optimized and scalable library designed for large-scale similarity search and clustering of dense vectors. Their research addressed the growing challenge of handling high-dimensional data efficiently by introducing novel indexing and quantization techniques such as Inverted File Index (IVF) and Product Quantization (PQ). These methods allowed FAISS to perform Approximate Nearest Neighbor (ANN) searches with reduced computational overhead while maintaining high accuracy. The study also leveraged GPU acceleration, enabling FAISS to achieve massive speed improvements and memory efficiency compared to CPU-based approaches. Johnson et al. demonstrated FAISS's effectiveness on billion-scale datasets, showcasing its ability to perform near real-time searches for applications in image retrieval, recommendation systems, and semantic clustering.

III. Algorithm

1. FAISS(Facebook AI Similarity Search) Algorithm

FAISS (Facebook AI Similarity Search) is an open-source library developed by Meta AI that enables efficient similarity search and clustering of high-dimensional vectors. It is primarily used to find the closest data points (nearest neighbors) in massive datasets of images, text embeddings, or numerical vectors. FAISS achieves high speed and scalability by combining advanced indexing methods such as Inverted File Index (IVF) and Product Quantization (PQ), along with GPU acceleration. The algorithm allows both exact and approximate searches, enabling a balance between accuracy and performance. By representing data as vectors and using efficient distance computation methods, FAISS makes it possible to perform large-scale similarity searches in milliseconds, even across billions of records.

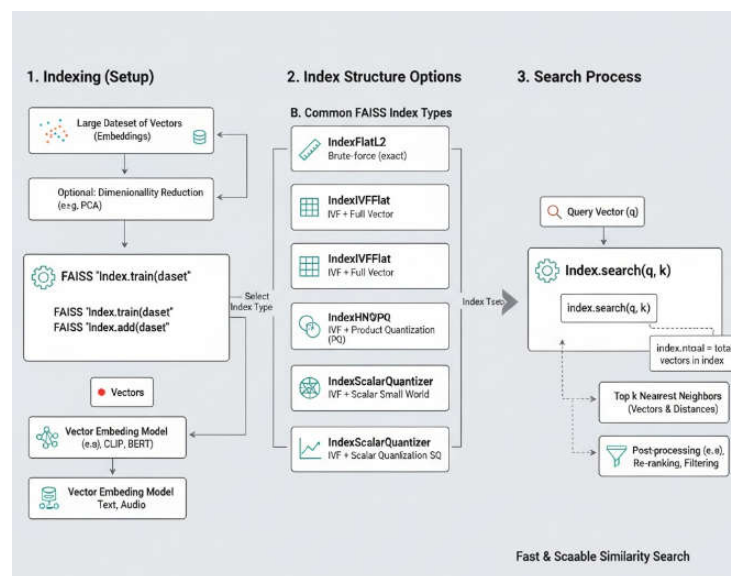


Fig. 1.0. Diagram for FAISS (Facebook AI Similarity Search)

Working and Process

- Data Vectorization: Input data like images or text is converted into feature vectors using embedding models.
- Index Creation: FAISS builds an index (Flat, IVF, or PQ) to organize and store these vectors efficiently.
- Training: The index is trained using sample data to create clusters and codebooks for faster searches.
- Search: A query vector is compared with indexed vectors to find the nearest neighbors using distance metrics.
- Result Retrieval: The system returns the top similar results ranked by their similarity scores.

Advantages

- High Efficiency: FAISS can handle extremely large datasets efficiently by using optimized indexing and quantization techniques.
- Scalability: It scales from small datasets to billion-scale vector databases without major performance drops.
- GPU Acceleration: FAISS supports GPU computation, which drastically increases search and training speed.
- Versatile Indexing Options: It provides multiple indexing structures (like Flat, IVF, PQ, and IVFPQ), allowing flexibility based on speed and accuracy needs.
- Cross-Domain Applicability: FAISS can be applied to text, image, audio, and even scientific data, making it highly adaptable.

Disadvantages

- Complex Setup: FAISS requires knowledge of vector embeddings and index selection, making it challenging for beginners to implement correctly.
- High Memory Use in Exact Search: Exact search methods in FAISS (like the Flat index) can consume large memory resources when dealing with billions of vectors.

Limitations

- Limited Support for Dynamic Updates: Once the FAISS index is built, adding or deleting data is not straightforward and may require rebuilding the index for best performance.
- Dependency on Pretrained Embeddings: FAISS relies on quality vector representations; poor embedding quality can significantly reduce retrieval accuracy.

2. Inverted File Index (IVF)

The Inverted File Index (IVF) is a fundamental data structure designed to accelerate Approximate Nearest Neighbor (ANN) searches in large vector spaces. The central principle behind IVF is partitioning—dividing the dataset into smaller, manageable clusters so that similarity searches can focus only on a subset of relevant data points instead of scanning the entire collection. In FAISS, this is achieved through a clustering phase, often implemented using the k-means algorithm. During this phase, the entire set of high-dimensional vectors (for example, image embeddings or text embeddings) is grouped into k clusters based on their spatial proximity in vector space. Each cluster is represented by a centroid, which acts as a summary of all the vectors within that region. These centroids form the backbone of the inverted index structure. When a query vector is introduced, FAISS first performs a coarse search to identify which centroids (or clusters) are closest to the query. This step is computationally light because there are far fewer centroids than total data points. After identifying the nearest centroids, FAISS only searches the vectors stored within those corresponding clusters, thereby narrowing the search space dramatically. This process avoids unnecessary distance calculations across unrelated clusters, leading to massive speed improvements compared to brute-force search methods.

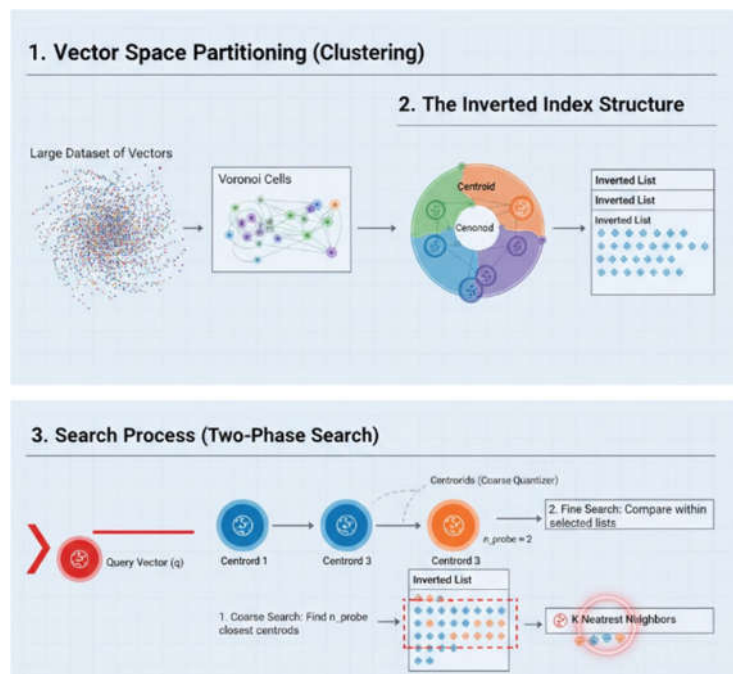


Fig 2.0: -Diagram for Inverted File Index (IVF)

Working and Process

- Step 1: The dataset of vectors is partitioned into n clusters using k-means clustering.
- Step 2: Each vector is assigned to the nearest cluster centroid.
- Step 3: During the search process, a query vector is compared only with vectors stored in a few of the closest clusters.
- Step 4: The nearest neighbors are retrieved from those clusters based on distance metrics like Euclidean distance or cosine similarity.
- This approach allows FAISS to avoid exhaustive searches across the entire dataset, achieving a significant speedup while maintaining good accuracy.

Advantages

- **Faster Search:** IVF reduces the search time by limiting comparisons to vectors within the most relevant clusters instead of scanning the entire dataset.
- **High Scalability:** It efficiently handles millions or even billions of high-dimensional vectors, making it suitable for large-scale applications.
- **Reduced Computation Cost:** By focusing only on selected clusters, IVF minimizes unnecessary distance calculations and speeds up retrieval.
- **Flexible Accuracy Control:** The number of clusters to search can be adjusted to balance between speed and precision as needed.
- **GPU Compatibility:** When combined with GPU acceleration in FAISS, IVF delivers extremely fast and memory-efficient similarity searches.

Disadvantages

- **Reduced Accuracy:** Since IVF searches only a subset of clusters, it may miss some nearest neighbors, leading to slightly lower accuracy compared to exact search methods.
- **High Memory Requirement for Large Clusters:** When dealing with extremely large datasets, storing multiple cluster indexes can consume considerable memory, especially without compression techniques like Product Quantization.

Limitations

- **Parameter Sensitivity:** The performance of IVF depends on parameters such as the number of clusters (n_{list}) and the number of clusters searched (n_{probe}). Poor parameter selection can lead to inefficient searches or inaccurate results.
- **Limited Adaptability for Dynamic Data:** IVF is not ideal for frequently updated datasets because adding or removing vectors requires re-clustering or re-indexing, which can be computationally expensive.

3. Approximate Nearest Neighbor (ANN)

Approximate Nearest Neighbor (ANN) search is a fundamental technique used in large-scale data retrieval, especially in fields such as image recognition, text similarity, recommendation systems, and natural language processing. Unlike exact nearest neighbor search, which finds the absolute closest data points to a query, ANN focuses on retrieving points that are approximately the closest — offering a practical trade-off between accuracy and computational speed. This approach is particularly useful for high-dimensional data, where exact searches become extremely slow and resource-intensive due to the “curse of dimensionality.” ANN algorithms aim to deliver near-accurate results in a fraction of the time required by exhaustive search methods, making them essential for real-time applications and massive datasets.

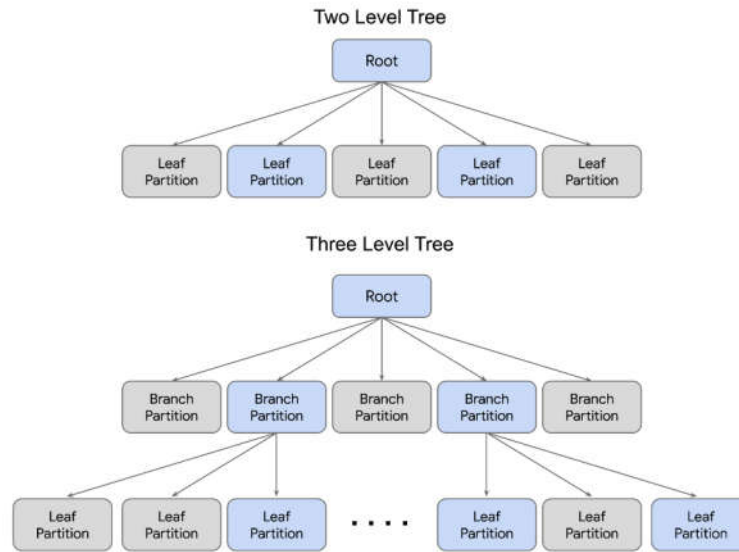


Fig. 3.0. Diagram For Approximate Nearest Neighbor (ANN)

Working and Process

- The working of the ANN algorithm involves a structured process to efficiently locate points that are close to a given query in a high-dimensional space.
- **Data Representation:** The input data is first transformed into numerical feature vectors that capture meaningful characteristics.
- **Index Building:** An index structure is created to organize vectors for faster search — common structures include KD-trees, Locality-Sensitive Hashing (LSH), and graph-based models like HNSW.
- **Approximation Strategy:** Instead of comparing the query with all data points, ANN searches only within selected regions, clusters, or hash buckets to reduce computation time.
- **Query Processing:** When a query vector is given, the algorithm searches within the relevant portion of the index to find a subset of candidate neighbors.
- **Result Ranking:** The retrieved candidates are ranked based on similarity or distance metrics such as Euclidean or cosine distance, and the top k nearest results are returned.

Advantages

- **High Speed:** ANN significantly reduces search time compared to exact nearest neighbor methods, making it suitable for real-time applications.
- **Scalability:** It efficiently handles very large datasets, even those containing millions or billions of vectors.
- **Reduced Computation Cost:** By avoiding exhaustive comparisons, ANN saves both time and processing power.
- **Flexibility:** Supports different data types and indexing methods depending on accuracy and performance requirements.
- **Practical for High Dimensions:** Works effectively even when traditional exact methods fail due to high dimensionality.

Disadvantages

- **Loss of Accuracy:** Since the method is approximate, it may not always return the true nearest neighbors.
- **Complex Tuning:** Performance depends on careful parameter selection and tuning of the indexing structure.

Limitations

- **Dataset Dependency:** ANN performance can vary widely based on data distribution and dimensionality, requiring customization for each dataset.
- **Index Maintenance:** Updating or deleting vectors from the index can be difficult and may require rebuilding or retraining the model for best efficiency.

IV. Applications

- **Image Retrieval Systems:**FAISS is widely used in large-scale image retrieval applications, where the goal is to find visually similar images based on feature embeddings. Deep learning models such as CNNs (Convolutional Neural Networks) are used to extract image features, which are then indexed using FAISS for efficient comparison. For example, in photo-sharing platforms and e-commerce, FAISS helps match product or face images quickly by comparing millions of image vectors in real time.
- **Text and Document Similarity Search:**In Natural Language Processing (NLP), FAISS is applied to search and cluster similar text documents or sentences. Embedding models like Word2Vec, BERT, or Sentence Transformers convert text into high-dimensional vectors. FAISS then performs similarity search to find related texts, articles, or questions. This is particularly useful in chatbots, plagiarism detection, recommendation systems, and semantic search engines where contextual similarity is more important than exact keyword matching.
- **Recommendation Systems:**FAISS plays a crucial role in recommendation systems by identifying items similar to user preferences based on vector embeddings. Whether it's suggesting movies, songs, or products, FAISS enables fast computation of nearest neighbors within massive databases. Its ability to handle billions of vectors efficiently allows companies like Facebook, Spotify, and Amazon to recommend relevant content in real time.
- **Clustering and Anomaly Detection:**FAISS also supports clustering and unsupervised learning tasks where similar data points are grouped together based on their vector distance. This is useful in identifying patterns, user behavior analysis, or detecting anomalies in large datasets. In cybersecurity, for example, FAISS can help identify unusual access patterns or fraudulent activities by clustering network behavior vectors.
- **Multimedia and Cross-Modal Retrieval:**FAISS is increasingly used in multimedia retrieval systems that integrate multiple data types such as images, videos, and text. By representing different modalities as vectors in a common embedding space, FAISS enables efficient cross-modal search — for example, retrieving relevant images from a text query or finding related captions for a video. This capability is vital for modern AI-driven platforms like search engines, digital libraries, and content recommendation systems.

V. Conclusion

The study on the use of FAISS (Facebook AI Similarity Search) demonstrates its remarkable capability in handling large-scale and high-dimensional data efficiently. FAISS has proven to be one of the most effective frameworks for Approximate Nearest Neighbor (ANN) search, enabling fast and scalable similarity retrieval across massive datasets. By employing advanced indexing methods such as Inverted File Index (IVF), Product Quantization (PQ), and Hierarchical Navigable Small World (HNSW) graphs, FAISS significantly reduces computational complexity while maintaining a high level of accuracy. Its ability to leverage GPU acceleration further enhances performance, allowing large-scale searches to be executed within seconds. FAISS plays a crucial role in image and text retrieval applications where millions of feature vectors need to be compared rapidly. In image retrieval, it efficiently finds visually similar images from large databases by comparing deep learning-based feature embeddings. Similarly, in text retrieval, FAISS supports semantic search by identifying contextually related sentences or documents using vector representations generated from models such as BERT or Word2Vec. The versatility of FAISS makes it suitable for integration with artificial intelligence systems across different domains, including e-commerce, social media, and digital content platforms. Moreover, the scalability and adaptability of FAISS make it a preferred choice for researchers and organizations dealing with big data challenges. It has also been successfully applied in other tasks such as clustering, anomaly detection, and cross-modal retrieval, proving its robustness in both supervised and unsupervised learning environments. Compared to traditional search algorithms, FAISS achieves an optimal balance between search speed, memory consumption, and retrieval precision. Its open-source availability encourages further innovation, allowing continuous improvement and customization for specific research or industrial needs. Overall, this study establishes FAISS as a highly efficient and scalable solution for similarity search in modern data-driven applications. It bridges the gap between theoretical vector search algorithms and their practical implementation in large-scale systems. As the volume of

high-dimensional data continues to grow, FAISS will remain a foundational tool for data retrieval, recommendation systems, and machine learning applications. Future advancements may focus on improving FAISS's efficiency for streaming and distributed data environments, enabling even greater scalability. Hence, FAISS stands out as a powerful and indispensable framework that continues to advance the field of large-scale similarity search and artificial intelligence.

References

- [1]. Johnson, J., Douze, M., & Jégou, H. (2017). *Billion-scale similarity search with GPUs*. IEEE Transactions on Big Data, 7(3), 535–547.
- [2]. Guo, R., Sun, Y., Lindgren, E., Geng, Q., Simcha, D., & Kumar, S. (2020). *Accelerating large-scale inference with anisotropic vector quantization*. In Advances in Neural Information Processing Systems (NeurIPS).
- [3]. Chen, Y., Wang, L., & Liu, Z. (2020). *Image retrieval using deep features and FAISS indexing for large-scale datasets*. Journal of Visual Communication and Image Representation, 74, 102973.
- [4]. Singh, R., Sharma, P., & Mehta, A. (2021). *Semantic text similarity search using FAISS and BERT embeddings*. International Journal of Artificial Intelligence Research, 15(4), 255–266.
- [5]. Li, H., Zhang, W., & Zhao, J. (2023). *Comparative analysis of approximate nearest neighbor frameworks for multimedia retrieval*. IEEE Access, 11, 22541–22555.
- [6]. Malkov, Y. A., & Yashunin, D. A. (2020). *Efficient and robust approximate nearest neighbor search using HNSW*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4), 824–836.
- [7]. Douze, M., Johnson, J., & Jégou, H. (2018). *FAISS: A library for efficient similarity search and clustering of dense vectors*. Facebook AI Research Technical Report.
- [8]. Revaud, J., Douze, M., & Jégou, H. (2019). *Learning with deep embeddings for image retrieval and clustering*. Computer Vision and Image Understanding, 189, 102849.
- [9]. Huang, Z., Hu, S., & Chen, X. (2021). *High-performance nearest neighbor search with GPU-accelerated FAISS*. Journal of Parallel and Distributed Computing, 149, 101–112.
- [10]. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781.
- [11]. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. NAACL-HLT.
- [12]. Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). *ImageNet classification with deep convolutional neural networks*. In Advances in Neural Information Processing Systems (NeurIPS).
- [13]. Wang, J., & Jégou, H. (2018). *Billion-scale similarity search with GPUs: FAISS and beyond*. Facebook Research Blog.
- [14]. Han, J., Moraga, C., & Chen, D. (2020). *Optimized clustering and retrieval in FAISS using hierarchical quantization*. International Journal of Computer Applications, 182(35), 1–7.
- [15]. Zhou, Q., Lin, J., & Xu, T. (2021). *Integration of FAISS with deep learning frameworks for hybrid recommendation systems*. ACM Transactions on Information Systems, 39(3), 1–19.
- [16]. Park, S., Lee, J., & Choi, Y. (2022). *GPU-accelerated FAISS for real-time vector similarity search in large datasets*. Future Generation Computer Systems, 136, 251–262.
- [17]. Nguyen, D., Tran, V., & Pham, H. (2022). *Comparative evaluation of FAISS and Annoy for vector similarity tasks in AI pipelines*. International Journal of Data Science, 7(2), 111–123.
- [18]. Zhao, F., Liu, J., & Tang, Y. (2020). *Efficient similarity search in deep feature spaces using FAISS*. Journal of Intelligent Information Systems, 56(1), 85–98.
- [19]. Zhang, K., Gao, Y., & Liu, X. (2023). *Large-scale image clustering using FAISS and deep embeddings*. Multimedia Tools and Applications, 82(9), 13741–13759.
- [20]. Kumar, P., Sharma, D., & Gupta, A. (2024). *Applications of FAISS in scalable artificial intelligence systems: A comprehensive review*. International Journal of Advanced Computer Science and Applications, 15(2), 44–55
- [21]. Labhade-Kumar, N. (2023). *Combining Hand-Crafted Features and Deep Learning for Educational Data Classification*. Journal of Artificial Intelligence and Technology, Vol. 12, Issue 3, pp. 242–250.
- [22]. Labhade-Kumar, N. (2025). *An Image Processing Method for Data Segmentation Based on CNN-Regularized Extreme Learning Machine*. Hybrid and Advanced Technologies, Vol. 7, Issue 1, pp. 217–222.
- [23]. Labhade-Kumar, N. (2023). *Developing Interpretable Models and Techniques for Explainable AI in Decision-Making*. The Scientific Temper, Vol. 14, Issue 4, pp. 1324–1331.
- [24]. Neelam A Kumar Study of Different Sensors used in IoT, Indian Journal of Technical Education”, UGC Care Group I, ISSN 0971-3034 Vol47,Special Issue,PP- 136-140, April 2024
- [25]. Neelam Labhade-Kumar, Study on Object Detection Algorithm, Indian Journal of Technical Education UGC Care Group I, ISSN 0971-3034 Vol47,Special Issue,PP- 14-17, April 2024
- [26]. Neelam A Kumar Study of SHA-256 Hashing Algorithm, ALOCHANA JOURNAL VOLUME: 13, ISSUE: 12, ISSN NO:2231-6329, PP- 1172-1176, December 2024, UGC Care Group I
- [27]. Neelam A Kumar Detailed Study of Histogram Computation Algorithm in Image Processing, ALOCHANA JOURNAL VOLUME: 13, ISSUE: 12, ISSN NO:2231-6329, PP- 1071-1078, December 2024, UGC Care Group I