GeoCluster: A latent variable generative model for continuous space geometric clustering

Minas Dioletis[1,3], Ioannis Z. Emiris[2,5], George Ioannakis[3], Evanthia Papadopoulou[6], Thomas Pappas[3], Panagiotis Repouskos[3], Panagiotis Rigas[3,4,5], and Charalambos Tzamos[3,7]

1 **School of Electrical and Computer Engineering, National Technical University of Athens, Greece**
minasdioletis@mail.ntua.gr
2 **Athena Research Center, Greece**
emiris@athenarc.gr
3 **Institute for Language and Speech Processing, Athena Research Center, Greece**
{minas.dioletis,gioannak,thomas.pappas,
panagiotis.repouskos,panagiotis.rigas,charalambos.tzamos}@athenarc.gr
4 **Archimedes, Athena Research Center, Greece**
panagiotis.rigas@athenarc.gr
5 **Department of Informatics & Telecommunication, National & Kapodistrian University of Athens, Greece**
{emiris,rigasp}@di.uoa.gr
6 **Faculty of Informatics, Università della Svizzera italiana, Switzerland**
evanthia.papadopoulou@usi.ch
7 **Czech Technical University in Prague, Czech Republic**
tzamos.charalampos@fel.cvut.cz

Abstract

Given a set of shapes realized in $\mathbb{R}^d$, an important but challenging task is, given a query point $p \in \mathbb{R}^d$, to find the nearest shape to p w.r.t. a given distance function. Finding approximate or exact nearest neighbors is a fundamental algorithmic problem, which so far has predominantly focused on point-sets. In this work, given only a point-shape distance function, we tackle the problem of approximating the nearest neighbor of a query point to a set of shapes of unknown properties. We design a shape-agnostic algorithm for partitioning the set of shapes hierarchically, and build a tree data structure for answering nearest neighbor queries. For partitioning the space in k parts, we propose a machine learning algorithm, in which the shapes are treated as high dimensional vectors. We evaluate our proposed method on an extensive set of synthetic experiments.

1 Introduction

Nearest Neighbor Search constitutes a fundamental algorithmic problem that remains an active research field due to its importance in a variety of settings. The Nearest Neighbor Search problem is defined as follows. Let $\mathcal{O}_d$ denote a finite collection of *objects* (shapes) in $\Omega \subseteq \mathbb{R}^d$ that satisfy a property $\mathcal{P}$. This property indicates the type of data that we have, namely cubes, spheres, ellipses, convex polytopes with bounded number of vertices, etc. The distance between a point $p \in \mathbb{R}^d$ and an object $O \in \mathcal{O}_d$ is denoted by $\mathcal{D}(p, O)$, and its analytical form depends on the property $\mathcal{P}$. In this work, we are interested in approximating the nearest neighbor O^* of a query point p from a collection of *objects* $\mathcal{O}_d$, such that:

$$O^* = \arg\min_{O \in \mathcal{O}_d} \mathcal{D}(p, O). \tag{1}$$

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

We assume that d a is small constant and provide experimental results for d = 2 and d = 3.

In the simple case when the *objects* represent points in $\mathbb{R}^d$, the problem of approximating the nearest neighbor has been studied extensively [12, 3, 10, 11, 16, 5].

Point-based nearest neighbor queries are essential in data analysis for a variety of applications [2]. Searching for the nearest neighbor exhaustively is infeasible because the linear, on the number of candidates, time complexity tends to be expensive in any real-world setting [21]. Hence, one turns to approximate search where the goal is to retrieve one or a set of *Approximate Nearest Neighbors (ANNs)* in a fast and effective manner, namely that achieves sublinear, logarithmic, or even constant query time.

Considering more complex than points, *objects*, in a recent work [1], nearest neighbor queries against line segments in $\mathbb{R}^d$ has been considered, for a fixed dimension d, yielding an $(1+\varepsilon)$-approximate nearest neighbor algorithm. In [7], the authors address the problem of finding the nearest neighbor to a set of ellipses in $\mathbb{R}^2$ by computing the Voronoi diagram of a set of ellipses. In [6], orthogonal polyhedra in $\mathbb{R}^2$ and $\mathbb{R}^3$ are considered, and the problem of nearest neighbor is tackled by computing the Voronoi diagram of the set of polyhedra, using the L_∞ metric. In [18], the authors propose an algorithm for computing the L_∞ Voronoi diagram of a set of non-orthogonal shapes in $\mathbb{R}^2$. To the best of our knowledge, prior work on shape-based ANN search is limited to specific *objects* families and fixed dimension.

In this paper, we consider *objects* that can be determined by a finite set of parameters, e.g. a line segment in $\mathbb{R}^2$ can be uniquely described by its two endpoints, which can be considered as a vector of four parameters. For such shape-agnostic class of *objects* in $\mathbb{R}^d$, we designed an algorithm based on a generative machine learning model that answers efficiently *ANN* queries to a given point. To demonstrate the performance and applicability of our method to different types of *objects*, we provide experimental results. Code is publicly available [1], with details on how install and train the model, coupled with a demo.

2 Method

Our method employs a latent-variable generative model for hierarchical clustering, creating a tree structure where nodes represent data space regions, not just centroids, and child nodes represent sub-regions of their parent. Such a hierarchical clustering has been used in [16]. Hierarchical clustering refers to a tree T with branching factor k, that is built by recursively partitioning the data into sub-clusters until individual or constant number of data points are reached. This deviates from traditional fixed-cardinality clustering by adapting to the data manifold's shape. Critics assess and value the edges in this tree, converting it to a weighted graph for navigational decision-making by an actor with a policy[14, 15]. Explicit data embedding in a higher-dimensional space transforms complex shapes into discrete points, enabling divergence calculations for iterative state refinement. This approach, termed GeoCluster, dynamically clusters data, facilitating efficient approximate nearest neighbor queries.

2.1 Architecture

The proposed architecture is centered around a clustering mechanism, forming the foundation of a tree. This process begins by embedding objects into the high-dimensional space. The

[1] https://github.com/PRigas96/GeoCluster

clustering approach is recursive, resulting in a tree structure where each node is not merely a centroid, but represents a specific area of the data manifold $\mathcal{M}$.

To complement the clustering process a critic that evaluates and assigns values to the edges connecting the nodes in this tree is used. This evaluation results in a weighted tree, where the edge weights reflect the critic's assessment of the relationship between nodes, given a condition query p.

For inference, the tree T is utilized, or its corresponding quantified manifold $\mathcal{S}^{'}$ obtained from $\mathcal{M}$ after training. Given a query point $q \in \mathcal{Q}$, T is inferred by the actor through its policy π. This is based on weights on nodes, that act as decision-making tools, guiding the inference process. This approach mirrors a cost-minimization strategy, akin to finding the shortest path in the tree based on cross-entropy minimization [15].

Upon reaching a leaf node, an exhaustive search is conducted among the remaining objects to find the nearest neighbor O^*. The tree's functionality is illustrated in Fig. 1, where each node, endowed with a critic function, actively contributes to the inferential process by assigning scores to its children.

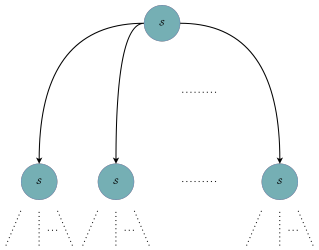

Figure 1 The tree constructed during training, and each node is associated with a critic.

2.1.1 Detailed Description

The architecture is presented at Fig. 2a, with the topology for the construction of a node in T. It consists of a latent variable $z \in \mathcal{Z}$ decoder module to produce the centroids $c \in \mathcal{S}$. A divergence Div then is calculated based on c and data $x \in \mathcal{M}$. Every f_{clk} Hz, a Gaussian sampler[4, 13, 20] is utilized and produces fuzzy centroids with radius $\zeta^{-1}Div$, scaled by a factor Sc, to alleviate the initialization problem caused by the many bad local minima, and initial centroids[17, 19]. This does not prevent clusters from being empty, which is utilized to further enable structure alignment by deleting non-participant centroids[8].

After states have been constructed, critic network C_S, a parametric function, is employed to assign low and high values to non probable and probable state transitions respectively, as shown in Fig. 2b. It is trained with data produced and labeled by an *uncertainty area sampler (UN)* module. The UN sampler consists of getting the Voronoi surfaces, sampling normal edges and points on those edges, conducting an efficient sampling in areas of uncertain prediction due to non-linear bounds between spaces.

The *clustering* network's latent variable is regularized through a one-hot encoder[9]. Representations c are regularized through $R_{rep} = \sum_i \sum_{j \neq i} d_p(c_i, c_j)^{-1}$, a repulsive loss that prevents trivial solutions and R_{proj} that enforces each representation to fall within its state.

To ensure this the states associated critic is utilized:

$$R_{proj} = \sum_{c \in \mathcal{S}} \sum_{x \in c} \sum_{u \in bbox} \text{ReLU}\left(\prod_{\forall e \in u} \frac{e - x}{|e - x|} \right) \min(||u - x||) + \sum_{S \in T} \omega_S \sum_{c \in S_\tau} \text{CE}(\text{Softmax}(C_s(c)), q_s), \quad (2)$$

where $ReLU(x) = \max(0, x)$ ensures initialization in root state, while cross entropy (CE), weighted by ω_S, that each a centroid in a path, satisfies all previous trajectory states, ensuring flow of information between different hierarchy level states. $Div = \sum_{x \in \mathcal{M}} E_w(x, \bar{z})$, where E_w is the states energy, is used to ensure clustering compactness[22].

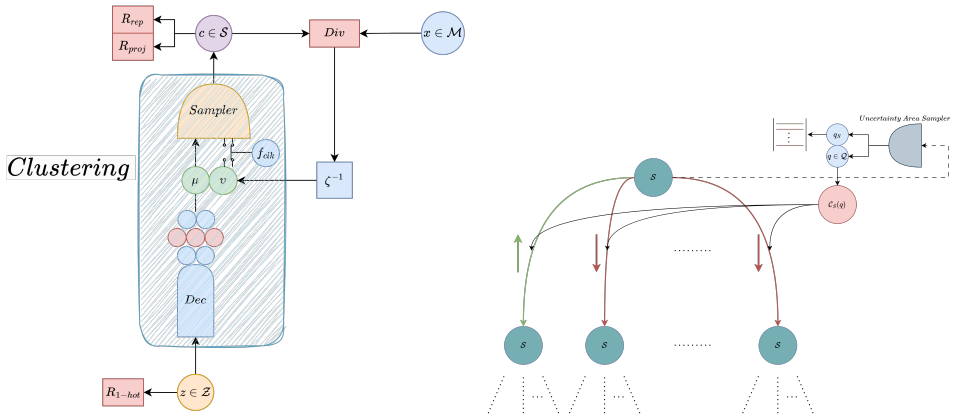

Figure 2 Proposed Network. *(a)* illustrates the network topology, for the construction of a node in the tree T. *(b)* showcases the critic network assigning weights in edges of a tree.

3 Experimental Results

In this section, we showcase the methods performance in three different aspects, the quality of produced representations and associated states, the search precision and the robustness.

3.1 Representation Evaluation

In Fig. 3 the centroids are shown for different structured data. It is clear that they capture the underlying structure effectively.

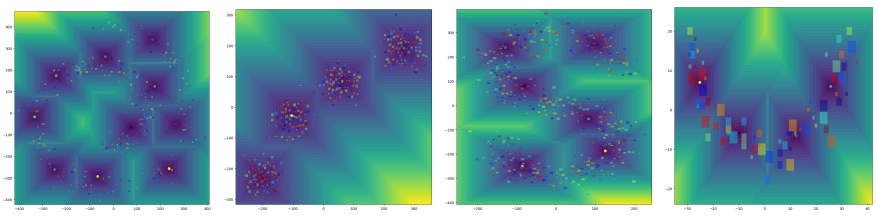

Figure 3 One-layer centroids, seen in dark blue areas for different structured 2D Data.

The resulted space for one layer of hierarchy is also shown for a 3D case in Fig. 4. Spaces can be non-linear and discontinuous.

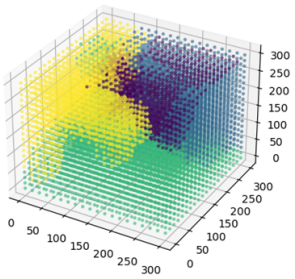

Figure 4 Quantified space $\mathcal{S}^{'}$ for 3D cuboids. Each color corresponds to a different cluster.

3.2 Search Precision and Parallelization

In this section, we delve into the model's search precision, specifically its efficacy in identifying complex geometric forms, as explicated in Tables 1 and 2. Table 1 provides insights the model's adeptness across various geometric shapes, demonstrating its commendable generalization capabilities. Notably, the model exhibits similar layer-wise accuracies, with final $acc = \prod_{i \in N} acc_i$, where N is the number of layers, being stable.

	Layer 1-2	Layer 2-3	Layer 3-4	Layer 4-5	Layer 5-6	Dimensions	Metric
Squares	82.8 ± 2.0	97.7 ± 0.7	99.8 ± 0.2	-	-	$\mathbb{R}^2$	L_∞
Cuboids	91.3 ± 0.8	87.9 ± 1.4	90.5 ± 2.4	93 ± 2.2	99.3 ± 0.7	$\mathbb{R}^3$	L_∞
Ellipses	95.7 ± 0.6	95.2 ± 1.1	97.7 ± 0.9	-	-	$\mathbb{R}^2$	L_2

Table 1 Table of Layer-wise Accuracies for Each Shape in the Dataset of 10,000 elements. The table displays mean accuracy and standard deviation for each shape. "Accuracy" is defined as the model's ability to precisely locate the nearest neighbor between layers, tested with randomly generated query points within the data's bounding area. Final accuracies can be obtained as a dot product of each layer.

The effectiveness of the model in accurately identifying and approximating the nearest neighbor is demonstrated in the results presented in Table 2. These results affirm the model's proficiency across various shapes, underscoring its strong generalization and scalability. Further, the data shown in Fig. 5 reinforces our assertion that the model efficiently captures the k-nearest neighbors, even when k is small, showcasing its robustness in neighbor detection.

	$k=1$	$k=2$	$k=5$	$k=10$	$k=1\%$	$k=5\%$	$k=10\%$	Layers
Squares	77.3 ± 2.3	79.1 ± 1.8	81.7 ± 2.4	83.4 ± 2.1	90.9 ± 2.0	99.7 ± 0.3	99.9 ± 0.1	5
Cuboids	61.9 ± 2.0	73.3 ± 1.6	85.3 ± 2.0	91.0 ± 1.2	99.0 ± 0.7	100	100	7
Ellipses	76.9 ± 1.6	79.3 ± 1.7	81.5 ± 1.8	83.8 ± 2.1	91.0 ± 0.9	98.4 ± 0.1	99.6 ± 0.4	5

Table 2 This table shows the accuracy of leaf nodes in the hierarchical structure for each shape in the dataset, with each shape having 10k instances. Here, "k" represents the number of top results considered for a successful search, indicating accurate neighbor detection among the first k results.

The results are also showcased for a soft-accuracy in non-randomly placed 2D data. The distance $\mathcal{D}(q, O^*)$, given a query q and its nearest neighbor O^* is measured. Then a relaxed criterion, that the nearest neighbor we find $\bar{O}$ is in a radius of O^*, is calculated. As shown

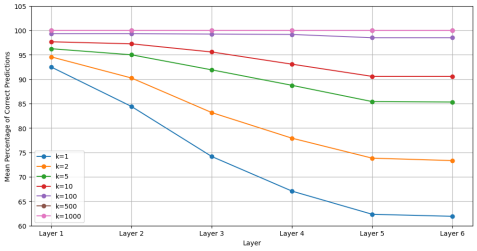

Figure 5 Cuboids: Mean Percentage of Correct Predictions per Layer for Different k-nearest neighbor Values.

in Fig. 6, a twofold increase in ball radius ration the accuracy converges, with a low mean bound of 95%.

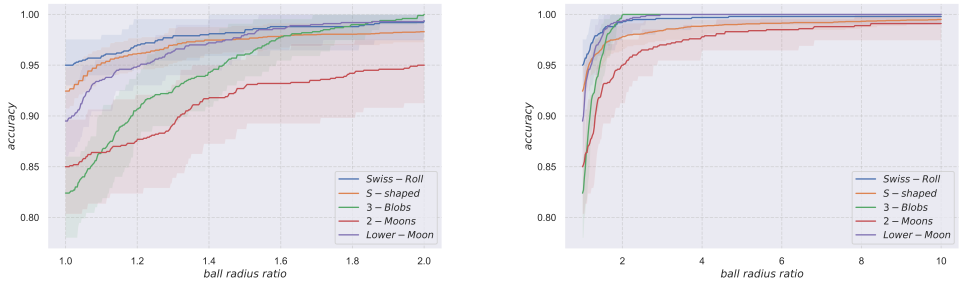

Figure 6 Soft-Accuracy tests different kind of structured 2D data, 1000 in number with 4 layers and width factor k=3. As ball radius ratio $\mathcal{D}(q, \bar{O})/\mathcal{D}(q, O^*)$, where $\bar{O}$ is the nearest neighbor found by our method and O^* is the true nearest neighbor, increases, accuracy converges.

Fig. 7 illustrates the model's superior performance in comparison to traditional serial search methods. Specifically, Fig. 7*a* highlights the model's adeptness at parallelization, a direct benefit of utilizing batching techniques. For a single node, as the quantity of query points escalates, our model consistently operates averagely at just $p = 1/100$th of the time required by linear search methods, denoted as $N \cdot \tau_1$. This efficiency underscores the stark contrast in performance scalability, particularly in handling large volumes of queries. Furthermore, Fig. 7*b* reveals that expanding the width factor k—which could potentially complicate the search process—does not detrimentally affect the model's performance. This observation confirms the model's robust scalability concerning data size, effectively demonstrating that its empirical average computational complexity remains practical and manageable, best captured by the expression $O(\lceil \frac{|Q|}{p} \rceil \log_k n)$, where Q is a set of queries, n the data objects and p an observed random value characterizing parallelization due to neural networks, shown in Fig. 7*a*. This notation adeptly reflects the combined influence of batching, hardware acceleration, and the model's algorithmic design on enhancing processing efficiency beyond simple linear expectations. Worst case complexity still remains $O(|Q| \log_k n)$, when no parallelization takes place.

3.3 Robustness

The robustness of our method is evaluated using various structured 2D data sets, as illustrated in Fig. 8. Although accuracy initially drops with high variance when the test set deviates from

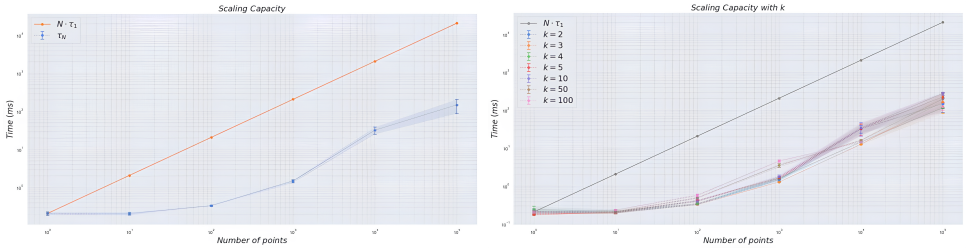

Figure 7 Performance (in seconds), in log scale, for a number of query points and one layer. τ_1 denotes a query pass and thus $N\tau_1$ can be regarded as $\mathcal{O}(n)$. *(a)* shows the difference between $\mathcal{O}(n)$ and our method, while *(b)* shows for different width factors (k).

the trained area, it quickly converges, demonstrating the network's effective extrapolation capabilities. Additionally, Table 1 further corroborates the method's robustness, highlighting its consistent performance across a diverse range of shapes and dimensions.

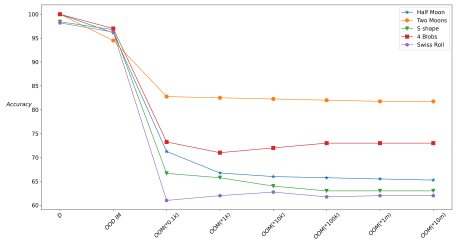

Figure 8 Robustness tests for different structured 2D data in one layer hierarchy. Eight columns show different areas, D for Data-trained area, *OOD IM* for data inside data manifold with trained area and *OOM*a* for outside manifold area scaled with a. Accuracy converges outside manifold rapidly but displays high variance for different structure

4 Conclusions

In this study, we leverage neural networks, specifically a latent variable generative model, to construct a tree. This is coupled with a critic mechanism to weight the edges, streamlining the nearest neighbor search across various domains and metrics. Addressing the challenge of point-to-shape nearest neighbor in 3D, even for simple shapes, our model stands out for its domain-agnostic nature. It requires only the definition of an embedding and a metric for divergence, demonstrating its versatility and effectiveness in simplifying complex geometric computations, while its inherent design makes it scalable.

5 Acknowledgment

This work has been partially co-financed by the European Union and the "Greece 2.0" national recovery and resilience plan, under the call "RESEARCH-CREATE-INNOVATE" (Code: TAEΔK-06168).

References

1 Ahmed Abdelkader and David M Mount. Approximate nearest-neighbor search for line segments. In *37th International Symposium on Computational Geometry*, 2021.

2 Yannis Avrithis, Yannis Kalantidis, Evangelos Anagnostopoulos, and Ioannis Z Emiris. Web-scale image clustering revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1502–1510, 2015.

3 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

4 Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

5 Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586, 2011.

6 Ioannis Z Emiris and Christina Katsamaki. Voronoi diagram of orthogonal polyhedra in two and three dimensions. In *International Symposium on Experimental Algorithms*, pages 1–20. Springer, 2019.

7 Ioannis Z Emiris, Elias P Tsigaridas, and George M Tzoumas. The predicates for the exact voronoi diagram of ellipses under the euclidiean metric. *International Journal of Computational Geometry & Applications*, 18(06):567–597, 2008.

8 Greg Hamerly and Charles Elkan. Learning the k in k-means. *Advances in neural information processing systems*, 16, 2003.

9 John T Hancock and Taghi M Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):1–41, 2020.

10 Sariel Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2001.

11 Sariel Har-Peled and Nirman Kumar. Approximating minimization diagrams and generalized proximity search. *SIAM Journal on Computing*, 44(4):944–974, 2015.

12 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

13 Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

14 Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

15 Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1), 2022.

16 Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2227–2240, 2014.

17 Shi Na, Liu Xumin, and Guan Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on intelligent information technology and security informatics*, pages 63–67. Ieee, 2010.

18 E. Papadopoulou and D. T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry and Applications*, 11(5):503–528, 2001. doi:10.1142/S0218195901000626.

19 Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

20 Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

21 Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - A survey. *Proc. IEEE*, 104(1):34–57, 2016. doi:10.1109/JPROC.2015.2487976.

22 Jyoti Yadav and Monika Sharma. A review of k-mean algorithm. *Int. J. Eng. Trends Technol*, 4(7):2972–2976, 2013.