Stratified Locality-Sensitive Hashing for Accelerated Physiological Time Series Retrieval

Yongwook Bryce Kim, Erik Hemberg, Una-May O'Reilly

Abstract— **We introduce stratified locality-sensitive hashing (SLSH) for retrieving similar physiological waveform time series. SLSH further accelerates the sublinear retrieval time obtained by the standard locality-sensitive hashing (LSH) method. The standard family of locality-sensitive hash functions is limited to provide only a single perspective on the data due to its one-to-one relationship to a distinct distance function for measuring similarity. SLSH incorporates multiple locality-sensitive hash families with various distance functions enabling it to examine the data with more diverse and refined perspectives. We provide the procedures of SLSH with locality-sensitive hash families for the l_1 and the cosine distances, and compare its performance to the standard LSH on an arterial blood pressure time series data extracted from the physiological waveform repository of the MIMIC2 database. The time to retrieve five most similar waveforms by SLSH is 14 times faster than the linear search and 1.7 times faster than the standard LSH when we allow 5% decrease in accuracy as a trade-off.**

I. Introduction

In data-driven medicine, efficiently leveraging massive datasets is a key challenge. We are particularly interested in the problem of quickly retrieving similar physiological waveforms. In the urgent care setting, promptness is crucial, so if a task can be completed dramatically faster, it is often acceptable to tolerate a slight decrease in accuracy. For a given query (such as a patient's record or a segment of the physiological waveforms of interest herein), its nearest neighbor (NN) retrieval returns a set of records that resemble the query. When records extend forward in time past that of the query, they may reveal outcomes of patients, chosen protocols, and diagnostics, which would be beneficial to predict acute, critical events of a patient.

Physiological data, in particular, have challenging properties because they are typically very high-dimensional and are subject to various structured and unstructured noise sources. In [1], we introduced a scalable retrieval system based on locality-sensitive hashing (LSH) for high-dimensional massive physiological data, which has a significantly faster querying time, while maintaining the retrieval quality in a competitive range in comparison to the linear, exhaustive k-nearest neighbor (KNN) search. LSH is a sublinear time, approximate search method [2] enabling a quick retrieval of a small approximate NN set. It is based on the idea of using a specialized hashing method to provide preliminary filtering of NN candidates to reduce the time cost of a follow-up linear search among them. LSH intrinsically introduces the trade-off between the level of approximation and speed.

When utilizing LSH, the appropriate choice of distance function for measuring similarity is critical because of the one-to-one relationship between the distance function and its unique corresponding family of locality-sensitive hash functions. Typically, a single locality-sensitive hash family offers only one perspective on the data with its associated distance function. For example, in Figure 1, when either the Manhattan l_1 or the Euclidean l_2 distance is used as a distance function, the similarity between waveform time series is mainly determined by the *amplitude* of the waveforms. On the other hand, when using the cosine distance, which has been widely used as the similarity measure for retrieval tasks in images [3] and speaker identification [4], the notion of similarity is based on the *shape* and the *angle* between waveform vectors as it requires data to lie on a unit sphere. It is also important to note that for many distance functions (the earth mover's distance, for instance), corresponding locality-sensitive hash functions do not always exist.

Interpreting clinical physiological waveforms, however, requires diverse perspectives on the data. Both the amplitude (e.g. mean blood pressure) and the shape of waveforms (e.g. trend and cycle frequency) contain important clinical information. For example, acute hypotensive episode is defined as a sudden dropping (shape) of blood pressure to below 60 *mmHg* (amplitude) for a prolonged period of time. In practice, users often do not precisely know which facets of similarity they are interested in. Thus, it would be highly beneficial to provide them a fast means of measuring similarity with multiple perspectives.

The current limit of LSH is that it can only hash the data with one family of locality-sensitive hash functions for a given distance function at a time. In order to explore the data with more diverse and refined perspectives, we introduce the new concept of Stratified LSH (SLSH), which incorporates hash families for multiple distance functions, where 1) the outer level LSH coarsely stratifies the data by amplitude using the l_1 distance, and 2) hierarchically, within each strata, the inner level LSH by the cosine distance more finely hashes the data according to angle and shape of time series.

In this work, we provide the procedures of SLSH and evaluate whether SLSH is advantageous and by how much, compared to the standard LSH and the linear KNN method. We present a guideline for setting the required parameters and demonstrate SLSH on a dataset of mean arterial blood pressure extracted from the intensive care unit (ICU) physiological waveform repository of the MIMIC-II database [5].

*Y.B. Kim, E. Hemberg, and U-M. O'Reilly are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. {ybkim, hembergerik, unamay}@csail.mit.edu

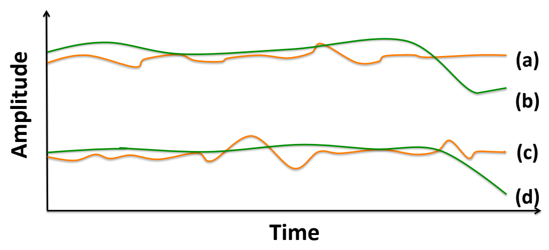

Fig. 1: An illustration of hypothetical time series with different amplitudes and shapes. By the l_1 distance, (a, b) and (c, d) are grouped as similar whereas by the cosine distance, which requires normalization, (a, c) and (b, d) are similar.

II. Related Work

In [6], Andoni *et al.* introduced a two-level hashing method which results in the best lower bound complexity out of the general LSH techniques. However, this work only provides theoretical analysis of its method, and lacks experimental evaluations and a practical implementation. To our best knowledge, SLSH is the first application of multi-level LSH for the retrieval task of physiological time series referencing a repository with thousands of patients.

III. Method: Locality-Sensitive Hashing

We use LSH as a means of accelerating retrieval speed through approximation of NNs. Whereas the KNN method exhaustively searches for the exact NNs, LSH aims to speed-up the search process by looking for approximate NNs instead. This approximation is tenably justifiable because even in exact search, the distance measure D is also only an approximation to the ground truth. For SLSH, we investigate the families for the l_1 and the cosine distances.

A. Locality-Sensitive Hash Family

The central idea of LSH is to hash data points by multiple hash functions with a special property that, for each hash function, the probability of hashing to the same hash value (i.e. collision) is much higher for points close in high-dimensional space than those that are far away from each other. To preserve this similarity-based locality, hash functions h are chosen from a hash family H that is (R, cR, P_1, P_2)*-sensitive*, i.e., for any points $p, q \in \mathbb{R}^d$,

- if $\|p - q\| \leq R$, then $\Pr_H[h(p) = h(q)] \geq P_1$
- if $\|p - q\| \geq cR$, then $\Pr_H[h(p) = h(q)] \leq P_2$

for constants $c, R > 0$, and $P_2 < P_1$. The gap between P_1 and P_2 should ideally be wide to increase the probability of collision. This is done by applying multiple hash functions.

B. Bit-Sampling based LSH for the l_1 Distance (L1LSH)

In this study, we utilize the bit sampling based hash family for l_1 distance (L1LSH), proposed in [7], $H_{L1} = \{h : \mathbb{X}^d \to \{0, 1\}\}$ such that

$$h(p) = \begin{cases} 0 & \text{if} \quad p_i < t_i \\ 1 & \text{if} \quad p_i \geq t_i \end{cases}$$

where p_i is the value on the i^{th} coordinate of $p \in \mathbb{X}^d$. i is a single dimension of the data chosen uniformly at random from $\{1, \dots, d\}$ and t_i is a threshold chosen uniformly from the range of p_i. We choose this LSH family because it is parameter-free and requires no tuning unlike other hash families for l_1 [8].

C. Random Projection based LSH for the Cosine Distance (COSLSH)

For $p, q \in \mathbb{X}^d$, the angle between them is $\theta(p, q) = \arccos(\frac{p \cdot q}{\|p\|\|q\|})$. Charikar *et al.* [9] defines the locality-hash family for the cosine distance, $H_{cos} = \{h : \mathbb{X}^d \to \{0, 1\}\}$ such that

$$h_r(p) = \begin{cases} 0 & \text{if} \quad p \cdot r < 0 \\ 1 & \text{if} \quad p \cdot r \geq 0 \end{cases}$$

where $r \in \mathbb{R}^d$ is constructed by picking each coordinate of r from the isotropic Gaussian distribution $N(0, 1)$. This function is equivalent to dividing the space into two sub-spaces by a randomly chosen hyperplane, and the hash value is determined by the side on which p lands. The cosine distance in LSH has a strict requirement that the data must lie on a unit sphere. Although data normalization is a safe preprocessing step to make in many application areas, when using COSLSH alone, it is not suitable on physiological time series since the amplitude information gets completely lost.

D. Standard LSH Procedures

The single level standard LSH [1] consists of two steps: 1) constructing an efficient data structure (hash table) to index (hash) the data for fast retrieval to follow, and 2) quickly retrieving the approximate nearest neighbors of the query of interest. There are two LSH parameters, the number of tables constructed L and the number of hash functions m used per table, which need to be chosen empirically. The optimal pair of (m, L) provides the most efficient data structure to index the data for the fastest retrieval. The standard procedure of the single level LSH is as follows (for a graphical illustration and more details on the advantages of LSH for physiological time series, refer to [1]).

Standard Construction:

1) For $l = [1, 2, \dots, L]$, choose m functions such that $g_l = (h_{1,l}, h_{2,l}, \dots, h_{m,l})$. For each l, hash functions $h_{j,l}$'s are randomly chosen from a LSH family H.
2) Construct L hash tables where each hash table T_l contains the dataset points hashed using the function g_l. The value of g_l for each data point defines its hash key.
3) Store all T_l's and g_l's applied. This step incurs a trivial memory cost because we only store the pointers to the data instead of the data itself.

Standard Retrieval: For a query point $q \in \mathbb{R}^d$,

1) Find the points from the colliding bucket $g_l(q)$ in the l^{th} hash table for each $l = [1, 2, \dots, L]$, where g_l's are the same set of hash functions used for construction. These points define the "candidate set."
2) Compute the distance from q to each of the points in the candidate set. This linear search is the major bottleneck of the algorithm, which we would like to minimize.

3) *Retrieve* the closest k NNs to the query q from the candidate set by distance.

When setting up the standard LSH, the desirable number of hash functions used per table is the number that would partition the original data as uniformly as possible over *all possible* hash buckets so that each contains only a small amount of the data. In such way, when finding a hash bucket that collides with a query, the space subject to the linear search is enormously reduced, resulting in a large speed-up gain. In practice, however, tables typically contain a few populous buckets which impose a large bottleneck. This highlights another benefit of SLSH besides allowing multiple perspectives on the data by adding another layer of LSH to yield more finely partitioned, evenly populated hash buckets avoiding the bottleneck.

E. Stratified LSH Procedures

The procedures of our multi-level SLSH are composed of the following three tasks built on top of the standard LSH. We first stratify the data according to L1LSH at the outer level. Then, on each bucket with a significant size ("populous bucket"), we probe deeper with COSLSH at the inner level. This is somewhat analogous to performing top-down hierarchical clustering. We finally retrieve the approximate nearest neighbors of a query of interest.

Outer Level LSH with L1LSH:

1) Apply the standard construction above with $m = m_{out}, L = L_{out}$ and $H = H_{L1}$. Choose m_{out} such that the median size of hash buckets is $\alpha\%$ of the original data size. Now, the data are divided into hash buckets which work as strata in each hash table.
2) Store all $g_{l,out}$.
3) For each outer hash table T_l, apply the inner level COSLSH only on *each hash bucket* whose size is larger than $\beta\%$ of the original data. β defines populous buckets $B_{l,i}$, where i is the bucket index.

Inner Level LSH with COSLSH:

1) For each table T_l, apply the standard construction on each $B_{l,i}$ with $H = H_{cos}$ and $m = m_{in}, L = L_{in}$.
2) Store all $g_{l,in}$ applied on each $B_{l,i}$ and the generated inner level hash tables $T_{l,i}$.

Querying: For a query point $q \in \mathbb{R}^d$,

1) Apply the step 1 of the standard retrieval on q with the hash functions $g_{l,out}$ from the outer L1LSH. If it does not collide with a populous bucket, this step returns the candidate set from the outer level excluding the points from the populous buckets.
2) For $g_{l,out}(q)$ that collides with any populous bucket $B_{l,i}$, apply the step 1 of the standard retrieval recursively on each populous bucket with the $g_{l,in}$ applied previously. This returns the inner level candidate set.
3) Aggregate the candidate sets from both the outer and inner level to compose the final candidate set, and apply the steps 2 and 3 of the standard retrieval to retrieve the query's k NNs.

F. Remarks

While we use L1LSH and COSLSH for the outer and inner level LSH, respectively, our SLSH framework can embrace any distance function which has a valid locality-sensitive hash family. For example, it is possible to reverse the order of L1LSH and COSLSH. However, when COSLSH is used first at the outer level, it requires the entire data to be normalized prior to hashing and still needs to keep the original unnormalized data to perform L1LSH at the inner level. For both time and memory costs, it is more inefficient than normalizing only subsets of data that belong to the populous buckets in our proposed procedure. In addition, first stratifying the data according to shape has a lower interpretability to characterize each strata when compared to first stratifying the data by amplitude.

IV. Experiment and Discussion

We evaluate whether SLSH is advantageous and by how much, compared to the single level standard LSH with L1LSH. We present our findings on our time series dataset of mean arterial blood pressure (MAP) extracted from the large ICU physiological waveform repository of MIMIC-II (Multi-parameter Intelligent Monitoring in Intensive Care) database [5]. We select patients who have segments of 300 (d) minutes of contiguous signal, which results in 6,467 segments from 2,291 patients. For the purpose of the retrieval task, this dataset both serves as the reference set from which neighbors are retrieved, and as the set of queries.

It is complicated to evaluate nearest neighbor retrieval accuracy because there is no ground truth, i.e., no single notion of similarity is perfect, each being dependent on some distance metric. As a practical baseline for comparison, we use the linear KNN search with the l_1 distance metric.

Given a pair of LSH parameters (*m*, *L*), for each query, we find *k*-approximate NNs via a choice of LSHs and compare the NN set to those from the KNN search. We define the retrieval accuracy as the fraction of the *k*-nearest neighbors that are retrieved by both methods. Accordingly, the overall retrieval accuracy is the average of the individual accuracies over all queries. We also investigate the retrieval time, where we define the speed-up factor of LSH as the retrieval time of a query by LSH relative to that by the KNN search. In order to evaluate SLSH, we compare its relative accuracy and speed-up gain to those of the standard single level LSH using L1LSH.

For each LSH type, we define the "optimal" values of (*m*, *L*) as the parameters which result in the fastest average retrieval time among the ones with average retrieval accuracies higher than 95% (i.e. maximum 5% accuracy trade-off tolerance). For the standard L1LSH, we vary *m* from 5 to 50 with an increment of 5 and *L* from 10 to 70 with an increment of 10. For the inner level SLSH by COSLSH, we use m_{in} from 1 to 11 with an increment of 2 and L_{in} from 1 to 16 with an increment of 3. For the outer level SLSH by L1LSH, we arbitrarily set $\alpha = 5\%$, $\beta = 5\%$, and $L_{out} = 10$ from our prior experience. We apply the above procedure to retrieve $k = 5$ NNs for demonstration.

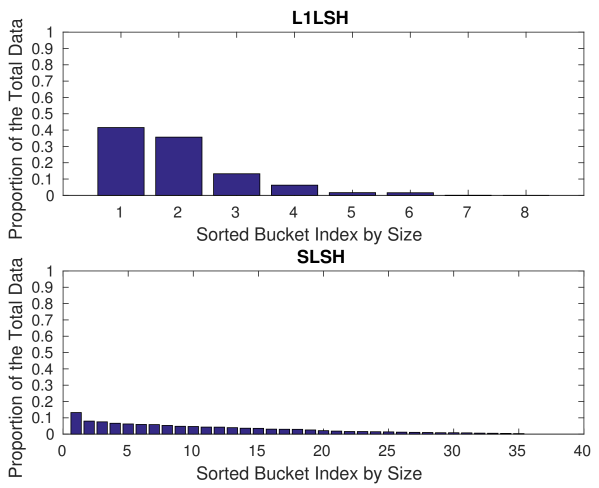

Fig. 2: (*Top*) The empirical distribution of the average bucket size over $L_{out} = 10$ tables by the outer L1LSH with $m_{out} = 3$, sorted from the largest to the smallest. The largest 3 populous buckets which account for more than 5% of the total data each are subject to the inner level SLSH for more fine-grained partitioning. (*Bottom*) The empirical distribution of the average bucket size over $L_{out} = 10$ tables by SLSH with $m_{out} = 3$, $m_{in} = 3$, and $L_{in} = 10$, sorted in descending order. SLSH results in a larger number of and more evenly distributed buckets than L1LSH.

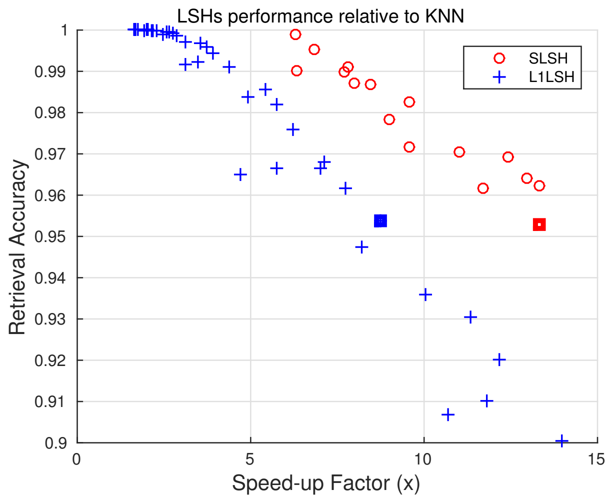

Fig. 3: Trade-off between retrieval accuracy and speed-up factor of the LSH performance relative to the exhaustive KNN for SLSH (red) and L1LSH (blue). Each point corresponds to a single parameter configuration of LSHs. Squared points indicate the optimal parameters for the maximum 5% trade-off in accuracy for SLSH and L1LSH.

A. Hash Table Bucket Distribution

We first find the appropriate m_{out} for the outer level SLSH. A good choice of m_{out} would partition the original data into multiple coarsely defined, interpretable strata by the amplitude, on which the inner level LSH can generate even more refined hash buckets. We gradually increase m_{out} from 1 up to a point where the median hash bucket size becomes equal to or less than $\alpha = 5\%$ of the total data size. For our dataset size of 6,467, $m_{out} = 3$ generates, on average over $L_{out} = 10$ tables, 8 buckets with the median bucket size of 312 (4.82% of the total dataset size) and the average size of 2604. As shown in Figure 2 (*Top*), the large difference between the median and average size implies that there exist more buckets which are rarely or under populated (long tail) besides a few large "populous buckets" which need to be more finely partitioned. This is a primary reason why we do not apply the inner level COSLSH on small buckets since these buckets are already well-defined for a fast retrieval. In our case, on average, 3.25 buckets out of 8 (41%) had a size greater than $\beta = 5\%$ of the total data and were subject to the inner level LSH. The reason why we observe this non-uniform distribution is because our data already has some structure in it. This is reasonable because the small strata come from the buckets with high and low mean amplitudes, corresponding to the cohorts of patients with a very high and low blood pressure, respectively.

Figure 2 (*Bottom*) shows the empirical distribution of the average bucket size over $L_{out} = 10$ tables by SLSH with $m_{out} = 3$, $m_{in} = 3$, and $L_{in} = 10$, sorted from the largest to the smallest. Only 14% of the total buckets (5 out of 35) account for more than 5% of the total data, and data are more evenly distributed over all buckets. As a result of adding the inner layer COSLSH, we obtain a set of many near uniformly distributed small buckets with SLSH, which satisfies the highly desirable property for a fast retrieval discussed in section III-D.

B. Trade-Off between Accuracy and Speed

Figure 3 shows the trade-off between the retrieval quality and time for the standard L1LSH and SLSH. Each point represents the average retrieval accuracy and the speed-up factor of each method over 10 trials for each pair of (*m*, *L*). The standard deviations were too small to be displayed on the figure. The square boxes indicate the optimal results for each type of LSH. For 5-NN retrieval by L1LSH with (*m*, *L*) = (30, 50), we achieve the optimal nearest neighbor retrieval 8.5 times faster than the linear KNN method when we allow up to 5% decrease in retrieval accuracy as a trade-off. On the other hand, SLSH with $(m_{in}, L_{in}) = (3, 7)$ and $(m_{out}, L_{out}) = (3, 10)$ achieves the optimal retrieval 14 times faster than KNN, again allowing less than 5% decrease in accuracy. Thus, SLSH is 1.74 times faster than L1LSH alone. The inner level SLSH requires fewer hash functions and tables since there were only an average of 3.25 "populous buckets" in each table with sizes much smaller than the total size of the data. We achieve a large speed up gain with SLSH by avoiding the bottleneck because the additional time taken to apply hash functions at the inner level is much smaller than the time required to conduct the linear search in populous buckets. All in all, the experiment shows that with a carefully chosen set of parameters, SLSH has a much faster retrieval speed than the standard LSH with L1LSH alone.

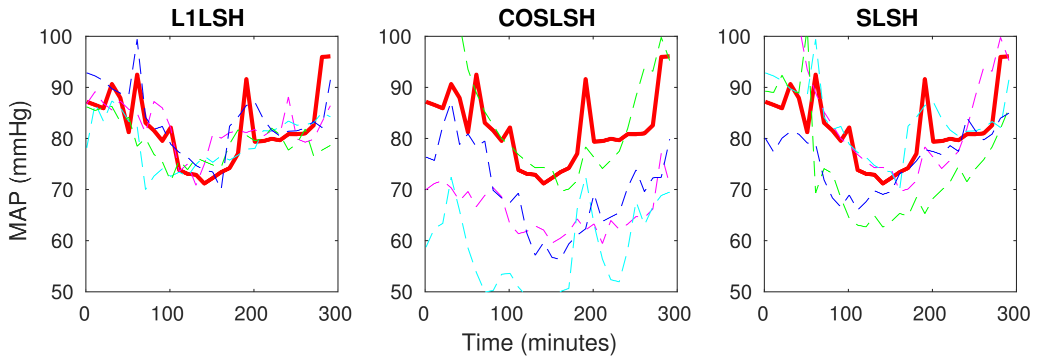

Fig. 4: The nearest neighbors (colored dashed lines) of a MAP waveform query (red line) retrieved by (*Left*) the standard L1LSH, (*Middle*) the standard COSLSH, and (*Right*) SLSH. Each set is similar to the query in terms of amplitude, shape, and *both* amplitude *and* shape by L1LSH, COSLSH, and SLSH, respectively.

C. Retrieved Nearest Neighbor Set

Figure 4 illustrates the retrieved nearest neighbor sets of a MAP time series query using the standard single-level L1LSH, the standard single-level COSLSH, and SLSH. For the given query (Segment ID 6404, red line), in Figure 4 (*Left*), we observe that the set of NNs (colored dashed lines) retrieved by L1LSH is tightly located close to the query in terms of the mean amplitude but oblivious to their various shapes. Likewise, in Figure 4 (*Middle*), the NNs retrieved by COSLSH all resemble the shape of the query, but their amplitudes are well-spread across various levels. Figure 4 (*Right*) shows a qualitative evaluation of SLSH. The NNs obtained via SLSH not only have shapes that are similar to that of the query, but also have their mean amplitudes much closer the query compared to the set retrieved by COSLSH. Satisfying the notion of similarity in terms of both amplitude *and* shape, this qualitatively verifies that SLSH is able to address the data from multiple and more refined perspectives.

V. Future Work

Since the parameters for the outer level SLSH and the choice of distance functions were set arbitrarily, we plan to design a more systematic way of finding the optimal SLSH setup and conducting extensive sensitivity analysis of those parameters with various distance functions. We are particularly interested in investigating the impact of changing the order of hash families, and in examining how predictive accuracy changes due to stratification. In general, the larger a dataset is, the more dramatic the effect of LSH will be due to its sublinear time complexity. Thus, we plan to apply SLSH on much larger datasets with a higher resolution. We also aim to provide a descriptive visualization framework of our SLSH method for supporting clinical examination of matched time series to predict acute, critical events in the ICU setting.

Acknowledgment

This work was supported by the generous fundings from BigData@CSAIL and Kwanjeong Educational Foundation. The authors would like to thank Ludwig Schmidt and Piotr Indyk for helpful discussions and their feedback.

References

[1] Y. B. Kim and U.-M. O'Reilly, "Large-scale physiological waveform retrieval via locality-sensitive hashing," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pp. 5829–5833, IEEE, 2015.

[2] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of Computing*, pp. 604–613, ACM, 1998.

[3] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2130–2137, IEEE, 2009.

[4] L. Schmidt, M. Sharifi, and I. L. Moreno, "Large-scale speaker identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 1650–1654, IEEE, 2014.

[5] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark, "Multiparameter intelligent monitoring in intensive care II (MIMIC-II): A public-access intensive care unit database," *Critical Care Medicine*, vol. 39, no. 5, p. 952, 2011.

[6] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn, "Beyond locality-sensitive hashing," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1018–1028, SIAM, 2014.

[7] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pp. 518–529, Morgan Kaufmann Publishers Inc., 1999.

[8] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pp. 459–468, IEEE, 2006.

[9] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of Computing*, pp. 380–388, ACM, 2002.