Large-Scale Physiological Waveform Retrieval via Locality-Sensitive Hashing

Yongwook Bryce Kim and Una-May O'Reilly

Abstract— **We propose a fast, scalable locality-sensitive hashing method for the problem of retrieving similar physiological waveform time series. When compared to the naive *k*-nearest neighbor search, the method vastly speeds up the retrieval time of similar physiological waveforms without sacrificing significant accuracy. Our result shows that we can achieve 95% retrieval accuracy or better with up to an order of magnitude of speed-up. The extra time required in advance to create the optimal data structure is recovered when query quantity equals 15% of the repository, while the method incurs a trivial additional memory cost. We demonstrate the effectiveness of this method on an arterial blood pressure time series dataset extracted from the ICU physiological waveform repository of the MIMIC-II database.**

I. Introduction

The amount of data collected in the medical community has recently been exploding and is becoming more overwhelming due to widespread use of affordable sensors and storage devices. The ever increasing volume and detail of information captured from hospitals and personal healthcare devices direct the medical practice toward more data-driven, evidence-based medicine. Efficiently leveraging these massive datasets is a key challenge that, when resolved, will support a new paradigm of scientific discovery and operational innovation of medicine.

Information retrieval plays a crucial role permitting medical practitioners and researchers to acquire high-quality relevant information from a massive repository of medical records. One of the core problems in information retrieval is the nearest neighbor (NN) search. For a given query (such as a patient's record, a list of symptoms, or a piece of the physiological waveforms of interest herein), NN retrieval returns a set of records that are similar to the query. The NN set offers extrapolative information. It may reveal a complex pattern. When records extend forward in time past that of the query, they reveal outcomes of patients, chosen protocols, and diagnostics. Questions such as, for people with the same symptoms, what critical events subsequently occurred, what treatment produced the best recovery, what side-effects were observed, can be answered.

The main axes of challenge in retrieving such information are time and scale. Fast processing of physiological waveform data is becoming essential in medical practice, especially in intensive care units (ICU). Plus, the size of medical record corpora that we extract information from is vast and keeps increasing even at this moment. There is a huge need for large-scale, yet accurate, data processing in almost real time.

Although the naive NN search method [1] for retrieval works very well in practice with moderately sized data, its performance deteriorates rapidly for large, high-dimensional data. Our goal is to build a scalable retrieval system for high-dimensional massive physiological data, with a significantly faster querying time, while maintaining the retrieval quality in a reasonable range. This is justifiable because we expect to be retrieving fairly large quantities of similar information from which we aim to develop just-in-time predictive models. In order to achieve this goal, we propose a retrieval method based on locality-sensitive hashing (LSH) [2], which allows a very fast approximate nearest neighbor search in high dimensions. LSH is based on the simple idea of using a specialized hashing method to provide rapid, preliminary filtering of NN candidates and reduce the time cost of a follow-up linear search among them. LSH has the unique property that similar elements are statistically likely to be hashed into the same value.

In this work, we evaluate whether LSH is advantageous, and by how much, on a dataset of mean arterial blood pressure (ABP) extracted from the large ICU physiological waveform repository of the MIMIC-II database [3]. In comparison to the naive *k*-nearest neighbor search, our result indicates that our LSH based method can achieve 95% retrieval accuracy or better with up to an order of magnitude faster querying time. There is an extra time cost to create the optimal LSH data structure in advance, but it is recovered when query quantity equals 15% of the repository while the additionally incurred memory cost is trivial. To our best knowledge, this work is the first application of locality-sensitive hashing based system for the retrieval task of physiological time series at this detail referencing a repository with thousands of patients.

II. Related Work

There has been a series of work which showed the effectiveness of simple NN methods over many popular parametric models for time series classification. For instance, [4] showed that the one-nearest-neighbor classifier with the dynamic time warping distance measure has a superior performance over multi-layer perceptron neural network, hidden Markov model, and decision tree. In [5], the authors showed that their nearest-neighbor based classifier performs better than support vector machine, naive Bayes, and C4.5 decision tree for classifying patients with abnormal hearts from a small electrocardiography (ECG) dataset.

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

LSH has been successfully applied in retrieval tasks in many applications such as images [6], speaker identification [7], and music search [8]. LSH was also used in indexing ECG time series using salient segmentation of data, but the data, containing only very short segments, was not large enough to show significant advantages [9].

III. Problem Statement

We define the nearest neighbor search problem as follows: Given a set P of n d-dimensional points $\{x_1, \ldots, x_n\} \subset \mathbb{X}^d$ and a distance measure $D : \mathbb{X}^d \times \mathbb{X}^d \to \mathbb{R}^+$, for a query $q \in \mathbb{X}^d$, find its nearest neighbor $x_i = \arg\min_i D(x_i, q)$ from P. For a given query, our goal is to accurately retrieve its k nearest neighbors in the shortest amount of time possible. We compare the retrieval accuracy and querying time of the LSH based method to the naive linear NN search for various k and LSH parameter settings, and assess the trade-off between accuracy and speed-up factor.

IV. Methods

A. k-Nearest Neighbor Method: Linear Search

k-nearest neighbor (KNN) method is a simple nonparametric instance-based learning algorithm. In essence, it memorizes the entire reference set and finds a group of k samples that are closest to a query by exhaustively going through every point in the reference set and computing its distance to the query. Typically, one uses the NN set to extrapolate a class label or response variable for the query based on predominance.

Unlike parametric models such as dynamic Bayesian network, KNN lets the data speak for itself because it does not summarize by a number of parameter values of a certain model or make any assumption on the underlying state and distribution. Since it is very difficult to assume the underlying mechanism of human physiological signals, NN methods can be advantageous.

In spite of its simplicity and practicality, there are some pitfalls. First, KNN is heavily influenced by the choice of distance metric and neighbor weighting rules. Plus, it becomes highly impractical when the dimensionality of data is high or as the quantity of data grows, which occurs with physiological waveforms, since distance measures break down in high dimensions [10], and repeatedly computing distance calculations on every sample in the reference set for each query is extremely expensive. It was also empirically demonstrated that the performance of the KNN method is highly sensitive to data conditions such as size and class balance [11].

B. Locality-Sensitive Hashing

In order to search through a vast amount of physiological data, we propose to apply locality-sensitive hashing for efficient waveform retrieval. Whereas the naive NN method searches for the exact NNs, LSH aims to speed-up the search process by looking for approximate nearest neighbors instead. Approximate neighbors are valuable because even in the exact search, the distance measure D is also only an approximation to the ground truth.

The central idea of LSH is to hash the data points by multiple hash functions where, for each hash function, the probability of hashing to the same hash value (collision) is much higher for points close in high-dimensional space than those that are far away from each other. To preserve this 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 be ideally wide to increase the probability of collision. This is done by concatenating multiple hash functions.

In our study, we utilize the hash family for l_1 distance, proposed in [12], $H = \{h : \mathbb{X}^d \to \{0, 1\}\}$ such that

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

where p_i is the value on the ith coordinate of $p \in \mathbb{X}^d$. i is a single dimension of the data chosen uniformly at random from $\{1, \ldots, d\}$ and t is a threshold chosen uniformly from the range of the data in that dimension. We choose this family of hash functions because it is parameter-free and requires no tuning unlike other more sophisticated hash families for l_1 [13]. Plus, this family is equivalent to the thoroughly studied bit sampling based hash family for the Hamming distance. d-dimensional P can be embedded into the Hamming cube of dimension $d' = wd$ by applying a unary function on each coordinate in P, where w is the largest coordinate of all points in P. It is a known fact that the Hamming distance on this embedded space preserves the l_1 distance in the original space. For the detailed implementation procedures, analysis, and theoretical justification, one should refer to [12].

LSH Construction LSH has a fixed cost construction phase that supports subsequent retrievals. We construct L hash tables each using m independently designed hash functions over the reference set. Each hash function $h_{j,l}$ is independently, uniformly chosen at random from a locality-sensitive family H for $j = 1, \ldots, m$ and $l = 1, \ldots, L$. For each table T_l, the results of the m hashing functions are concatenated as $g_l(p) = (h_{1,l}(p), \ldots, h_{m,l}(p))$ for every point p in the reference set and become a bucket identifier. The overall procedure is illustrated in Figure 1. When hashed by g, for two points p and q to belong to the same hash bucket, their hash values have to match for every one of m distinct hash functions h. Therefore, the larger/smaller the quantity m of hash functions, the stricter/more-approximate the match. This parameter m is what we will explore experimentally to derive an optimal trade-off in accuracy and time savings. It is important to note that LSH tables only store the pointers to the data instead of the original data itself. By applying standard hashing to store only non-empty buckets, it creates only a trivial additional memory cost of only $O(nL)$.

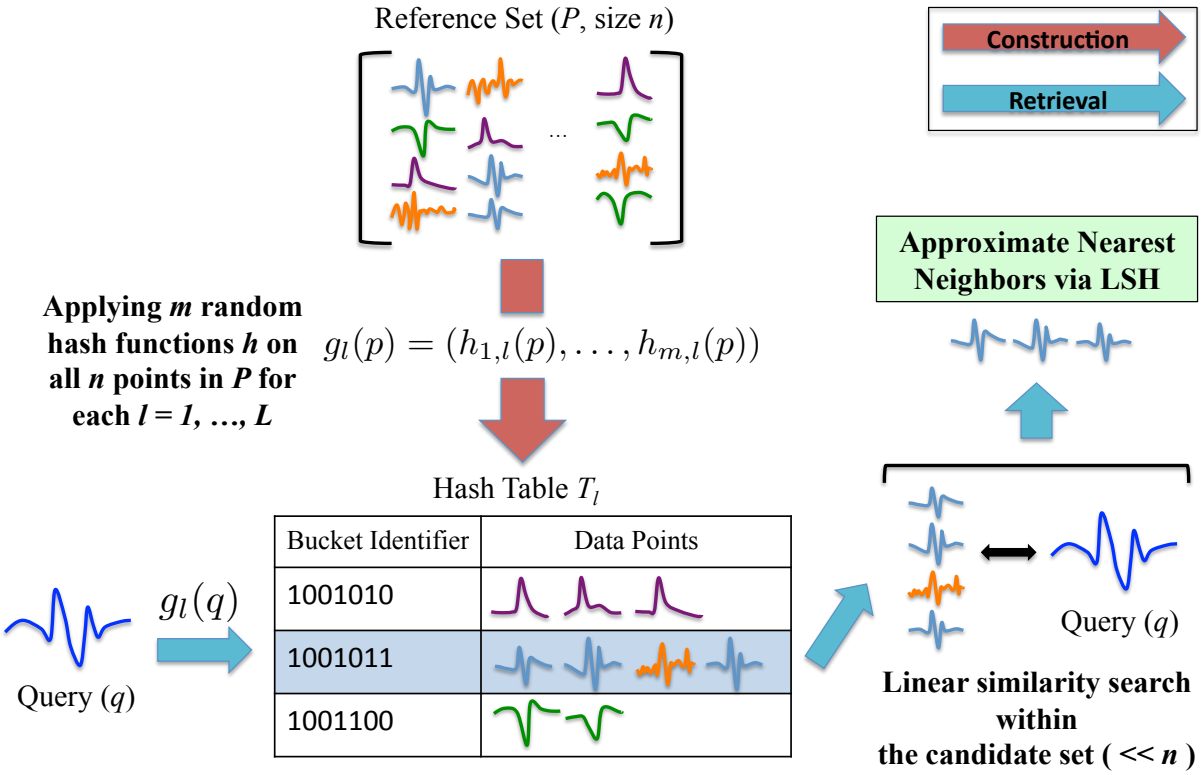

Fig. 1. The overview of LSH construction (red arrow) and retrieval (blue arrow) procedures for a single hash table. For retrieval, a query is first hashed to find the points included in its matching bucket in the hash table. Then, we linearly compute the distance between the query and such set of points (the candidate set with a size significantly smaller than that of the reference set) to retrieve the approximate nearest neighbors of the query. For multiple tables, the final candidate set for the linear similarity search consists of the points included in all matching buckets from all L hash tables.

LSH Retrieval To find the approximate nearest neighbors of a query point q, we first hash the query and retrieve points stored in the matching bucket $g_l(q)$ in each hash table T_l. Then, we aggregate the points included in the matching buckets from all tables. This defines the candidate set, on which we linearly compute the distance from q to each point in the candidate set to find the k approximate nearest neighbors of q (Figure 1). A query is hashed for each table, so L, the quantity of tables, is an additional LSH parameter. As the quantity of tables goes up, more approximate matches, each from different tables, are possible which increases the chance of finding the exact nearest neighbors of the query. In some cases, the size of the candidate set is smaller than k and the query fails to retrieve k NNs. It is important to note that the search time of LSH is guaranteed to be sublinear to the size of the data [12], compared to the linear time cost of the naive NN method. This can make a huge difference in search time in a massive data repository.

Advantages By transforming the high-dimensional data into the space of short hash values generated by m hash functions, we effectively achieve dimensionality reduction to a lower order embedding space where the notion of similarity is well preserved. Moreover, while KNN suffers from the selection bias of the distance metric, LSH inherently offers a broader, less-biased coverage over the non-linear characteristics of waveform signals because each hash function h serves as a different basis of comparing points with a low resolution, "weak" similarity. Furthermore, both LSH construction and retrieval processes are distributable since each table is generated and queried independently. Also, it is scalable because for new data, only the small step of applying the kept hash functions g is needed, which does not require a recomputation over the entire data.

V. Experiment

A. Data

Our data comes from the MIMIC (Multi-parameter Intelligent Monitoring in Intensive Care) II version 3 database which contains physiological waveform time series signals and clinical notes collected from the ICU units at the Beth Israel Deaconess Medical Center in Boston, MA [3]. It is one of the largest clinical medical databases that are currently publicly available. In this work, we select the ABP waveforms because our goal is to eventually forecast acute conditions such as sepsis and hypotension based on blood pressure. Out of the 25,328 patients in the MIMIC-II database, 6,232 patients have their associated ABP waveform records. Since the waveforms were collected for many hours at 125 Hz for each patient, the entire ABP data comprises 240,000 plus hours of records requiring about 2 terabytes of data in its compressed form.

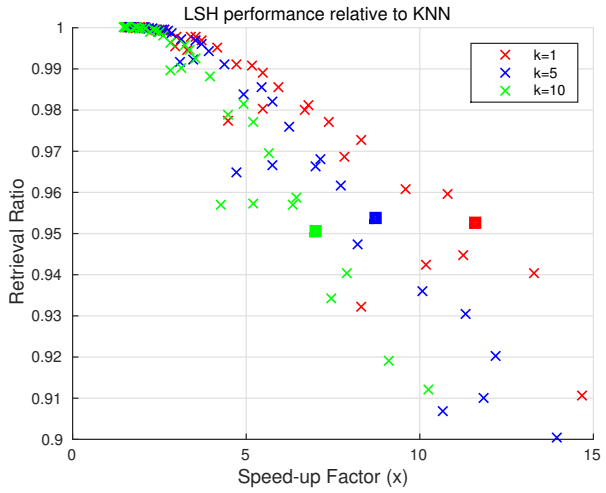

Fig. 2. Trade-off between retrieval accuracy and speed-up factor of the LSH performance relative to the naive KNN. Each point corresponds to a parameter configuration of LSH. Squared points indicate the optimal parameters for different values of k.

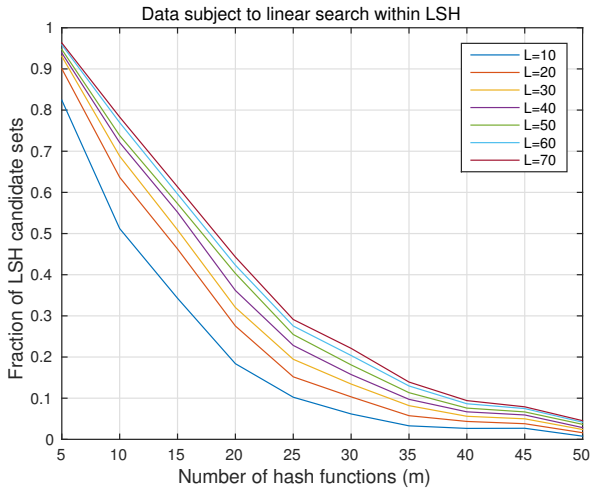

Fig. 3. Fraction of the data subject to the linear search within LSH (candidate set) for different values of LSH parameters.

For each patient record, we apply an onset detection algorithm on the ABP waveform to find the beginning and end of each beat, examine its validity, and calculate its corresponding mean arterial pressure (MAP) value, which is approximated by a weighted average of systolic and diastolic pressure within a cardiac cycle [14]. Then, we transform the time series of per-beat MAP values to the time series of per-minute average of MAP values for each patient. We select patients who have segments of 300 (d) minutes of contiguous signal, which results in 6,467 segments (each with a unique ID) 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. We use the range of [49.1, 109.5] *mmHG* for the randomly picked threshold t of each hash function. This corresponds to the values two standard deviations away from the mean on each side.

B. Nearest Neighbor Retrieval

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. Practically, we proceed by using the naive linear method with the l_1 norm distance metric as our baseline. It returns the result of an exhaustive search, and LSH will be compared to this "most accurate" (though most costly) method in terms of retrieval accuracy and time. Using the same distance metric in the final linear search step of LSH unifies the comparison basis.

Given a pair of LSH parameters (m, L), for each query, we find k-approximate NNs via LSH and compare them to those from the naive linear search. We define the retrieval accuracy as the fraction of the k-nearest neighbors that are retrieved by both methods (sharing the identical segment ID). We do not consider the order within the k-NNs, but care only about the intersection. 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 linear method.

We define the optimal values of (m, L) as the parameters which result the fastest average retrieval time among the ones with average retrieval accuracies higher than 95%. We vary m from 5 to 50 with an increment of 5 and L from 10 to 70 with an increment of 10. We apply the above procedure to retrieve 1-NN, 5-NNs, and 10-NNs.

C. Results

Figure 2 shows the trade-off between the retrieval quality and time. Each point represents the average retrieval accuracy and the speed-up factor of LSH over 10 trials for each pair of (m, L). The square boxes indicate the optimal parameters for each neighborhood size, k. For 1-NN, with (m, L) = (30, 40), we achieve the optimal nearest neighbor retrieval 12 times faster with LSH than the linear search, while sacrificing less than 5% accuracy. For 5-NN and 10-NN, we find the optimal LSH parameters are (30, 50) and (30, 60), respectively, and we still get reasonable retrieval times 8.5 and 7 times faster than the naive exhaustive search. We observe that the optimal number of hash functions (m = 30) remains the same while only the optimal number of hash tables (L) increases for a higher number of k-NN retrieval. This implies that we can choose m and then expand along the L axis as we need bigger and bigger similarity sets.

For fixed m and L, only the retrieval accuracies degrade with an increasing k while the retrieval times remain almost the same. This is an expected behavior since it is much harder for the farthest neighbors among the k-NNs from both methods to match for a large k, while retrieving more neighbors adds only a negligible time cost.

While all queries successfully return their 1-NNs, the fraction of queries that are not able to retrieve the requested number of NNs (where the size of the candidate set of a

query was smaller than k) was merely 0.11% and 0.19% of the entire data for 5-NN and 10-NN, respectively.

While LSH is faster and close to perfect accuracy, it does incur two preliminary costs: time to hash the data m times over L tables and storing hash tables. We investigated the break-even point of this cost with respect to query time saved relative to the linear search time. Given the target accuracy and the speed-up, the time to construct the optimal LSH data structure was approximately equivalent to the retrieval time of 15% of the total queries. So after processing 15% of the queries, we would have saved enough time to make up for the cost of building the optimal LSH data structure. The extra storage cost of hash tables was less than 1% of that of the original data since the hash tables only contain pointers to the original data.

Figure 3 presents the fraction of the data which are candidates for the linear search within LSH. For our range of optimal LSH parameters (30 hash functions per table), we observe that only 20% or less of the entire data is subject to the linear search within LSH. This is useful because even with many hash tables, the data subject to the most time costly step of LSH retrieval is a small fraction of the entire data especially when a large number of hash functions are used to build the tables.

VI. Conclusion

In this work, we have proposed a fast, scalable locality-sensitive hashing method for the retrieval problem of similar physiological waveform time series. We demonstrated this method on our arterial blood pressure time series repository extracted from the massive MIMIC-II database. When compared to the naive k-nearest neighbor search method, we can achieve 95% retrieval accuracy or better with up to 12 times of speed-up. The extra time required in advance to create the optimal data structure was recovered when query quantity was equal to 15% of the repository, while the method incurred a trivial memory cost. With this efficient retrieval model, we plan to extend our work to improve the system for processing a variety of complex query types with a diverse set of hash families, and to the forecasting and prediction problem of acute, critical events in ICUs.

ACKNOWLEDGMENT

This work was supported by Li Ka Shing Foundation and Kwanjeong Educational Foundation. The authors would like to thank for their financial support. The authors also thank Franck Dernoncourt and Kalyan Veeramachaneni for initial data preparation, as well as Ludwig Schmidt and Piotr Indyk for helpful discussions.

References

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[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] 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.

[4] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 1033–1040, ACM, 2006.

[5] B. Hu and E. Keogh, "Time series classification under more realistic assumptions," in *SIAM Conference on Data Mining (SDM)*, 2013.

[6] 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.

[7] 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.

[8] M. Ryynanen and A. Klapuri, "Query by humming of midi and audio using locality sensitive hashing," in *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, pp. 2249–2252, IEEE, 2008.

[9] J. Woodbridge, B. Mortazavi, A. A. Bui, and M. Sarrafzadeh, "High performance biomedical time series indexes using salient segmentation," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 5086–5089, IEEE, 2012.

[10] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," in *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pp. 420–434, Springer-Verlag, 2001.

[11] Y. B. Kim, J. Seo, and U.-M. O'Reilly, "Large-scale methodological comparison of acute hypotensive episode forecasting using mimic2 physiological waveforms," in *Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on*, pp. 319–324, IEEE, 2014.

[12] 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.

[13] 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.

[14] A. Waldin, K. Veeramachaneni, and U.-M. O'Reilly, "Learning blood pressure behavior from large physiological waveform repositories," in *ICML Workshop on Role of Machine Learning in Transforming Healthcare*, 2013.