# An Examination of Multivariate Time Series Hashing with Applications to Health Care

David C. Kale,*† Dian Gong,* Zhengping Che,*
Yan Liu, Gerard Medioni
Computer Science Department
University of Southern California
Los Angeles, CA 90089
{dkale,diangong,zche}@usc.edu
{yanliu.cs,medioni}@usc.edu

Randall Wetzel, Patrick Ross
Laura P. and Leland K. Whittier Virtual Pediatric Intensive Care Unit
Children's Hospital Los Angeles
Los Angeles, CA 90027
{rwetzel,pross}@chla.usc.edu

***Abstract*—As large-scale multivariate time series data become increasingly common in application domains, such as health care and traffic analysis, researchers are challenged to build efficient tools to analyze it and provide useful insights. Similarity search, as a basic operator for many machine learning and data mining algorithms, has been extensively studied before, leading to several efficient solutions. However, similarity search for multivariate time series data is intrinsically challenging because (1) there is no conclusive agreement on what is a good similarity metric for multivariate time series data and (2) calculating similarity scores between two time series is often computationally expensive. In this paper, we address this problem by applying a generalized hashing framework, namely kernelized locality sensitive hashing, to accelerate time series similarity search with a series of representative similarity metrics. Experiment results on three large-scale clinical data sets demonstrate the effectiveness of the proposed approach.**

## I. Introduction

Multivariate time series data are becoming ubiquitous *and* big. Nowhere is this trend more obvious than in health care, with the growing adoption of electronic health records (EHRs) systems. According to a 2009 survey, hospital *intensive care units* (ICUs) in the United States (US) treated nearly 55,000 patients per day,[1] generating digital health databases containing millions of individual measurements, many of which constitute multivariate time series. Clinicians naturally want to utilize these data in new and innovative ways to aid in the diagnosis and treatment of new patients. An increasingly popular idea is to search these databases to find "patients like mine," i.e., past cases that are similar to the present one [1].

This classic data mining task, known as *similarity search*, must be both accurate and fast, which depends crucially on two choices: representation and similarity measure. For traditional data types (e.g., structured data, free text, images, etc.), the standard approach is to define a set of features that we extract from each object in our database and then apply straightforward measures of similarity (e.g., Euclidean distance) to these. This has been applied to time series data [2], but designing good features can be difficult and time-consuming.

Researchers have shown empirically that the best representation for time series is often the data themselves, combined with specialized similarity measures [3]. The classic example is *dynamic time warping* (DTW), an extension of Euclidean distance that permits nonlinear warping along the temporal axis in order to find the optimal alignment between two time series [4]. The choice in time series similarity measures ranges from simple to complex approaches based on fitting parametric models [5]. Indeed, there has been an explosion in the number and variety of time series similarity and distance metrics proposed in the literature over the last decade [6] [7] [8] [9].

There are two critical things to observe about these competing similarity metrics, especially if we want to use them to implement fast similarity search for multivariate time series: that *different similarities work best for different data and problems*; and that *the most effective similarity measures are often computationally expensive and ill-suited to large scale search*. The first point was best demonstrated by the thorough empirical evaluation in [9], in which no one metric worked best for all data and problems. Choosing the right similarity (much like designing good features) requires experience, intuition, and experimentation. The second point is more nuanced; some approaches can often be sped up by using a combination of good engineering and heuristics (e.g., DTW [10]). However, these speed-ups do not generalize beyond specific tasks or to other measures that we may want to use.

In this paper, we investigate a general solution that applies to a large class of time series similarities: *kernelized hashing*. Hashing has been used to build fast search and retrieval over massive databases of text and images [11] [12] [13]. It utilizes one or more hash functions to map the input data to a fixed-length representation (typically a binary code), which then can be used as indexes in a large-scale storage architecture. Well-constructed hash functions will assign similar codes to similar objects, allowing us to store them together and to find them with just a quick lookup.

* These authors contributed equally.
† Also affiliated with the VPICU and CHLA.
[1]From the *American Hospital Association Hospital Statistics* survey conducted in 2009 and published in 2011 by Health Forum, LLC, and the American Hospital Association.

We apply *kernelized locality-sensitive hashing* (KLSH) [14] to the problem of fast search over multivariate time series. KLSH kernelizes the LSH search framework so that it can be used with arbitrary time series similarity metrics. In this paper we combine it with a modified *Euclidean distance*, *multivariate dynamic time warping* [10], *global alignment kernels* [15], and *vector autoregressive kernels* [8]. We show that it provides significant speed-ups versus exhaustive search across all similarity metrics without significantly compromising the quality of the search results.

We present empirical results that back up our argument, using data sets that include two large, anonymized databases of medical time series from Children's Hospital Los Angeles (CHLA): the 10,000 patient PICU data set [5] and a collection of vital signs time series from over 60,000 surgeries. We find, first of all, that our intuition is confirmed: there is no single best similarity measure across all data sets. Second, we show that KLSH speeds up similarity search for *all* similarity metrics with an acceptable impact on search result quality. Thus, kernelized hashing provides a much more robust and flexible approach to speeding up large scale time series similarity search than attempting to optimize any one distance measure. We believe that this work is but a first step toward a comprehensive approach to fast similarity search for time series (and other structured objects) and that it opens the door to innovative work on time series hashing.

## II. Background

### A. Notation

We define a univariate time series of length $T$ as a set of $T$ samples from a random process parameterized by time (and indexed by $t$): $\boldsymbol{x} = \{x(1) \dots x(t) \dots x(T)\}$. In this paper, we assume that time is discrete and that measurements are regularly sampled with no missing values. We will denote a single univariate time series as $\boldsymbol{x} \in \mathcal{R}^T$. A *multivariate time series* (MVT) is a set of $P$ time series (one per each of $P$ variables) sampled at the same time intervals:

$$\boldsymbol{X} = \left\{ \begin{bmatrix} x_1(1) \\ x_2(1) \\ \vdots \\ x_P(1) \end{bmatrix} \cdots \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_P(t) \end{bmatrix} \cdots \begin{bmatrix} x_1(T) \\ x_2(T) \\ \vdots \\ x_P(T) \end{bmatrix} \right\}$$

We can represent a MVT as a $P$-by-$T$ matrix: $\boldsymbol{X} \in \mathcal{R}^{P \times T}$, where the $i$th row is the $i$th univariate time series $\boldsymbol{x}_i \in \mathcal{R}^T$ and column $t$ is a $P$-vector of observations at time $t$: $\boldsymbol{x}(t) \in \mathcal{R}^P$. We refer to the dimensionality of this MVT as $P$ and consistently use a subscript plus lowercase bold ($\boldsymbol{x}_i$) to indicate the $i$th dimensional time series in a single MVT $\boldsymbol{X}$. Throughout this paper, we will often refer to $P$-dimension MVTs simply as "time series" when our meaning is clear from context. Suppose now that we have a database (which we will refer to as $\mathcal{D}$) of $P$-dimension MVTs, $N$ in total: $\mathcal{D} = \{\boldsymbol{X}_j : \boldsymbol{X}_j \in \mathcal{R}^{P \times T_j}\}_{j=1}^{N}$. We will consistently use a subscript plus uppercase bold ($\boldsymbol{X}_j$) to index MVTs in a database. All of the time series in our database are of the same dimension $P$ but may vary in length.

### B. Problem Definition

We are interested in the *multivariate time series similarity search* problem: given a query time series $\boldsymbol{X}^q \in \mathcal{R}^{P \times T_q}$, we are interested in finding the time series $\boldsymbol{X}^* \in \mathcal{D}$ that is most similar to $\boldsymbol{X}^q$, given an arbitrary definition of *similarity* between MVTs $S(\boldsymbol{X}, \boldsymbol{X}^q)$:

$$\boldsymbol{X}^* = \underset{\boldsymbol{X} \in \mathcal{D}}{\operatorname{argmax}}\ S(\boldsymbol{X}^q, \boldsymbol{X})$$

i.e., the time series $\boldsymbol{X}^* \in \mathcal{D}$ that maximizes $S(\boldsymbol{X}^q, \cdot)$. Alternatively, if we are given a *distance* measure $D$ instead of similarity, we seek to minimize $D(\boldsymbol{X}^q, \cdot)$. More generally we are interested in the $k$ *nearest neighbors (knn)* problem: finding the $K$ most similar time series to $\boldsymbol{X}^q$. Furthermore, we are interested in performing this search as quickly as possible.

### C. Related Work

Fast search in a database of variable length MVTs presents two interrelated challenges that thwart most standard approaches:

1) Designing a time series similarity (or distance) measure that is both effective and fast can be difficult.
2) Many traditional approximate search methods cannot easily be applied to variable length MVTs.

Computing the similarity between variable length time series can be quite computationally expensive and so an approach based on an exhaustive search (i.e., *linear scan*) of the database will not scale to really large data sets (i.e., some combination of large $P$, large $T$, and large $N$). This problem is not unique to variable length time series. In computer vision, for example, researchers have found that the best representations for images tend to be extremely high dimensional feature vectors, which can be expensive in terms of both memory and computation [12]. With such data, linear scan searches using Euclidean distance become prohibitively slow, necessitating the development of alternative approaches.

One possible solution is to use a fast approximate search for nearest neighbors, perhaps using an alternative time series representation. A framework that has enjoyed widespread success and popularity is *symbolic approximate aggregation* (SAX), which discretizes the continuous observation space and replaces real values with symbols [6]. By converting the data to a discrete representation, we gain access to fast algorithms for mining sequences of symbols (e.g., longest common subsequence). Subsequent work has shown that SAX does a good job of approximating the true similarity between short time series and can be used to perform scalable indexing of large time series databases [16]. How to apply SAX to multivariate time series remains an open research question.

Another alternative representation that enables both efficient storage *and* fast search is binary codes, such as those produced by *hashing*. Locality sensitive hashing (LSH) is one of the most widely used approaches to constructing fixed length binary representations of data for fast approximate nearest neighbor search; it is based on the intuition that the probability that two objects share the same code should be proportional to

their similarity [11]. LSH has received numerous extensions, including the incorporation of supervision through the use of metric learning [17]. There is also a wide variety of alternative methods for learning binary embeddings of data which can then be used for hashing and fast comparison. These include semantic hashing [18]; spectral hashing [19]; shift-invariant kernel hashing [20]; directly learning a hamming distance metric [21]; etc. All of these approaches share a common limitation: they require fixed size data representations, which prohibits their direct application to time series data.

Two major advances in the last five years should enable us to perform fast approximate nearest neighbor search over MVTs via hashing. The first is *kernelized locality sensitive hashing* (KLSH) [14]. KLSH permits the hashing of objects for which similarity is defined using arbitrary kernel functions. The second major advance has been a large number of proposed time series similarity and distance measures. Many of these can naturally be formulated into valid kernels and used within the KLSH framework, allowing us to combine the power of these similarity measures with the speed of hashing based search.

## III. Kernelized Hashing of Time Series

We begin with a thorough review of kernel-based hashing and of similarity kernels for time series. We then describe briefly how to put these together into a unified kernelized time series hashing framework. We then rapidly move on to our empirical investigation of the efficacy of this approach.

### A. Kernelized locality sensitive hashing

*Locality sensitive hashing* (LSH) enables fast similarity search by generating bit vector representations of objects, which can be compared very rapidly on modern hardware [11]. Given an object $\boldsymbol{X}$, we can generate a $B$-bit *hash code* for the object using a collection of $B$ hash functions $\{h_1, \ldots, h_B\}$. For each hash function, the probability that two objects receive the same bit (1 or 0) is proportional to their similarity: $P\{h_i(\boldsymbol{X}) = h_i(\boldsymbol{X}')\} \propto S(\boldsymbol{X}, \boldsymbol{X}')$ for all $i = 1, \ldots, B$. A common way to achieve this is by defining $h_i(\boldsymbol{X}) = 1$ if $\boldsymbol{a}^\top \boldsymbol{X} \geq 0$ and 0 otherwise, where $\boldsymbol{a}$ is a random hyperplane sampled from a zero mean, unit variance Gaussian: $\boldsymbol{a} \sim \mathcal{N}(0, \boldsymbol{I})$ [22]. Standard LSH assumes fixed size feature vectors and uses cosine similarity. It cannot be used directly with more specialized notions of similarity or objects with variable sizes or special structure (e.g., time series).

Kernel methods allow us to apply machine learning to problems where the data does not live in a fixed dimensional space or where we want to use similarity measures that incorporate specialized information or structure. Assume we have a positive definite *kernel function* $\kappa$ that produces a cosine-like similarity between $\boldsymbol{X}$ and $\boldsymbol{X}'$: $\kappa(\boldsymbol{X}, \boldsymbol{X}')$. Here $\kappa$ *induces* a feature space $\phi$ for which we cannot directly examine $\phi(\boldsymbol{X})$ or calculate, for example, a Euclidean distance between two examples ($\|\phi(\boldsymbol{X}) - \phi(\boldsymbol{X}')\|_2$). On the other hand, we can compute their similarity using $\kappa$ since $\kappa(\boldsymbol{X}, \boldsymbol{X}') = \phi(\boldsymbol{X})^\top \phi(\boldsymbol{X}')$.

We can use this kernel function to perform kernelized LSH in much the same way that we use kernel methods for support vector machines [14]. We compute the random hyperplane $\boldsymbol{a}$ as a weighted sum over $M \ll N$ training examples in the kernel-induced feature space: $\boldsymbol{a} = \sum_{j=1}^{M} \boldsymbol{w}_j \phi(\boldsymbol{X}_j)$. Again, we cannot explicitly compute this, but we observe that we *can* evaluate $\boldsymbol{a}^\top \phi(\boldsymbol{X}')$ for some new example $\phi(\boldsymbol{X}')$:

$$\begin{aligned} \boldsymbol{a}^\top \phi(\boldsymbol{X}') &= \left( \sum_{j=1}^{M} \boldsymbol{w}_j \phi(\boldsymbol{X}_j) \right) \phi(\boldsymbol{X}') \\ &= \sum_{j=1}^{M} \boldsymbol{w}_j \phi(\boldsymbol{X}_j) \phi(\boldsymbol{X}') \\ &= \sum_{j=1}^{M} \boldsymbol{w}_j \kappa(\boldsymbol{X}_j, \boldsymbol{X}') \end{aligned}$$

and so our KLSH hashing function becomes

$$h_{\boldsymbol{a}}(\boldsymbol{X}') = \begin{cases} 1 & \text{if sign} \left( \sum_{j=1}^{M} \boldsymbol{w}_j \kappa(\boldsymbol{X}_j, \boldsymbol{X}') \right) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Two questions remain: how big should $M$ be (and how do we choose our $M$ points?); and how do we choose our weights $\boldsymbol{w}_j$ such that $\boldsymbol{a}$ is a "valid" hyperplane for performing LSH (i.e., is $\boldsymbol{a} \sim \mathcal{N}(0, \boldsymbol{I})$)? Clearly if $M$ is very large (i.e., $M \approx N$), then evaluating hash function becomes almost as expensive as performing a linear scan using our kernel-based similarity. Thus, we want $M \ll N$. [14] suggests sampling $M = O(\sqrt{N})$ points at random, which guarantees sublinear runtime and usually produces good accuracy. For the second question, we choose $\boldsymbol{w} = \boldsymbol{K}^{-1/2} \boldsymbol{e}_R$. $\boldsymbol{K} \in \mathcal{R}^{M \times M}$ is our similarity kernel matrix over our $M$ samples (i.e., $\boldsymbol{K}_{jl} = \kappa(\boldsymbol{X}_j, \boldsymbol{X}_l)$). $\boldsymbol{e}_R \in \{0, 1\}^M$ is an indicator vector with only $R < M$ nonzero entries, corresponding to a subset of $R$ points chosen at random from the $M$ points included in $\boldsymbol{K}$. [14] gives a derivation showing that, by the central limit theorem, $\boldsymbol{a}$ will be approximately distributed according to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_P)$.

The main computational expense of KLSH is the initial training, which requires computing and the inverting the $M$-by-$M$ kernel matrix $K$ (an $O(M^3)$ operation) to find $\boldsymbol{w}$. Searching for the nearest neighbors of a new query $\boldsymbol{X}^q$ is very fast. First, we evaluate our kernel function $O(BM)$ times to compute $B$ bits. By choosing $M \ll N$, we ensure that this is much faster than performing an exhaustive search, which requires $O(N)$ kernel evaluations. In the second step, we search over binary hash codes to find nearest neighbors, which is trivially fast, even in large data sets. Finally, we can optionally refine our set of nearest neighbors by evaluating the true similarity over a small set of candidates [13]. The result is a significant speed-up over an exhaustive search using the ground truth similarity metric.

### B. Time series similarity

Time series similarity measures can roughly be classified into two groups: those that capture shape and those that

capture higher level or global structure [9]. The most popular shape-based approach is dynamic time warping (DTW) [4]. It has been shown to be very effective in a wide variety of settings and applications. It is computationally expensive ($O(T^2)$ for comparing two time series each of length $T$), but researchers have shown a variety of ways to speed it up [23][24][25][10]. A more significant weakness is the fact that it does not obey the triangle inequality and so does not constitute a true distance metric. This prevents its use in kernel methods without modification [26]. A generalization of DTW called the Global Alignment (GA) kernel, which computes the similarity between two time series based on all possible alignments, has been shown to produce positive definite kernels and is often competitive with DTW [15].

Structure-based similarity measures attempt to capture higher level structure in the time series. [27] first convert univariate time series to a symbolic representation and then construct histograms over a "vocabulary" of symbol subsequences. They call this a *bag-of-patterns* representation. Also included in this general category are model-based approaches, which compare time series using an assumed model. [7] describe a kernel based on *echo state networks* that first trains a recurrent reservoir model and then compares two time series in the model space [7]. [8] propose an autoregressive time series kernel (ARK) based on linear vector autoregressive models that eliminates the two-step "train, then compare" process by showing it can be formulated as a covariance kernel.

Here we describe four representative similarity measures that can be combined with kernelized hashing to yield fast similarity search over multivariate time series. However, our framework applies to any time series similarity or distance.

**Multivariate dynamic time warping.** *Dynamic time warping* was first proposed nearly fifty years ago and remains one of the gold standards for comparing time series. Suppose we have a pair of time series $\boldsymbol{X} \in \mathcal{R}^{P \times T}$ and $\boldsymbol{X}' \in \mathcal{R}^{P \times T'}$ and a measure of discrepancy $d(\boldsymbol{x}(t_1), \boldsymbol{x}'(t_2))$ between a pair of time points $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}'(t_2)$, one from each time series, where $t_1$ and $t_2$ need not be equal. For example, we might use the squared Euclidean distance between these points:

$$d(\boldsymbol{x}(t_1), \boldsymbol{x}'(t_2)) = \|\boldsymbol{x}(t_1) - \boldsymbol{x}'(t_2)\|_2^2 = \sum_{i=1}^{P} \left(x_i(t_1) - x_i'(t_2)\right)^2$$

Next we define an alignment (or *warping function*) $\boldsymbol{\lambda} = \{(1,1), \ldots, (t_j, t_j'), \ldots, (T, T')\}$ as a list of non-decreasing pairs of indices of $\boldsymbol{X}$, $\boldsymbol{X}'$ [15]. $\boldsymbol{\lambda}$ satisfies these properties:

- $|\boldsymbol{\lambda}| \leq T + T' - 1$.
- $\lambda(j+1) \geq \lambda(j)$ for all $j = 1, \ldots, |\boldsymbol{\lambda}|$.
- $(1,1) \leq \lambda(j) \leq (T, T')$ for all $j = 1, \ldots, |\boldsymbol{\lambda}|$.
- $\lambda(j+1) - \lambda(j) \in \{(0,1), (1,0), (1,1)\}$ for all $j = 1, \ldots, |\boldsymbol{\lambda}|$.

The $j$th pair of indices aligns the points $\boldsymbol{x}(\lambda(j,1))$ and $\boldsymbol{x}(\lambda(j,2))$. Every point must be aligned, and we only allow forward movements along $\boldsymbol{X}$ or $\boldsymbol{X}'$. We then define the $\boldsymbol{\lambda}$-alignment distance between $\boldsymbol{X}$ and $\boldsymbol{X}'$ as

$$D_{\boldsymbol{\lambda}}(\boldsymbol{X}, \boldsymbol{X}') = \sum_{j=1}^{|\lambda|} d(\boldsymbol{x}(\lambda(j,1)), \boldsymbol{x}'(\lambda(j,2))$$

We then define the *multivariate dynamic time warping* (MDTW) distance as the minimum $\boldsymbol{\lambda}$-alignment distance over a set of possible alignments given by $\Lambda(\boldsymbol{X}, \boldsymbol{X}')$:

$$\mathrm{MDTW}(\boldsymbol{X}, \boldsymbol{X}') = \frac{1}{|\lambda^*|} \min_{\lambda \in \Lambda(\boldsymbol{X}, \boldsymbol{X}')} D_{\boldsymbol{\lambda}}(\boldsymbol{X}, \boldsymbol{X}')$$

where $\lambda^* = \arg\min_{\lambda \in \Lambda(\boldsymbol{X}, \boldsymbol{X}')} D_{\boldsymbol{\lambda}}(\boldsymbol{X}, \boldsymbol{X}')$. We normalize the $\boldsymbol{\lambda}$-alignment distance by $|\lambda^*|$ so that we can compare MDTW distances within a database of variable length time series. To convert MDTW distance to similarity, we use a radial basis function (RBF) kernel: $\kappa_{\mathrm{MDTW}}(\boldsymbol{X}, \boldsymbol{X}') = \exp\{-\mathrm{MDTW}(\boldsymbol{X}, \boldsymbol{X}')/(2\sigma)\}$ where $\sigma$ is a parameter we can choose or tune. In our experiments, we let $\sigma = \mathrm{median}_{\boldsymbol{X}, \boldsymbol{X}' \in \mathcal{D}}\ \mathrm{MDTW}(\boldsymbol{X}, \boldsymbol{X}')$, i.e., the median MDTW distance between time series in our database.

MDTW is computationally expensive, requiring $O(T^2)$ evaluations of our discrepancy function. One of the main ways to speed it up is to impose constraints on our set of alignments $\boldsymbol{\Lambda}$. For example, the Sakoe-Chiba band requires that aligned points be within a certain number of steps of one another, i.e., that $|\lambda(j,1) - \lambda(j,2)| \leq C$ [23]. In practice such constraints are implemented by using a limited window of possible alignments, but formally we add weights to the definition of $D_{\boldsymbol{\lambda}}$, such that any two points that violate a constraint (e.g., fall outside the Sakoe-Chiba band) receive a very large discrepancy. The computational complexity of MDTW with the Sakoe-Chiba band is $O(TC)$, which can be a relatively large gain if $T$ is large but $C$ is small. The price we pay for the speed-up is potentially suboptimal alignments.

**Global alignment kernels.** MDTW does not obey the triangle inequality and cannot be used (without modification) to define a positive definite kernel, a requirement for KLSH. MDTW also has a practical limitation: it defines distance based only on the optimal alignment between time series, ignoring other potentially poor alignments. A reasonable alternative might consider all possible alignments, in order to reward pairs of time series with multiple good alignments.

*Global alignment kernels* address both the theoretical and pragmatic limitations of MDTW [15]. We can think of a global alignment (GA) kernel as a weighted average over all possible $\boldsymbol{\lambda}$-alignment distances in $\Lambda(\boldsymbol{X}, \boldsymbol{X}')$, smoothing out the $\boldsymbol{\lambda}$-distances of outlier alignments (i.e., unusually small or large distances). More formally, [15] defines a global alignment

kernel as an exponentiated soft-minimum:

$$\begin{aligned}\kappa_{\mathrm{GA}}(\boldsymbol{X},\boldsymbol{X}') &= \sum_{\boldsymbol{\lambda}\in\Lambda(\boldsymbol{X},\boldsymbol{X}')} \exp\{-D_{\boldsymbol{\lambda}}(\boldsymbol{X},\boldsymbol{X}')\} \\ &= \sum_{\boldsymbol{\lambda}\in\Lambda(\boldsymbol{X},\boldsymbol{X}')} \exp\{-\sum_{j=1}^{|\boldsymbol{\lambda}|} d(\boldsymbol{x}(\lambda(j,1)),\boldsymbol{x}'(\lambda(j,2))\} \\ &= \sum_{\boldsymbol{\lambda}\in\Lambda(\boldsymbol{X},\boldsymbol{X}')} \prod_{j=1}^{|\boldsymbol{\lambda}|} \exp\{-d(\boldsymbol{x}(\lambda(j,1)),\boldsymbol{x}'(\lambda(j,2))\} \\ &= \sum_{\boldsymbol{\lambda}\in\Lambda(\boldsymbol{X},\boldsymbol{X}')} \prod_{j=1}^{|\boldsymbol{\lambda}|} s(\boldsymbol{x}(\lambda(j,1)),\boldsymbol{x}'(\lambda(j,2))\end{aligned}$$

where the function $s(\boldsymbol{x}(t_1),\boldsymbol{x}'(t_2))$ gives a measure of similarity between time points, analogous to the discrepancy function. This is also called the *local kernel* within the framework of *mapping kernels*, a recent generalization of convolutional kernels. [28] describe a set of conditions under which a mapping kernel is guaranteed to be positive definite for all local kernels, and [15] show that GA kernel satisfies these with only the mild assumption of *geometric divisibility*. Informally, this means that for positive definite local kernel $s(\boldsymbol{x},\boldsymbol{y})$, the ratio $s(\boldsymbol{x},\boldsymbol{y})/(1+s(\boldsymbol{x},\boldsymbol{y}))$ is also positive definite. This means that we should only use geometrically divisible kernels in our GA kernel definition to ensure its positive definiteness. This excludes, for example, Gaussian and Laplace kernels [15].

Like MDTW, GA kernel runs in $O(T^2)$ but can be sped up to $O(TC)$ using a second local kernel $\omega(t,t')$ over positions (instead of values), which is then multiplied by the similarity kernel. [15] recommend using a *triangular kernel*: $\omega(t,t') = (1-|t-t'|/C)_+$. This is analogous to the Sakoe-Chiba band.

**Vector autoregressive kernels.** The previous two kernels measure time series similarity based on shape. An alternative paradigm involves comparing time series in terms of higher level structure, such as temporal dependencies and correlations between different variables. Such structure is often detected by first specifying a type of model (often parametric and probabilistic) and then fitting a separate model to each time series to be compared. We can then measure similarity between time series based on the similarity of their model parameters.

One simple but very effective model is *linear vector autoregression* (VAR) [29]. An order-$L$ VAR or VAR($L$) model for a $P$ dimensional time series specifies that $x_i(t)$ is equal to a linear combination of observations of all $P$ variables from the previous $L$ time steps plus some zero mean Gaussian noise: $\boldsymbol{x}_t = \sum_{l=1}^{L} \boldsymbol{A}_l \boldsymbol{x}_{t-l} + \boldsymbol{b} + \boldsymbol{\varepsilon}_t$. $L$ is the lag, $\boldsymbol{A}_1,\ldots,\boldsymbol{A}_L \in \mathcal{R}^{P\times P}$ are the transition matrices, $\boldsymbol{b}\in\mathcal{R}^P$ the intercept, and $\boldsymbol{\varepsilon}\sim\mathcal{N}(0,\Sigma)$ the noise. We can rapidly train a VAR($L$) model on a time series $\boldsymbol{X}\in\mathcal{R}^{P\times T}$ using linear regression. We first preprocess the data into two matrices, $\boldsymbol{Z}\in\mathcal{R}^{P\times(T-L)}$ and $\boldsymbol{W}\in\mathcal{R}^{(PL+1)\times(T-L)}$. $\boldsymbol{Z}$ contains $T-L$ observations of each variable, which we stack horizontally to form a matrix of responses. $\boldsymbol{W}$ contains $T-L$ lagged windows that we use to predict the next time step, which we reshape into feature vectors for our regression. We then solve the ordinary least squares problem $\boldsymbol{Z}=\boldsymbol{B}\boldsymbol{W}$. The resulting $\widehat{\boldsymbol{B}} = \boldsymbol{Z}\boldsymbol{W}^{-1}$ contains our transition matrices and intercept. We can then compare two time series on the basis of their respective VAR($L$) transition matrices and intercepts.

[8] do away with the two step "train, then compare" process for VAR-based similarity by formulating a VAR-related covariance kernel using the $\boldsymbol{Z}$ and $\boldsymbol{W}$ matrices. Using a Bayesian linear regression framework with a non-informative prior, they define the following kernel function:

$$\begin{aligned}&\kappa_{\mathrm{VAR}}(\boldsymbol{X},\boldsymbol{X}') \\ &\quad= \left(|\mathcal{W}^\top\mathcal{W}\Delta+\boldsymbol{I}_c|^{1-\alpha} + |\mathcal{W}^\top\mathcal{W}\Delta+\mathcal{Z}^\top\mathcal{Z}\Delta+\boldsymbol{I}_c|^{\alpha}\right)^{-P/2}\end{aligned}$$

where $\mathcal{W}=[\boldsymbol{W}\ \boldsymbol{W}']$, $\mathcal{Z}=[\boldsymbol{Z}\ \boldsymbol{Z}']$, $c=T+T'-1$, and $\alpha$ depends on $P$ and the degrees of freedom $d$ of the matrix-normal inverse Wishart prior but can be treated as a tunable parameter. Note that this is the *Gram matrix* formulation; there is also a *Variance matrix* formulation. They are equivalent but give different computational performance, depending on the relative sizes of $T$ and $P$. [8] demonstrate that under certain conditions (specifically $d>P-1$), $\kappa_{\mathrm{VAR}}$ is positive definite and infinitely divisible, such that $\kappa_{\mathrm{VAR}}^{1/n}$ is also positive definite for all $n\in\mathbb{R}$. The latter property is especially convenient because it means that we can substitute a different exponent for $-P/2$ to make the result more numerically stable.

The VAR kernel is computationally expensive. The Gram formulation above is $\widetilde{O}(P+T^3)$, while the Variance formulation is $\widetilde{O}(T+P^3)$. If $T$ or $P$ is small (i.e., our time series are short or low dimensional), then we have a choice of formulation that will run efficiently. However, if $T$ and $P$ are both large, the VAR kernel may be slow and even intractable.

**Euclidean distance.** Euclidean distance (ED) is among the simplest of time series distance measures. We can think of ED as an alignment-based kernel with a fixed specific alignment. The main open question is how to apply ED for two time series with different lengths; there are a variety of choices (truncation, extension, resampling, interpolation), though none seems particularly justified. We choose a simple strategy: if the shorter time series is $\boldsymbol{X}'$, we append $T-T'$ copies of $\boldsymbol{x}'(T')$ to the end of $\boldsymbol{X}'$ so that the result has the same length as $\boldsymbol{X}$. Using the language of $\boldsymbol{\lambda}$-alignments, we define the ED alignment of two time series $\boldsymbol{X}$ and $\boldsymbol{X}'$ with $T'<T$:

$$\boldsymbol{\lambda}_{\mathrm{ED}} = \{(1,1),\ldots,(T',T'),\ldots,(T-1,T'),(T,T')\}$$

and then define variable length time series ED as

$$ED(\boldsymbol{X},\boldsymbol{X}') = \frac{1}{T}\sum_{j=1}^{T} d\left(\boldsymbol{x}(\lambda_{\mathrm{ED}}(j,1)),\boldsymbol{x}'(\lambda_{\mathrm{ED}}(j,2))\right)$$

This seems preferable to, for example, truncating the longer time series (which discards potentially useful information) or just choosing an alignment at random. We might consider using a model (e.g., fitting a line or training a VAR($L$) model) to extend the shorter time series, but doing so adds computational complexity for a presumably small improvement, when the primary virtue of ED is its $O(N)$ speed. As in the case of

MDTW, we normalize by the length of the longer time series so we that can fairly compare distances within our database.

To convert Euclidean distance to similarity, we use a radial basis function (RBF) kernel: $\kappa_{\mathrm{ED}}(\boldsymbol{X}, \boldsymbol{X}') = \exp\{-ED(\boldsymbol{X}, \boldsymbol{X}')/(2\sigma)\}$ where $\sigma$ is a parameter we can choose or tune. In our experiments, we let $\sigma = \mathrm{median}_{\boldsymbol{X},\boldsymbol{X}' \in \mathcal{D}}\ ED(\boldsymbol{X}, \boldsymbol{X}')$, i.e., the median Euclidean distance between time series in our data.

While this seems hopelessly naive, researchers have repeatedly shown that ED is often competitive with more sophisticated approaches, particularly for large databases of long time series [9]. Our approach seems reasonable for clinical time series: extending a shorter time series by repeating the final measurement assumes that the patient's final condition is relatively stable, which is true for both healthy patients who were discharged and in-hospital mortalities.

### C. Kernelized hashing framework

Now we provide an overview of a kernelized time series hashing framework. Suppose that we have a database $\mathcal{D}$ of $P$-dimensional, variable length MVTs, where $|\mathcal{D}| = N$ and that we want to represent each with a $B$ bit binary code. We can apply KLSH by performing the following steps:

- Choose a similarity kernel (e.g., MDTW, GA, VAR, ED).
- Randomly choose a sample $\mathcal{S} = \{\boldsymbol{X}_j\}_{j=1}^{M} \subset \mathcal{D}$, with $M \ll N$, and form the $M$-by-$M$ kernel matrix $\boldsymbol{K}$.
- Form $B$ hash functions. For the $j$th function, choose a subsample $\mathcal{R}_j \subset \mathcal{S}$ of $R < M$ examples to estimate the $j$th weights vector $\boldsymbol{w}_j = \boldsymbol{K}^{-1/2}\boldsymbol{e}_{\mathcal{R}_j}$.
- Use the hash functions to generate $B$ bit binary codes for each time series in $\mathcal{D}$.
- For a new query point $\boldsymbol{X}^q$, generate its $B$-bit hash code.
- Search for the $k$-nearest neighbors of $\boldsymbol{X}^q$ using standard LSH techniques (e.g., linear scan of binary codes or a hash plus a small number of object comparisons).

We can see that while the training stage (i.e., learning our hashing functions) is time-consuming, searching for the nearest neighbors of a new query should be dramatically faster. An exhaustive search over binary codes requires $O(N)$ comparisons, but on modern hardware, these are incredibly fast. If we apply the $(1+\varepsilon)$-near neighbor indexing strategy of [11], then we use the binary codes to generate a list of $O(N^{1/(1+\varepsilon)})$ candidate nearest neighbors and then perform that many similarity comparisons to choose the $K$ best (the choice of $\varepsilon$ trades off speed and accuracy). Our perceived speedup will also governed by two other factors: (1) our choice and implementation of similarity (or distance) function; and (2) the "size" of our database in terms of $N$, $P$, and $T$. If we have a fast similarity function (e.g., ED or MDTW with a narrow Sakoe-Chiba band and early stopping heuristics) and a small number of short univariate time series, then we may notice little or no gain. Even for slower similarity functions, if $N$ is relatively small and we use clever indexing or storage, we may perceive little or no speedup. However, for an expensive similarity function (of the kind that we use for multivariate time series) and a large database, we can expect substantial gains.

## IV. Experiments

### A. Data

We performed experiments with a number of different data sets from different domains, including two large, real world clinical databases and one EEG database. Each data set has its own unique set of properties, dynamics, and challenges, but they are all multivariate time series data sets with moderate to large numbers of examples $N$, moderate dimensionality $P$, and medium to long (and often variable) lengths $T$.

**PICU data.** The *PICU data set* (`picu`), a version of which was first described in [5], is a fully anonymized collection of clinical MVTs recorded over a decade in the pediatric ICU at Children's Hospital LA (CHLA). The version we work with has roughly 10,000 MVTs of $P = 13$ variables, including vital signs (e.g., heart rate), lab results (e.g., glucose), and subjective assessments (e.g., Glascow coma score). The duration of ICU visits is highly variable, though the vast majority of stays last fewer than seven days ($T = 128$ for an hourly sampling rate) and over half last fewer than two days ($T = 48$). This data set in part motivates our interest in this problem, as doctors and clinical researchers are increasingly interested in quantifying *patient similarity* in terms of complex temporal patterns (rather than *a priori* personal facts, like age or gender) and of discovering or searching for similar cases in large historical EHR databases.

This data comes with many interesting potential labels and responses; we utilize two in our experiments. The first is the patient's *primary diagnostic category*; each patient is assigned a single primary diagnosis, which comes from one of eleven different categories (there are originally fourteen categories, but we combine some of the extremely small ones). We use this category as a multiclass label. The second response we use is the *Pediatric Cerebral Performance Category* (PCPC) code, an integer-valued score between 1 and 5 indicating a patient's level of cognitive function [30].

Real world clinical time series data are *very* challenging to work with, and standard techniques often fail dramatically. [5] describe the challenges (and potential) of this of data in elegant detail. These include missing time series, irregular sampling, wildly different dynamics between variables, noise, and age dependency. Like [5], we impose an hourly bucketing on the data (taking the mean of multiple measurements within the same bucket). Where this creates missing values, we propagate forward the previous measurement; variables with a sampling frequency of less than an hour typically change very slowly, so this is a fairly reasonable assumption. When a time series are missing entirely, we impute a normal value for this variable; this is also a reasonable assumption, as variables are often missing because clinical staff believed them to be normal and chose not to measure them.

**Surgical data.** The *surgical data set* (`surgical`) includes fully anonymous MVT time series extracted from an operating room anesthesia database at CHLA. It includes over

60,000 MVTs that describe patient status and treatment during surgery. This data is sampled at a much higher frequency than the `picu` data set (roughly one measurement/minute, versus one measurement/hour), yielding richer and longer time series. There are $P = 15$ variables, consisting largely of vital signs, and length varies between $T = 30$ samples and values of $T$ in the thousands. The median value of $T$ is 80. `surgical` exhibits many of the same phenomena as `picu` (missing data, irregular sampling, noise), though less extreme due to the higher sampling frequency. We apply the same kind of preprocessing to `surgical` as we did to `picu`.

Clinical researchers are interested in searching this database for adverse events (for example, a desaturation or a bronchospasm). However, only about $1\%$ of cases have a label (all positive); the remaining cases have no label, positive or negative. This is both a limitation and an opportunity. It limits our ability to assess the performance of search algorithms on the currently labeled data set. However, it also illustrates a potential use case for time series similarity search over large medical databases: finding and labeling potential adverse events in such databases would enable both clinical research and quality assurance studies. However, finding such events is a *needle in a haystack* problem for clinicians; the current suite of software tools available at most hospitals limit them to searching for keywords in free text notes (unreliable) or "eyeballing" thousands of cases (tedious *and* unreliable).

**EEG data.** The *EEG Database data set*[2] (`eeg`) is a classic machine learning benchmark data set from the *UCI Machine Learning Repository* [31] that was originally collected during a study on the correlation between EEG patterns and alcoholism. It is a reasonably large MVT data set, including nearly 11,000 MVTs with $P = 64$ channels and consistent length $T = 256$. While it is smaller than `surgical`, it has four times as many variables and is longer on average. It includes binary labels indicating whether the subject is an alcoholic and is reasonably class balanced ($64\%$ versus $36\%$).

**Normalization.** In the above data sets, the ranges and magnitudes of different variables vary substantially; using this data as is could bias our time series distances. Furthermore, it is well known that both the shape-based kernels (MDTW and GA) work best with zero mean, unit variance data and that VAR models make similar assumptions, including stationarity. Thus, we applied standardization, including shifting (e.g., by the mean) and scaling (e.g., by the standard deviation). We used the *overall* mean and standard deviation (i.e., computed across all training data), as opposed to each individual MVT's statistics. While this does not in fact ensure that each time series is zero mean, unit variance, we feel this was a reasonable compromise for our purposes.

### *B. Design and goals*

The goal of our experiments is to demonstrate that kernelized hashing can be utilized to speed up time series similarity search based on a variety of different similarity metrics, as well as to confirm our hypothesis that no one similarity measure is best. To that end, we evaluate the performance of our algorithms using the following three criteria inspired from parallel work on hashing in computer vision [12]:

1) **Semantic accuracy:** For data sets where ground truth labels are available, we measure the semantic accuracy $\mathbf{sacc}(\boldsymbol{X}^q, k\mathrm{nn}(\boldsymbol{X}^q))$ of a $k$nn search by the fraction of nearest neighbor labels that agree with the $\boldsymbol{X}^q$'s label, where $\mathbf{sacc}(\boldsymbol{X}^q, \{\boldsymbol{X}_j\}_{j=1}^K) = \left(\sum_{j=1}^K \mathbb{1}\{y^q = y_j\}\right)/K$. This evaluates the potential efficacy of using our search procedure for, e.g., $k$nn classification tasks. We are less interested in the actual semantic accuracy of a particular search technique than we are in the *gap* (or difference) between the semantic accuracy of an exhaustive search using a similarity measure and the semantic accuracy of a KLSH search based on the same measure. We define this gap as

$$\begin{aligned}&\mathbf{gap}\left(\boldsymbol{X}^q, k\mathrm{nn}_{\mathrm{GT}}, k\mathrm{nn}_{\mathrm{KLSH}}\right)\\ &= \mathbf{sacc}(\boldsymbol{X}^q, k\mathrm{nn}_{\mathrm{GT}}(\boldsymbol{X}^q)) - \mathbf{sacc}(\boldsymbol{X}^q, k\mathrm{nn}_{\mathrm{KLSH}}(\boldsymbol{X}^q))\end{aligned}$$

   If this gap is small, then our hashing procedure is doing a good job of approximating an exhaustive search; if it is large, then we face a trade off between computational performance and accuracy. Because we use only predefined kernels, there is no reason to expect our hashing procedure to beat the exhaustive search, and such results are probably anomalous. Note that we use the same measure of accuracy for both binary and multiclass labels. For the integer-valued PCPC response in `picu`, we measure semantic accuracy using the average mean squared error (MSE) between the query and nearest neighbor responses.
2) **Nearest neighbor recall:** We can assess the efficacy of our hashing procedure by examining the degree to which it approximates the true neighborhood around a query $\boldsymbol{X}^q$. We measure this in the form of *recall*: how many nearest neighbors does KLSH need to retrieve in order to discover a fixed number $K$ of *ground truth nearest neighbors* (i.e., using the similarity measure). This can be plotted like a *recall-precision curve*, with the number of nearest neighbors returned by KLSH on the $x$-axis and the count of the $K$ target nearest neighbors found on the $y$ axis.
3) **Relative speed-up:** finally, we are interested in just how much of a speed-up we get using hashing vs. an exhaustive search. We measure this as a ratio of the average query time for the exhaustive search ($\mathrm{time}_g$) divided by the average query time for a hashing-based search ($\mathrm{time}_h$): $\mathbf{speedup} = \mathrm{time}_g/\mathrm{time}_h$. The larger this number, the bigger the speed-up.

For each experiment, we perform *stratified 10-fold* or *5-fold cross-validation* to estimate our performance statistics. We average accuracy, recall, and speed-up across queries and then take the mean of averages across folds. We use no special preprocessing of the data beyond basic standardization and do

[2]https://archive.ics.uci.edu/ml/datasets/EEG+Database

not tailor our kernels to each data set. For the VAR kernel, we used a fixed lag of $L = 5$, as suggested by [8]. When performing hashing-based search for nearest neighbors, we perform a single linear scan in the space of binary codes, with no subsequent set of comparisons using the core similarity function. It is worth noting that including this last step would improve our accuracy.

### C. Results and Discussion

**Table I** shows the *relative speed-up* ($\text{time}_g/\text{time}_h$) that we get from using KLSH for each kernel: we define this as the ratio of search time using the true similarity function ($\text{time}_g$) divided by the search time using the hashing-based approach ($\text{time}_h$). We observe a fairly significant gain in speed across the board, demonstrating the generality of our approach. There are two other important trends in the speed-ups. First, the more expensive the kernel, the more speed-up we receive from using hashing. Second, the speed-up also increases as our data set size grows. The alignment kernels (MDTW and GA) receive over two orders of magnitude boost in their speed for the huge `surgical` data set. These results confirm our intuition that hashing is a very useful tool for time series similarity search, if our search accuracy is acceptable. The bottom right bar plot in **Figure 1** shows the actual average query times on the `picu` data set. The ground truth VAR kernel proved intractable for the `surgical` and `eeg` data sets; in this case, hashing is the only viable solution.

We present semantic accuracy and gap results in **Table II**. We see that for all kernels, the gap in accuracy between the ground truth similarity metric and KLSH search is usually small and always less than 0.05. This is pretty remarkable given that we are retrieving nearest neighbors using a pure linear scan of the binary hash codes and performing no distance-based comparisons to refine our results. Also worth noting is that, as we anticipated, there is not a decisive winner among the kernels, though GA tends to have the best performance (in terms of both semantic accuracy and gap) on average. ED performs surprisingly well on `picu`; we see in **Figure 1** that it has the best gap for both diagnosis accuracy and PCPC MSE loss.

One curious result in **Figure 1** is the discrepancy between the semantic accuracy results in the top left plot and the 10-nearest neighbor (10nn) recall in the bottom left. KLSH does a much worse job of capturing the true neighborhood structure of MDTW and VAR than it does with GA, but there is relatively small difference in terms of semantic accuracy. This can be explained by the labels that we chose to use for our `picu` experiments. Notice in **Table II** that all ground truth similarities have low 10nn semantic accuracy (around 0.2), suggesting that there is little correlation between the structure of any local similarity neighborhood and the labels. This is not surprising, given what we know about the source of our labels. We used patient primary diagnosis codes from a custom database for our classification task. Patients typically receive multiple diagnostic codes, but only one is designated as the primary diagnosis, based on a variety of factors not limited to patient status and physiology.

For that reason, we feel that the neighborhood recall results are more significant for assessing the success of our time series hashing framework. **Figure 1** and **Figure 2** show the 10nn recall results for `picu` and `surgical`, respectively. We see that GA does a modest job of recalling the true neighborhood structure for query points. ED works well for `picu` but is terrible for `surgical`. MDTW does not appear to work well at all; this confirms what theory suggests, namely that MDTW is a poor kernel without significant modifications. GA is appears to be the superior alignment-based similarity for kernel methods. Again, we note that no one similarity works best for all data sets.

It is disappointing that VAR does not perform well on the `picu` data set (the only one for which the ground truth search completed). There are several possible explanations, the main one being that it makes strong modeling assumptions about the data (linear correlations, stationarity, etc.) that are likely not true for these real world data sets. Linear models seem especially poorly suited for the PICU data where the sampling rate is so low. Additionally, VAR has many tunable parameters with which we did not experiment, foremost among them the lag parameter $L$. We omit `eeg` results because the ground truth VAR search failed to terminate; this represents an extreme but important case where hashing is the only feasible option.

**Relative speed-up**

| Kernel | `picu` | `surgical` | `eeg` |
|---|---|---|---|
| ED | 9.78 | 76.50 | 18.70 |
| MDTW | 19.13 | 163.00 | 31.73 |
| GA | 18.17 | 175.02 | 32.08 |
| VAR | 20.34 | – | – |

TABLE I: Relative speed-up ($\text{time}_g/\text{time}_h$) for each time series kernel and data set, when performing an exhaustive search. The ground truth VAR kernel was intractable for `surgical` and `eeg`.

**10nn semantic accuracy** and **gap**

| | | `picu` (diagosis) | `eeg` | `picu` (PCPC $L_2$ loss) |
|---|---|---|---|---|
| ED | **sacc** | 0.205 | 0.561 | **2.36** |
| | **gap** | **0.010** | 0.046 | **0.053** |
| MDTW | **sacc** | 0.204 | 0.565 | 2.53 |
| | **gap** | 0.047 | 0.049 | 0.41 |
| GA | **sacc** | **0.210** | **0.579** | 2.45 |
| | **gap** | 0.032 | **0.036** | 0.193 |
| VAR | **sacc** | 0.181 | – | 2.41 |
| | **gap** | 0.040 | – | 0.121 |

TABLE II: Semantic (label) accuracy and gap for 10nn retrieval across data sets and kernels. For PCPC codes, we measure the average squared difference from the query point's score (i.e., lower is better).

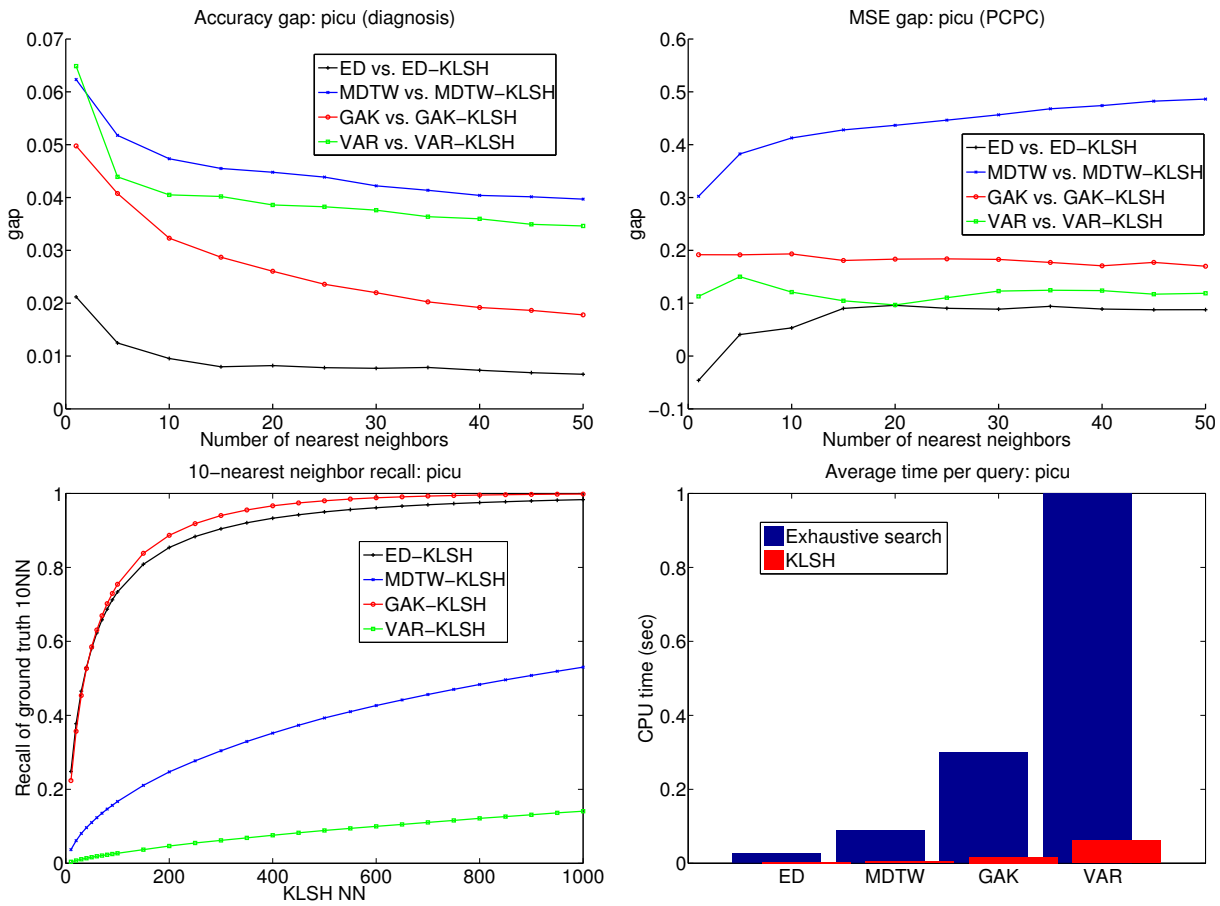


Fig. 1: Results for `picu` data set. *Top row*: semantic accuracy gap for diagnosis label and PCPC code. *Bottom left*: 10-nearest neighbor recall. *Bottom right*: average query times.

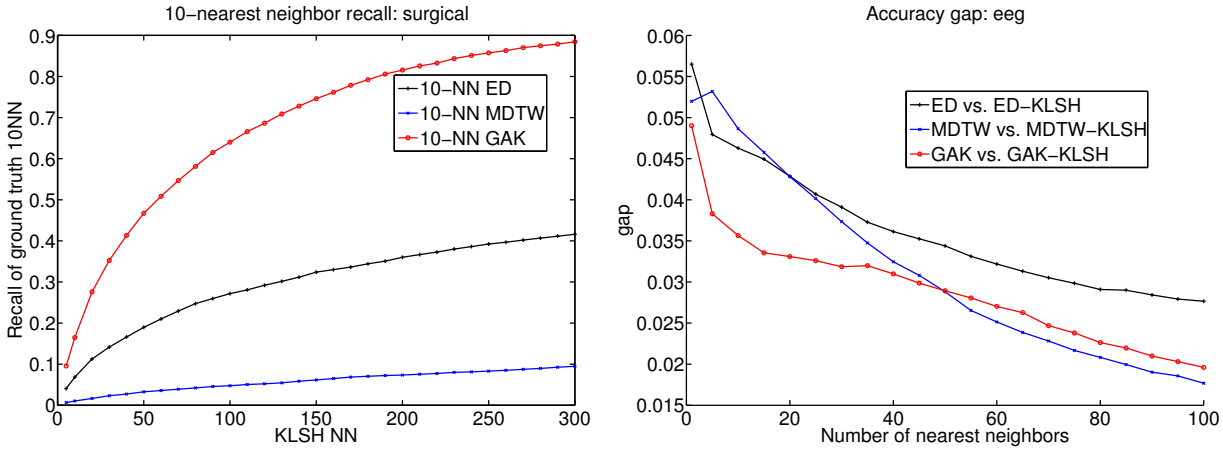


Fig. 2: Left: 10-nearest neighbor recall for `surgical`. Right: semantic accuracy gap for `eeg`.

## V. Conclusion and Future Work

There is a growing need for fast and accurate similarity search over multivariate time series, particularly in health care. Many time series similarity measures have been proposed by researchers, but no one metric works best across all data sets and problems. What is more, most of these approaches do not scale to large time series data sets. Kernelized hashing presents a flexible, unified approach to speeding up time series search, regardless of choice of representation and distance measure. We have described how the general framework of *kernalized locality-sensitive hashing* (KLSH) can be combined with arbitrary time series similarity metrics to yield efficient and

accurate similarity search. Using three large, complex medical data sets, we demonstrated empirically that this framework is orders of magnitude faster than existing approaches and provides an acceptable trade off between speed and accuracy.

The future of this line of work is quite promising. A logical next step would be to incorporate other time series metrics into the KLSH framework and even combine multiple metrics to capture different notions of similarity. We would also like to incorporate supervision into the hash function learning, which has been shown to work well for images [17] [18]. We focused our attention on locality sensitive hashing and a handful of representative similarity functions, but the kernel approach may be applied to other hashing frameworks. Finally, we feel that the major limitation of this framework is that it is non-adaptive. We would like to explore adaptive, learning-based approaches for selecting coding functions, such as deep learning [21] [18]. We feel that these will not only improve the performance of hashing-based search but may also be useful for discovering interesting structure in and generating novel representations of multivariate time series.

## ACKNOWLEDGMENTS

The research was sponsored by the NSF research grants IIS-1134990 and IIS-1254206 and the Okawa Foundation Research Award. David Kale is supported by the Alfred E. Mann Innovation in Engineering Doctoral Fellowship. The VPICU is supported by grants from the Laura P. and Leland K. Whitter Foundation. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agency, or the U.S. Government. We would also like to thank Sanjeev Kumar, Ping Li, and our anonymous reviewers for their helpful feedback.

## REFERENCES

[1] A. Perer and J. Sun, "Matrixflow: Temporal network visual analytics to track symptom evolution during disease progression." *AMIA Annu Symp Proc*, vol. 2012, pp. 716–25, 2012.

[2] L. Lehman, M. Saeed, G. Moody, and R. Mark, "Similarity-based searching in multi-parameter time series databases." *Comput Cardiol*, vol. 35, no. 4749126, pp. 653–656, 2008.

[3] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003.

[4] T. K. Vintsyuk, "Speech discrimination by dynamic programming," *Cybernetics*, vol. 4, no. 1, pp. 52–57, 1968.

[5] B. M. Marlin, D. C. Kale, R. G. Khemani, and R. C. Wetzel, "Unsupervised pattern discovery in electronic health care data using probabilistic clustering models," in *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, 2012, pp. 389–398.

[6] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[7] H. Chen, F. Tang, P. Tino, and X. Yao, "Model-based kernel for efficient time series analysis," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 392–400.

[8] M. Cuturi and A. Doucet, "Autoregressive kernels for time series," *arXiv:1101.0673 [stat.ML]*, 2011. [Online]. Available: http://arxiv.org/abs/1101.0673

[9] J. Lin, S. Williamson, K. D. Borne, and D. De Barr, "Pattern recognition in time series," in *Advances in Machine Learning and Data Mining for Astronomy*, ser. Data Mining and Knowledge Discovery series, M. Way, J. Scargle, K. Ali, and A. Srivastava, Eds. CRC Press, 2012, ch. 28.

[10] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 262–270.

[11] 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*, 1999, pp. 518–529.

[12] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," in *Machine Learning for Computer Vision*, ser. Studies in Computational Intelligence, R. Cipolla, S. Battiato, and G. M. Farinella, Eds. Springer Berlin Heidelberg, 2013, vol. 411, pp. 49–87.

[13] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[14] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1092–1104, June 2012.

[15] M. Cuturi, "Fast global alignment kernels." in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 929–936.

[16] J. Shieh and E. Keogh, "iSAX: Indexing and mining terabyte sized time series," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 623–631.

[17] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 209–216.

[18] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[19] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1753–1760.

[20] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1509–1517.

[21] M. Norouzi, D. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1070–1078.

[22] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002, pp. 380–388.

[23] H. Sakoe and S. Chiba, "Readings in speech recognition," 1990, ch. Dynamic Programming Algorithm Optimization for Spoken Word Recognition, pp. 159–165.

[24] E. Keogh, "Exact indexing of dynamic time warping," in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002, pp. 406–417.

[25] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[26] H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in support vector machine," in *Advances in Neural Information Processing Systems 14*, 2002, pp. 921–928.

[27] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *J. Intell. Inf. Syst.*, vol. 39, no. 2, pp. 287–315, Oct. 2012.

[28] K. Shin and T. Kuboyama, "A generalization of haussler's convolution kernel mapping kernel and its application to tree kernels," *Journal of Computer Science and Technology*, vol. 25, no. 5, pp. 1040–1054, 2010.

[29] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Springer, 2005.

[30] D. Fiser, "Assessing the outcome of pediatric intensive care," *Pediatrics*, vol. 121, no. 1, pp. 68–74, 1992.

[31] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml