AFRL-RV-PS-
TR-2021-0056

TIME SERIES DATA MINING TECHNIQUES FOR SEISMIC ANAYLSIS

Abdullah A. Mueen, et al.

University of New Mexico
Department of Computer Science
1 University of New Mexico
Albuquerque, NM 87131

05 May 2021

Final Report

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-5776

DTIC COPY

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by AFMC/PA and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RV-PS-TR-2021-0056 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//	//SIGNED//
Dr. Frederick Schult	Erin N. Pettyjohn, Chief
Program Manager, AFRL/RVBN	AFRL Geospace Technologies Division

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
05-05-2021	Final Report	24 Apr 2017 – 05 May 2021

4. TITLE AND SUBTITLE
Time Series Data Mining Techniques for Seismic Data Analysis

5a. CONTRACT NUMBER
FA9453-17-C-0024

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62601F

6. AUTHOR(S)
Abdullah A. Mueen, Mohammad Ashraf Siddiquee, Sheng Zhong, and Farhan Asif Chowdhury

5d. PROJECT NUMBER
1010

5e. TASK NUMBER
EF129469

5f. WORK UNIT NUMBER
V132

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of New Mexico
Department of Computer Science
1 University of New Mexico
Albuquerque, NM 87131

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Space Vehicles Directorate
3550 Aberdeen Avenue SE
Kirtland AFB, NM 87117-5776

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RVBN

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RV-PS-TR-2021-0056

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited (AFMC-2021-2860 dtd 26 Aug 2021).

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We applied advanced data mining techniques for seismic data processing, developing seismic signal discovery, classification and monitoring algorithms. In signal discovery we developed a semi-supervised motif discovery algorithm that forms a nearest neighbor graph to identify chains of nearest neighbors from the given events and demonstrated that the chains are likely to identify hidden patterns in the data. In signal classification we developed a machine-learned model that takes input from a signal detector and produces phase types as output for a signal associator. The model is a combination of convolutional and long short-term memory networks. It outperforms existing baselines and has consistently good performance for novel sources and stations. In signal monitoring we developed a technique that efficiently computes both filtering and correlation in a single step. We achieved an orderof magnitude speed-up by maintaining frequency transforms over sliding windows. The method is exact, devoid of sensitive parameters, and easily parallelizable. We have provided a publicly available real-time system that employs the algorithm for monitoring a seismic network.

15. SUBJECT TERMS
seismic network processing, machine learning, seismic phase identification, seismic signal detection, seismic event detection

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Frederick R. Schult
Unclassified	Unclassified	Unclassified	Unlimited	82	**19b. TELEPHONE NUMBER** *(include area code)*

This page is intentionally left blank.

Approved for public release; distribution is unlimited.

TABLE OF CONTENTS

Approved for public release; distribution is unlimited.

TABLE OF CONTENTS (continued)

Approved for public release; distribution is unlimited.

LIST OF FIGURES

Approved for public release; distribution is unlimited.

LIST OF FIGURES

Figure **Page**

Approved for public release; distribution is unlimited.

LIST OF TABLES

Approved for public release; distribution is unlimited.

Abstract

The aim of this contract was to explore advanced data mining techniques for seismic data processing. More specifically, we developed seismic signal discovery, classification and monitoring algorithms. The objective was to improve upon the state-of-the-art techniques in accuracy, novelty, and efficiency.

Signal Discovery: We start with research on a semi-supervised motif discovery algorithm for signal detection. Unlike semi-supervised clustering, classification, and rule discovery; semi-supervised motif discovery is a surprisingly unexplored area in data mining. *Semi-supervised Motif Discovery* finds hidden patterns in long time series (i.e. seismic signals) when a few arbitrarily known patterns are given. A naive approach is to exploit the known patterns and perform similarity search (i.e. template matching) within a radius of the patterns (i.e. correlation threshold). However, this method would find only similar shapes and would be limited in discovering new shapes. In contrast, traditional unsupervised motif discovery algorithms detect new shapes, while missing some patterns because the given information is not utilized. We developed a semi-supervised motif discovery algorithm (SeiSMo) that forms a nearest neighbor graph to identify chains of nearest neighbors from the given events. We demonstrate that the chains are likely to identify hidden patterns in the data.

Signal Classification: Automated seismic phase identification is an integrated component of large scale seismic monitoring applications, including earthquake warning systems and underground explosion monitoring. Accurate, fast, and fine-grained phase identification is instrumental for earthquake location estimation, understanding Earth's crustal and mantle structure for predictive modeling, etc. However, existing operational systems utilize multiple nearby stations for precise identification, which delays response time with added complexity and manual interventions. Moreover, single-station systems mostly perform coarse phase identification. In this project, we revisit seismic phase classification as an integrated part of a seismic processing pipeline. We develop a machine-learned model Phase Identifier (FASER), that takes input from a signal detector and produces phase types as output for a signal associator. The model is a combination of convolutional and long short-term memory networks. Our method identifies finer wave types, including crustal and mantle phases. We conduct comprehensive experiments on real datasets to show that FASERoutperforms existing baselines. We eval-uate FASER holding out sources and stations across the world to demonstrate consistent performance for novel sources and stations.

Signal Monitoring: Seismic monitoring systems have hundreds or thousands of distributed sensors gathering and transmitting real-time streaming data. To detect seismic events early, one might want to compute pairwise correlation across the disparate signals generated by the sensors. Since the data sources (e.g., sensors) are spatially separated, it is essential to consider the lagged correlation between the signals. Besides, many applications require processing a specific band of frequencies depending on the event's type, demanding a pre-processing step of filtering before computing correlations. Due to the high-speed of data generation and a large number of sensors in these systems, the operations of filtering and lagged cross-correlation need to be efficient to provide real-time responses without data losses. In this project, we propose a technique named FilCorr that efficiently computes both operations

Approved for public release; distribution is unlimited.

in one single step. We achieve an order of magnitude speed-up by maintaining frequency transforms over sliding windows. Our method is exact, devoid of sensitive parameters, and easily parallelizable. Besides our algorithm, we also provide a publicly available real-time system named SeisViz that employs FilCorr for monitoring a seismic network.

Approved for public release; distribution is unlimited.

This page is intentionally left blank.

Approved for public release; distribution is unlimited.

Chapter 1

SeiSMo: Semi-supervised Seismic Signal Discovery

1.1 Summary: Signal Discovery

Unlike semi-supervised clustering, classification, and rule discovery; semi-supervised motif discovery is a surprisingly unexplored area in data mining. *Semi-supervised Motif Discovery* finds hidden patterns in long time series when a few arbitrarily known patterns are given. A naive approach is to exploit the known patterns and perform a similarity search within a radius of the patterns. However, this method would find only similar shapes and would be limited in discovering new shapes. In contrast, traditional unsupervised motif discovery algorithms detect new shapes, while missing some patterns because the given information is not utilized.

We propose a semi-supervised motif discovery algorithm that forms a nearest neighbor graph to identify chains of nearest neighbors from the given events. We demonstrate that the chains are likely to identify hidden patterns in the data. We have applied the method to find novel events in several geoscientific datasets more accurately than existing methods.

1.2 Introduction: Signal Discovery

Motif discovery from time series data is a well studied problem in data mining. The typical objective in motif discovery is to identify approximately repeating segments in a time series. Each pattern that repeats significantly, either with high number of occurrences or with high similarity among the occurrences, is called a motif. For example, in Figure 1, there are four occurrences of a sinusoidal motif.

Existing motif discovery algorithms perform unsupervised searches for the most similar occurrences of a motif and, expand the motif set by identifying relatively less similar occurrences [42][81]. However, it may be possible that some occurrences of a motif are known. Such information can lead us to *semi-supervised* motif discovery algorithms, which would be *faster* than the completely unsupervised algorithms and, more *robust* than simple similarity search with a domain dependent parameter.

Semi-supervised motif discovery has the potential to enable data mining in domains where

Approved for public release; distribution is unlimited.

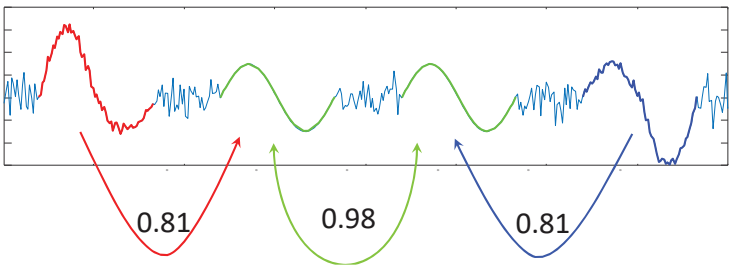

Figure 1. A time series with a sinusoidal motif appearing four times. *The arrows below show the nearest neighbor and correlation coefficients. If the red(left) and blue(right) sinusoids are given as example occurrences of the motif, it is intuitive that pure/middle sinusoids should also be occurrences. Traditional unsupervised motif discovery algorithms can find the pure/ middle sinusoids at the cost of expensive because computation of quadratic time complexity. Moreover, recognizing the middle/green sinusoids as motifs does not help in selecting a threshold for further similarity searches.*

significant manual annotation exists already. In seismic data analysis, well curated catalogs of seismic events are maintained by Incorporated Research Institutions for Seismology (IRIS) [2], Northern California Earthquake Data Center (NCEDC) [3]], Southern California Earthquake Data Center (SCEDC) [5] etc. Yet, many events of interest, especially low magnitude ones, are hidden in historical data that can produce valuable insights for geological scientists.

In the first part of this project, we propose a semi-supervised motif discovery algorithm. We name our technique **SeiSMo** (Semi-Supervised Motif). The algorithm performs an iterative nearest neighbor search to find other occurrences of the given motifs. SeiSMo has the following desirable properties:

- SeiSMo is an order of magnitude faster than unsupervised algorithms and can scale to millions of observations.
- SeiSMo works with any distance measure.
- SeiSMo is deterministic and less sensitive to parameters.
- SeiSMo can discover motifs with domain significance.

We develop two optimization strategies to speed up the computation of SeiSMo over a naive approach. We have applied SeiSMo on four seismological datasets from three states in the U.S. to discover numerous uncataloged seismic events that were missed by existing unsupervised techniques. SeiSMo is very promising in detecting *low magnitude* events which are not commonly cataloged by any manual effort.

Challenges in Semi-supervised Motif Discovery

One may consider a simple similarity search within a threshold of the given motifs to identify the remaining occurrences of the motifs. However, depending on the quality of the given events, similarity searches achieve mixed results.

Motif discovery is a special case of clustering where the goal is to identify a few clusters (typically less than five) of highly coherent subsequences in a time series, while leaving

Approved for public release; distribution is unlimited.

most of the data unclustered. In semi-supervised motif discovery, we define that one or more subsequences from one or more motifs are labeled; and the goal is to find all of the motifs. Most of the current work on semi-supervised algorithms for time series data focus on classification and clustering [9][14], in general. We provide an efficient semi-supervised motif discovery algorithm with applications to seismic data analysis.

To elaborate this idea, consider a set of points in two dimensional space in Figure 2(a). Note that time series segments are points in very high dimensional space. We consider two dimensional points for better illustration. The dense lump is a motif with *five* occurrences or repetitions. State-of-the-art motif discovery algorithms will find the closest pair of points (marked in red color) in Figure 2(a), and then search the region within a radius/threshold, typically proportional to the closest pair distance. In the Figure 2(a), all of the five occurrences can be identified by existing techniques for the illustrated radius/threshold.

Now consider the same set of points, along with the star-marked points showing the given occurrences representing some physical events (e.g. earthquakes). In Figure 2(b), all of the points in the dense lump are closer to the stars, hence, they are very well detectable by similarity search [81] based on the same radius as in 2(a). In Figure 2(c), no point is closer to the stars, hence, no motif would be found. In Figure 2(d), two of the five points will be found by the same search radius. Thus, the number of similar points detected by similarity search depends on the position of the stars and the radius/threshold value.

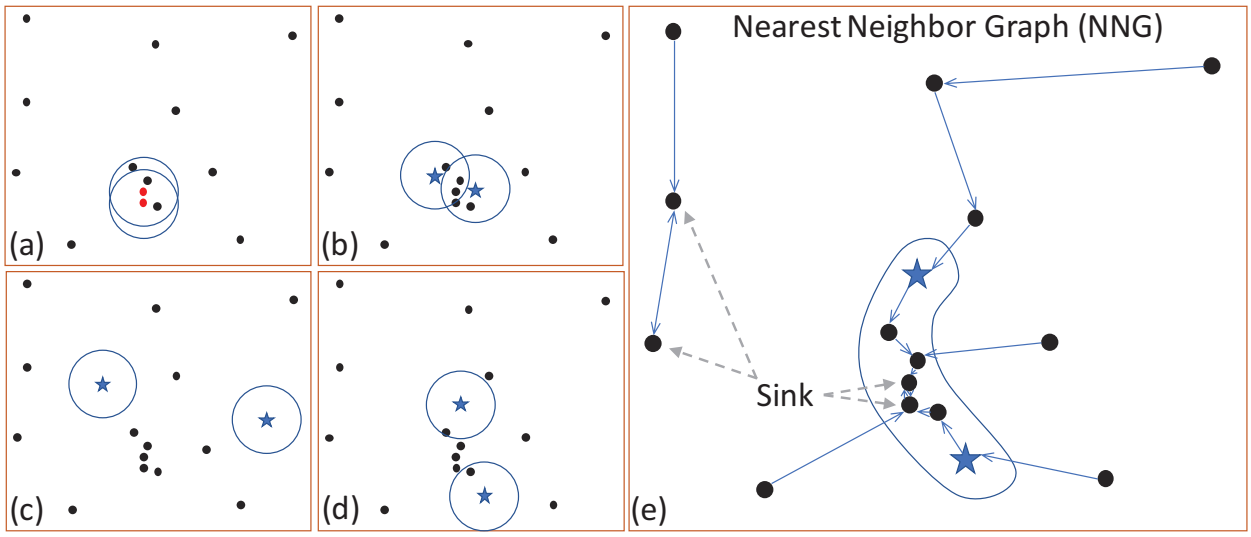

Figure 2. A set of two-dimensional points. Unsupervised radial search at the closest-pair (a). Successful (b), failed (c) and partially successful (d) similarity search at given (i.e., star) points. 2(e) is an enlarged version of 2(d). Nearest neighbor chains starting from the stars contains all the five points (e).

To draw a quick contrast and build intuition, we enlarge Figure 2(d) in 2(e), where SeiSMo can find all five occurrences in all of the scenarios (b-d). An arrow in Figure 2(e) connects a point to its nearest neighbor. If we follow the chain of nearest neighbors starting at the star points, we reach the same pair of points, which we call a *sink*, that are the nearest neighbors of each other. If we label all nodes on these chains as occurrences of the given motif, we correctly identify the five points without any threshold parameter.

Approved for public release; distribution is unlimited.

1.3 Methods, Assumptions, and Procedures: Signal Discovery

1.3.1 Background and Notation

We define *time series* as a sequence of real numbers measuring a quantity at a fixed sampling rate. A *time series subsequence* is a continuous segment of a time series. We can extract $n - m + 1$ time series subsequences of length m from a long time series of length $n >> m$. Time series subsequences are not independent of each other. Overlapping subsequences are trivially close in the high dimensional space, however, they are not interesting for mining purposes.

To elaborate on this fact, in Figure 3, we show a toy time series and all the subsequences of length three as points in three dimensional space. The points are not independently scattered, instead, the points form a trail where the nearest neighbor of a point (i.e. a window) is the next point on the trail (i.e. next slide of the window). To avoid such trivial nearest neighbors, we define a threshold O_t (typically 50%) to set *the minimum overlap between two subsequences* required to identify them as trivial matches. In other words, if two subsequences share more than half of their observations we consider them trivial because almost all computational properties (moments, frequency distribution, etc.) of two half-overlapping subsequences are governed by the overlapping segments and the lost information is at the Nyquist frequency. The choice of $O_t = 50\%$ has been empirically validated in related work in the literature [81][87]. In Figure 3(right), we connect a point with its non-overlapping ($< O_t$) nearest neighbor. Clearly, the nearest neighbors now contain more information, hence are non-trivial.

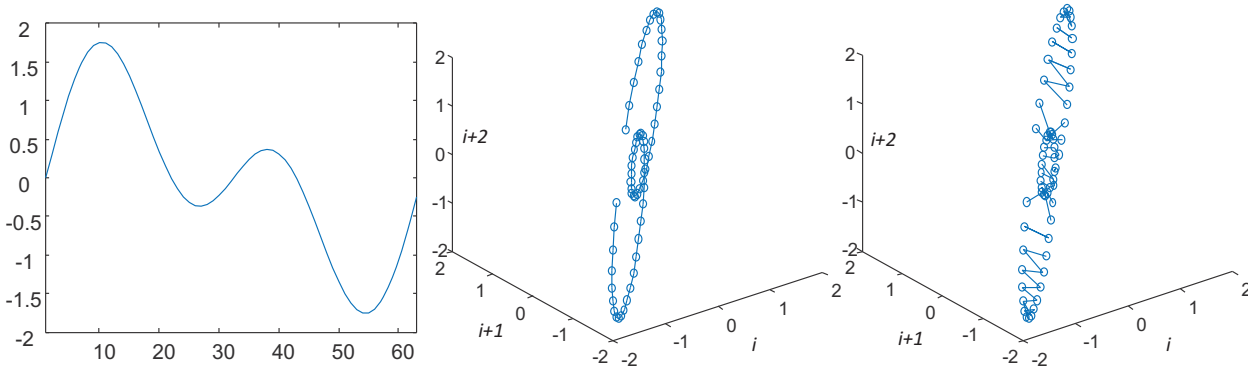

Figure 3. (left) A toy time series. (middle) The subsequences of length three in a 3D space form a trail. The nearest neighbor of a point/subsequence is trivially the next point on the trail. (right) The nearest non-overlapping neighbors are not trivial and, can possibly be in anywhere on the trail.

Nearest Neighbor: Given a subsequence $S_{i,m}$, that starts at the i^{th} index in the time series and is of length m, we define the nearest neighbor of $S_{i,m}$ as below,

$$\min_{|j-i|\geq m} d(zNorm(S_{i,m}), zNorm(S_{j,m})) \tag{1}$$

Approved for public release; distribution is unlimited.

Here, d can be any distance function defined to calculate distance between two equi-length time series. Typical distance functions are Euclidean distance [44], Dynamic Time Warping distance [53], Longest Common Subsequences distance [70], and Move-Split-Merge distance [69]. $zNorm$ is the standard z-normalization defined as $zNorm(x) = \frac{x-\mu}{\sigma}$, where μ and σ are estimates of mean and standard deviation of the observations in the vector x. Searching for the nearest neighbor of a query subsequence is extensively studied under all of these distance measures. In this work, we search for nearest neighbors under Euclidean distance using MASS [44] and DTW using UCR_Suite [53].

Nearest Neighbor Graph (NNG): An NNG is a graph (V, E) where V is the set of all subsequences of length m from a time series of length n. $(u \rightarrow v) \in E$ if v is the nearest neighbor of u as defined above. The *length* of an edge $(u \rightarrow v) \in E$ is the z-normalized distance between u and v.

Properties of NNG:

- The NNG is a collection of paths.
- Only cycles of size two are possible, which are called *sinks* in this work.
- The edges along the paths are non-increasing in length. Every path ends in a *sink*.

The above properties of an NNG are invariant to the distance measure used to find the nearest neighbors. Multiple unique paths can lead to a sink. We define such paths as **Confocal Paths** highlighting the fact that they all end in the same sink. We define the **support** of a sink, P, by the number of given occurrences of a motif (i.e. star points in Figure 2 and Figure 4) on any confocal path leading to the sink. The support of the sinks in Figure 2(e) are zero and two. See Figure 4 for more examples.

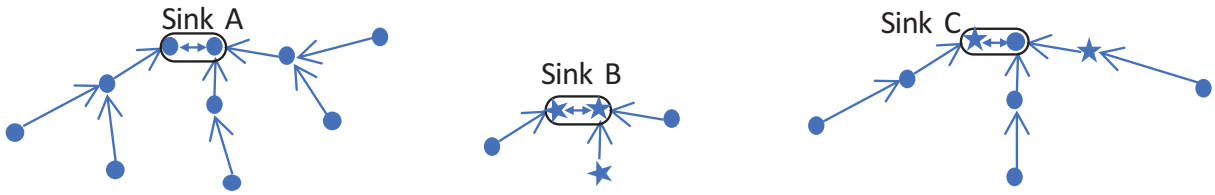

Figure 4. Three sinks (A,B and C) and their sets of Confocal Paths. The supports of A,B and C are zero, three and two, respectively

The intuition behind SeiSMo is that, if there are several paths from several labeled nodes/subsequences that lead to the same sink, the likelihood of the paths containing more unlabeled nodes (i.e., motif) is high. The reason is that the distances are progressively shorter as we follow paths from the labeled nodes/subsequences towards the sink. Since every node leads to a sink, we do not consider sinks with support $P = 1$. We only label sinks with $P \geq 2$ and, all nodes on paths from the labeled nodes to that sink. The enclosed region in Figure 2(e) shows the labeled nodes by SeiSMo. The advantage of this method is that there is no domain dependent parameter to tune. One may simply perform more strict motif discovery by choosing a higher threshold of support, e.g. $P \geq 3$. Note that the value of P is a design choice for the practitioners and not a parameter to the algorithm.

Approved for public release; distribution is unlimited.

1.3.2 Related Work

Searching for the nearest neighbor of a query subsequence is done under Euclidean [44] or Dynamic Time Warping [53] distance. Both of these techniques are efficient and suitable for our algorithmic optimizations. SeiSMo is not limited to any specific distance measure, moreover an ensembling effect is observed when we combine outputs from SeiSMo using Euclidean and DTW distances.

Existing time series motif discovery algorithms are all unsupervised [42][40][39], focusing on various operational aspects of the tasks such as exact motifs [42], online motifs [40], variable length motifs [39], multi-dimensional motifs [80] and rare motifs [10]. Domain knowledge has been incorporated in motif discovery in the form of annotation vectors [16]; however, such annotation denotes preferred regions of the time series to guide the search for motifs, as opposed to providing instances/occurrences of a motif.

Semi-supervised approaches have been developed for time series classification extensively [72][14]. Classification tasks propagate labels to all unlabeled instances of the training data, while semi-supervised motif discovery algorithms are not required to label all subsequences of a time series. Time series classification, in general, works on a set of independent time series. SeiSMo works on subsequences of a long time series.

The proposed method is very relevant to classic density based techniques for unsupervised clustering (e.g., DBSCAN [20]) and outlier detection (e.g., Local Outlier Factor [11]). However, the concept of density connectedness has never been applied to motif discovery. To be more specific, SeiSMo has prefixed the two parameters in those algorithms, $minPts(= 1)$ and $\epsilon(= \infty)$, that most density based techniques require tuning.

NNGs on independent data points have been studied in computational geometry for decades [52]. However, our problem definition requires careful handling of overlapping subsequences in a time series to avoid trivial matches, and our optimization techniques greatly improve the construction of NNG on high dimensional data.

One important point of distinction is worth noting. Motif discovery from time series data is fundamentally different from motif discovery from discrete sequences such as DNA or protein sequences. To elaborate on the distinction, consider the query matching problem. Exactly matching a query string takes $O(n)$ time using the KMP (Knuth-Morris-Pratt) algorithm. Exact matching is not defined for real sequences, while approximate matching takes at least $O(n \log n)$ time [44]. Similarly, motif discovery from time series data faces several challenges because of normalization, trivial matches and diverse distance functions, while motif discovery from discrete sequence data enjoys simple match/no-match relationships between observations.

A recent practice in Seismology is to apply machine learning algorithms to learn from a vast amount of labeled data collected over many decades. As shown later in this report, SeiSMo can detect events that well-trained deep models cannot detect[50]. For the sake of argument, if we assume the nearest neighbor search for all subsequences can be modeled by a convolutional and a pooling layer, the recursive search for the nearest neighbors in SeiSMo would need a variable number of layers. With a sufficiently large number of layers and units in each layer, it should be possible to model SeiSMo, however, that is beyond the scope of this report.

Approved for public release; distribution is unlimited.

1.3.3 SeiSMo: Semi-supervised Motif

In Algorithm 1, we describe SeiSMo. The algorithm requires the positions of the known/seed subsequences in the time series as input. The output of the algorithm is a set of positions of newly detected events. We assume there is a maximum length for all the given and output occurrences of motifs. Such a length, that can contain most events, can be identified in most domains. For example, in local and near regional seismology, a 10-second window is generally enough for mine blasts and 20-second window is good for earthquakes.

The algorithm computes the nearest neighbors (NN), recursively, from each seed subsequence (Line 3-9). The iterative search ends when the current subsequence and the nearest of the nearest neighbor subsequence are more than O_t overlapped. Thus, we identify sinks using overlapping subsequences, as opposed to matching the exact locations of a subsequence. This is a necessary deviation from an exact location-based sink identification. To explain why, consider $S_{1,100}$. If the nearest neighbor of $S_{1,100}$ is $S_{501,100}$ and the nearest neighbor of $S_{501,100}$ is $S_{2,100}$, the algorithm will terminate iterating at Line 5, which would not be the case if the algorithm checked for $NN(NN(currentNode) = currentNode$ in Line 5.

Since the algorithm stops on a condition that uses overlap between subsequences, our algorithm is dependent on the order of the data, as many other data mining algorithms are [54][41]. In addition, in SeiSMo, several sinks with large overlaps are considered as one sink. All of the sinks reachable from all of the seed events are collected in a list, $SinkList$.

The second half of the algorithm counts the *support* of each sink (Line 10-13). Again, two sinks, (u, v) and (p, q), are considered the same or equivalent (i.e. $(u, v) \equiv (p, q)$) if overlap between p and u, and q and v, both are more than O_t, without losing generality. The sink equivalence relation has the following properties.

- If (u, v) is a sink, then u and v are guaranteed to be non-overlapping by the NN operation. The order of u and v can be swapped without loosing generality.
- sink equivalence is not transitive; $(u, v) \equiv (p, q)$ and $(u, v) \equiv (s, t)$ do not entail $(u, v) \equiv (s, t)$.
- Sink equivalence is symmetric. $(u, v) \equiv (p, q)$ entails $(p, q) \equiv (u, v)$.

Since we count support for one sink at a time, the lack of transitivity does not impact the counting process because of the symmetric property. Once high enough support is gathered, all nodes on the path from any seed event to the highly supported sink are added to the set of $newEvents$.

In the final stage, SeiSMo performs a **deduplication** operation to ensure non-overlapping events are output (Line 14). The deduplication operation is a self-join of the set of $newEvents$ on overlap of more than O_t. One may consider a more restrictive overlap threshold to limit the number of output patterns, however, we use the same overlap threshold O_t for consistency.

The Algorithm 1 does not require the seed events to be identical in length. If seeds are of different lengths, their chains of nearest neighbors will be of different lengths. When counting support, the algorithm considers O_t along with the length of the longer time series. More precisely, $(u, v) \equiv (p, q)$ if overlap between p and u, and q and v, both are more than $\lceil O_t m \rceil$, where m is the maximum of the lengths of u, v, p and q.

Approved for public release; distribution is unlimited.

Algorithm 1 $SeiSMo(TS, SP)$

Require: $TS \leftarrow$ a continuous time-series, $SP \leftarrow$ positions of seed events
Ensure: Output of newly detected seismic events from the time-series TS using the given seeds SP

1: **for** every seed $s \in SP$ **do**
2: $currentNode \leftarrow s$
3: $neighborNode \leftarrow \text{NN}(currentNode)$
4: $neighborOfNeighbor \leftarrow \text{NN}(neighborNode)$
5: **while** Overlap between $neighborOfNeighbor$ and $currentNode < O_t$ **do**
6: $currentNode \leftarrow neighborNode$
7: $neighborNode \leftarrow neighborOfNeighbor$
8: $neighborOfNeighbor \leftarrow \text{NN}(neighborNode)$
9: Insert $(currentNode, neighborNode)$ in the $SinkList$
10: **for** each $sink$ in $SinkList$ **do**
11: Count $support$ of $sink$
12: **if** $support$ of $sink \geq 2$ **then**
13: $newEvents \leftarrow newEvents \cup$ Nodes from all seeds to $sink$
14: deduplicate $newEvents$ using O_t
15: **return** $newEvents$

Optimizations to SeiSMo

The most time critical part of SeiSMo is searching for the nearest neighbors in an iterative manner. The properties of NNGs allow us to develop two optimization techniques to speed up the search process.

Recursive Search Initialization: The nearest neighbor search algorithm [53] uses an initial *best-so-far* value to begin the search. If no prior knowledge exists, *best-so-far* is set to ∞. Since the distances along a path on NNGs are always non-increasing, we can initialize the search in Line 8 by the last nearest neighbor distance on the path. More precisely, $neighborOfNeighbor \leftarrow NN(neighborNode, d(currentNode, neighborNode))$ This helps the early-abandoning process greatly to stop calculating unpromising distances.

Path Pruning: The in-degree of a node on NNG can be zero or more; however, the out-degree of a node is exactly one. If a nearest neighbor discovered in Line 8 is a repeated discovery; we can prune the path by breaking the loop in Line 5. At the implementation level, we keep a hash of all visited nodes on the NNG and check for repeated visits in constant time. This approach ensures no node is visited more than once. Note that SeiSMo must increase the support of the sink the path was heading to before pruning any path. This requires some additional bookkeeping to maintain the support count for each sink dynamically.

Complexity of SeiSMo

Time complexity of SeiSMo algorithm is dominated by the iterative nearest neighbor search. If we use DTW distance, the cost of an NN search is $O(nm^2)$ where n is the length of the time series and m is the length of the motifs. The number of NN searches is proportional

Approved for public release; distribution is unlimited.

to the number of seeds S and lengths of the paths from seed to sink, P. Hence, the total complexity is $O(SPnm^2)$. In most applications, S and P are on the order of tens, while n is in millions. Hence, the complexity of the algorithm is largely dominated by the length of the time series.

Our pruning approaches do not reduce the asymptotic complexity of the algorithm. However, recursive search initialization reduces the effective value of n and path pruning reduces the effective value of P. The memory requirement for these optimization techniques is proportional to the number of sinks and average path size. These quantities are much smaller than the data size (n), hence, the algorithm is effectively in-situ.

1.4 Results and Discussion: Signal Discovery

1.4.1 Experimental Evaluation

All our experiments are reproducible. The code, data, spreadsheet and figures are available in [4]. We use six datasets for experiments including four real, one synthetic, and one simulated datasets:

- ***Synthetic*** white noise with implanted sinusoids containing 100,000 observations.
- ***California*** seismograms from station NC.CCOB between January 8, 2011 (00:00:00) and January 15, 2011 (00:00:00). The dataset contains 12,096,000 observations recorded at the rate of 20 samples per second.
- ***Oklahoma*** seismograms from station GS.OK029 between April 1, 2014 (00:00:00) and April 8, 2014 (00:00:00). The dataset contains 12,096,000 observations recorded at the rate of 20 samples per second.
- ***Wyoming*** seismograms from station ZH.RPCE between July 19, 2010 (00:00:00) and July 21, 2010 (00:00:00). The dataset contains 3,456,000 observations recorded at the rate of 20 samples per second.
- ***LabQuake*** or *LabQ* seismic data [75] generated by a frictional experiment in the Pennsylvania State University Rock and Sediment Mechanics Laboratory. This dataset is very well labeled and specifically suited for precision and recall analysis because of the known ground truth. Also, this data is suited for testing precision and recall on different SNR values. The dataset is publicly available in [1]. The dataset contains 125,000 observations.
- ***Southern California*** seismograms from station CI.DRE between April 3, 2010 (00:00:00) and April 10, 2010 (00:00:00). The dataset contains 12,096,000 observations recorded at the rate of 20 samples per second.

We define **Precision** and **Recall** for motif discovery in the following way. Let us assume the number of occurrences of a motif is N. Let us assume a motif discovery algorithm finds Q occurrences of the motif and, P of them are from the set of known occurrences. The

Approved for public release; distribution is unlimited.

$precision$ is $\frac{P}{Q}$ and $recall$ is $\frac{P}{N}$.
Signal-to-noise-ratio (SNR) is a measure of signal strength relatiave to background noise. Our objective is to empirically evaluate precision and recall of our method using the same seismogram modified in such a way that each version of the seismogram has a different SNR value.

Precision is our primary metric of comparison because false detection costs valuable human-hours for further filtering. In contrast, there is no ground truth about the total number of undetected events in a seismogram. Therefore, recall calculation is not practical on real world datasets.

Sanity Check

We generate a synthetic dataset by implanting sinusoids of length 200 in a series of 100K white noise samples. The variances of the sinusoids are 10% of the variances of the noise. We use a random 40% of the sinusoids as seed events. We vary the number of implants from 64 to 1024 by iterative doubling, and measure the precision and recall of motif detection. Our precision is always 100% on synthetic data. This is reassuring, and not surprising, as motif discovery algorithms are precision-focused by construction. SeiSMo achieved a recall rate of 35.02% on average, with 3.51% standard deviation. To improve the recall rate, one may relax the algorithm further by using a minimum support of one. However, on real datasets, the impact on precision can be significant.

Comparison to Fingerprint And Similarity Thresholding (FAST)

We compare SeiSMo with a naive semi-supervised motif discovery algorithm based on simi-larity search within a radius (RS). We set the search radius equivalent to 80% of Pearson's correlation coefficient [81]. We compare the methods on two synthetic datasets: *Synthetic* and *LabQuake* datasets. Both of these datasets have manually inserted events enabling us to calculate precision and recall. The seeds are all of length 400 for this experiment.

Quantitative comparison to similarity search within a radius (RS) is unfair in the sense that there is a difference in degrees of freedom. RS can perform anywhere between very good to the worst. In contrast, SeiSMo has no degrees of freedom, it is exact and determin-istic. The output produced by SeiSMo is robust and significant. Similarity search within a threshold is very useful in interactive data mining, however, for the purpose of motif (i.e. hidden event) discovery SeiSMo off loads the human involvement in the mining process by eliminating the parameter.

On simple sinusoidal synthetic data, both RS and SeiSMo achieve perfect precision, while RS achieves higher recall for a search threshold of 0.87. Note that sinusoidal data is the best case for motif discovery. In contrast, on *LabQuake* data, SeiSMo performs much better than radial search in precision and recall. Although simulated, *LabQuake* events contain similar noise levels as real earthquake signals contain.

Comparison to Methods in Seismology

Although SeiSMo is more robust than naive radial search, we must compare SeiSMo with existing earthquake detection algorithms from the seismology community. Two of our datasets,

Approved for public release; distribution is unlimited.

Table 1. Precision and recall of different methods on simulated and synthetic datasets.

Dataset	LabQuake	Synthetic
Length	125K	100K
Number of Seed Events	82	59
Total Number of Events	130	150
SeiSMo- New Detections	33	36
RS - New Detections	0	88
SeiSMo- Precision	96.97%	100%
RS - Precision	0%	100%
SeiSMo- Recall	66.67%	39.56%
RS - Recall	0%	96.70%

We show these because there is no ground truth for seismic datasets, as the complete set of underground events is generally not known.

California and *Oklahoma*, are taken from existing work on hidden event discovery. We compare to both of them to check if SeiSMo adds any novel detections. The results are shown in Figure 5. All of our novel detections are manually verified.

In *California* data, we observe that SeiSMo adds few more (18) unique events to the events detected by an existing technique (ConvnetQuake), while radial search adds no value. We show the events in the subsequent sections, however, the fact that SeiSMo is precisely detecting more events in two independent real datasets from two states is showing the gen-eralizability of our technique.

In *Oklahoma* data, we observe the interesting fact that all methods are detecting a large number of unique events. This suggests that the dataset is rich in number of events and, the methods are capturing unique aspects of the events so should be able to work collaboratively.

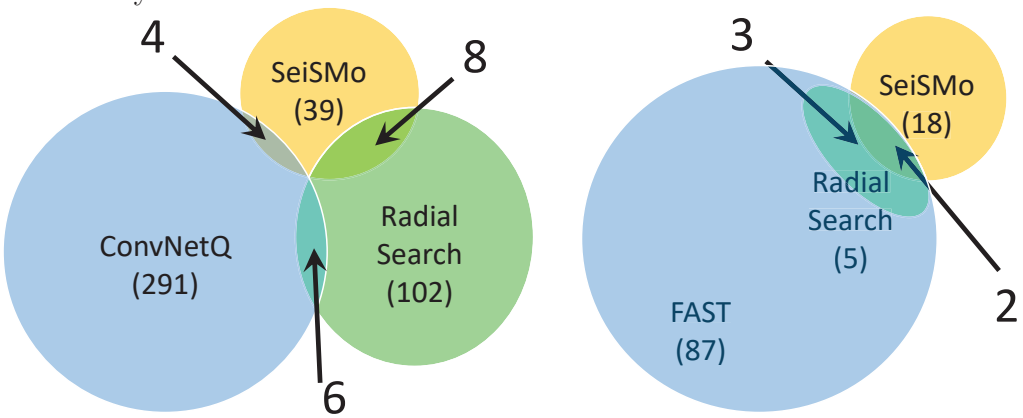

Figure 5. (left) Comparison to ConvNetQuake [50] and similarity search within a radius (RS) on Oklahoma dataset. (right) Comparison to FAST [82] and similarity search within a radius (RS) on California dataset.

Comparison to Semi-supervised Data Mining Method

Historically, semi-supervised methods have been developed to perform supervised learning in the absence of labeling. SeiSMo focuses on exploiting labels in performing unsupervised

Approved for public release; distribution is unlimited.

motif discovery. Although this key difference makes SeiSMo unique, we consider an existing semi-supervised learning algorithm for time series data, DTW-D [14] to detect hidden events.

Our approach is to train a semi-supervised classifier on the seed events (i.e. control) and on a mix of seed and SeiSMo events (i.e. treatment). In both of the scenarios, we consider the seismic events as positive instances and add an equal number of non-event segments as negative instances to balance the datasets. If the SeiSMo events are all detectable by DTW-D classifier, we expect to see an increase in classifier performance. If the SeiSMo events contain same amount of confusion as the seed events, the classifier accuracy should stay the same.

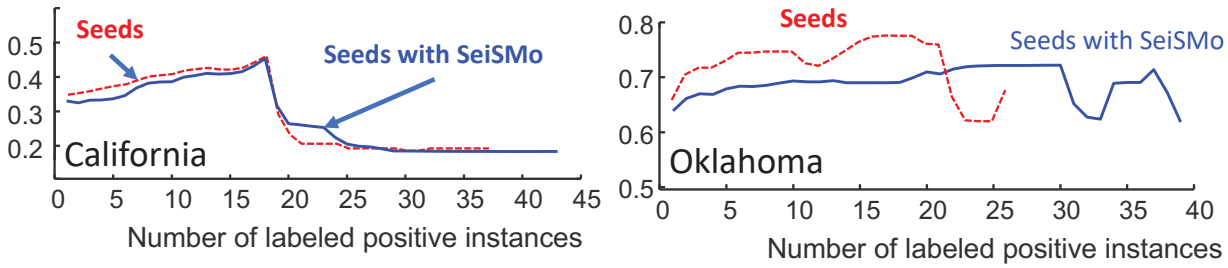

Figure 6. Applying the semi-supervised classification (DTW-D) algorithm to detect new events.

In Figure 6, we show the results on two of our datasets: California and Oklahoma. We see no significant difference between control and treatment on California dataset, which ideally implies that DTW-D classifier finds SeiSMo events very similar to the seed events. However, the accuracy of the DTW-D classifier is at most equal to the default accuracy (50%) on California dataset. Therefore, no conclusion can be made for this dataset. We see a significant decrease in accuracy once we add SeiSMo events to the seed events in the training dataset (Figure 6(right)). We conclude that DTW-D failed to learn about the unique SeiSMo events.

Note that DTW-D is not an event discovery algorithm. Hence, our demonstration of DTW-D classifier failing in event detection should not undermine its goodness as a classifier.

Efficiency

SeiSMo recursively searches for the nearest neighbors. The number of searches is generally more than the number of searches in radial search based methods. In Figure 7 (left), we show the execution time in seconds as we increase the data size up to a million observations. We use the synthetic data with 14 seeds. The optimization techniques speed up the computations by roughly a factor of two. The growth in execution time is roughly linear, suggesting scalability to even hundreds of millions of observations.

Unsupervised motif discovery algorithms using the recent Matrix Profile technique [81] requires an order of magnitude more time to identify the seed motif and expand to more occurrences. To find motifs in a 10^6 long time series, Matrix Profile runs for a day, while SeiSMo finishes in 25 seconds.

Execution speed grows with the number of seed events. Figure 7(right) shows the change in execution time as we implant more events and use more seeds. Similarity Search within a

Approved for public release; distribution is unlimited.

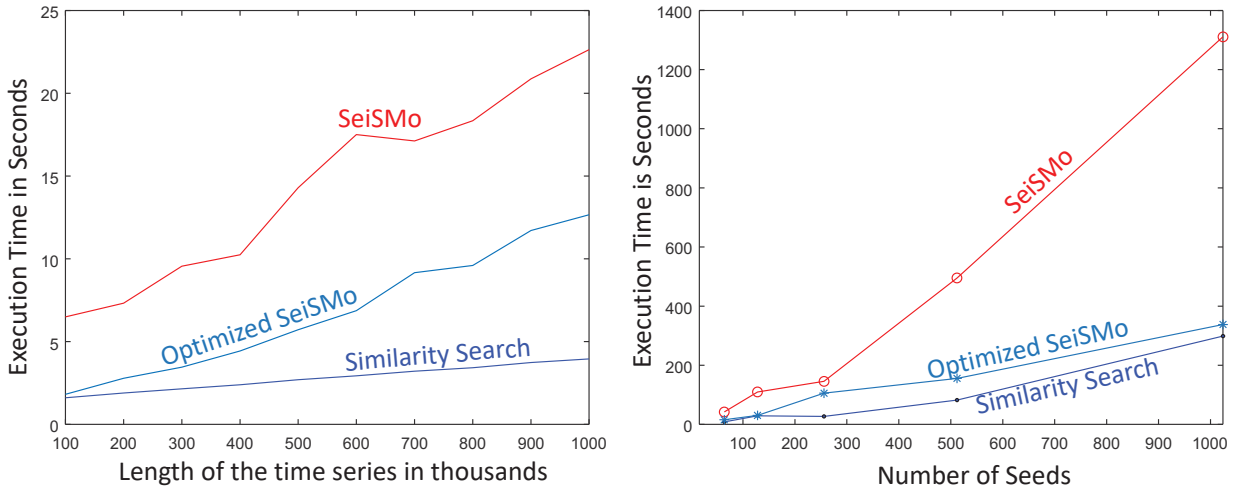

Figure 7. (left) Linear scalability and speedup by Optimization techniques. (right) Scalability with respect to number of seeds.

radius is generally faster, however, the optimized SeiSMo grows almost linearly with respect to n, without any sensitive parameter such as the radius.

Parameter Sensitivity

We have one parameter in SeiSMo. The support of the sink, P, is set to 2 for all of the experiments. However, to develop a reasoning about this parameter, we perform an experiment by varying the parameter P, and recording the precision and recall. In Figure 8 (right), we show the result. For both synthetic and *LabQuake* (LabQ) data, precision is generally insensitive to increasing support, while recall decreases with increasing support. This is intuitive because, increasing support restricts SeiSMo to detect only patterns with high confidence. Since precision is the key metric for automated signal detection, we claim insensitivity to the parameter P.

To evaluate the effect of noise, we incrementally add noise to *LabQuake* data and measure precision and recall. The results are shown in Figure 8(left). We see a drop in precision and recall when SNR is decreased. We calculate the SNR by taking the ratio of the absolute amplitudes of the signal and and noise.

$$SNR = \frac{mean\ of\ absolute\ signal\ amplitude}{mean\ of\ absolute\ noise\ amplitude} \tag{2}$$

Effect of Distance Measure

We find that Euclidean and Dynamic Time Warping (DTW) distances can produce unique hidden events from the same data. Hence, we suggest parallel execution of SeiSMo under all available distance measures and taking a union of the outputs as the final output. We empirically observe, on average in all our experiments, DTW contributes 60% to the newly detected events, while Euclidean distance contributes 40%. We hypothesize that more distance mea-sures will add more detections at a diminishing rate. We leave it for future work to validate the hypothesis.

Approved for public release; distribution is unlimited.

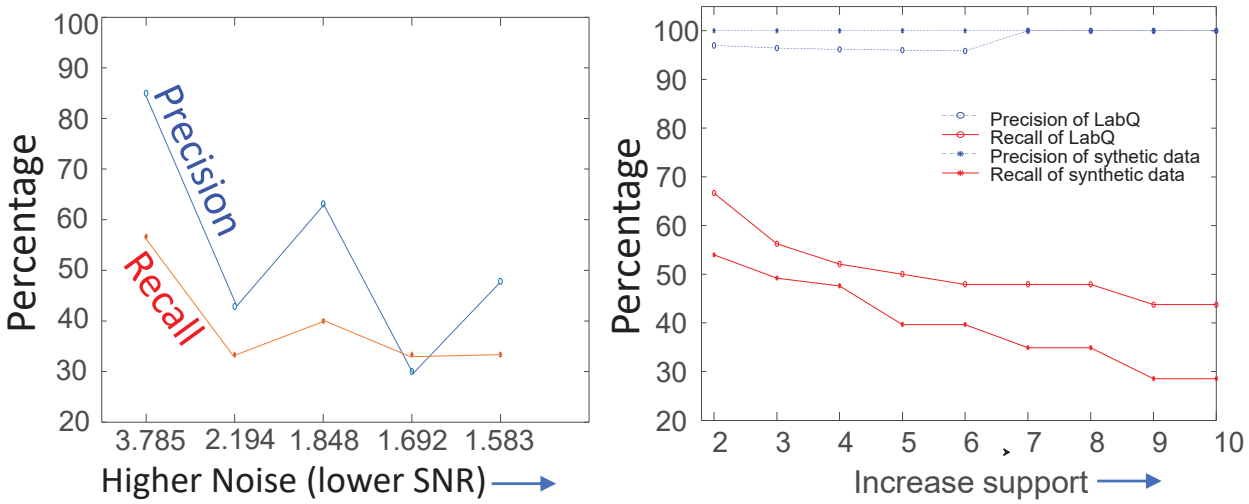

Figure 8. Parameter sensitivity of SeiSMo. (left) Precision and recall graph of SeiSMo with different noise level. Both precision and recall decrease when noise level is increased. (right) Precision and recall for varying support count, P.

Evaluation

It is very important to ensure that the events we detect are not caused by any electrical/mechanical malfunction; or by any minor incident (i.e. a vehicle passing by). We can ensure this by examining the signals from all three (horizontal North-South, horizontal East-West, and vertical) components, and from all of the components of nearby stations. Therefore, once detected, we evaluate if the detected signal is a real seismic event by visual examination of the other channels/components of the same stations and all channels of nearby stations. If more than one station records the same event within a reasonable delay, we confirm the signal as a real event. To illustrate, in Figure 9, we show the detected signal in the HH1 channel of OK029 station. We see that the other two channels, HH2 and HHZ, have strong signals. For further validation, we inspect a nearby station, in this case OK027, in the same time window. We see the presence of strong signals in all channels of OK027, which confirm that the detected signal represents a real event. Note that the signals in OK027 are consistently ahead of the signals in OK029 for all channels. This further validates that the seismic source is closer to OK029.

The challenges associated with such a visual validation technique are the possible absence of all the channels and absence of a nearby station. If none are available, we pessimistically consider the signal as a false detection.

1.4.2 Natural Seismic Events in California

Northern California Seismic Network

We present a motivating case study to demonstrate the utility of semi-supervised motif discovery from time series data. Seismic events are observable in seismographs, especially when the events are of high magnitude. However, low magnitude events are more frequent than higher ones, and often escape expert attention. Such events are not found in IRIS catalogs, although they are important for various geo-scientific analyses.

Approved for public release; distribution is unlimited.

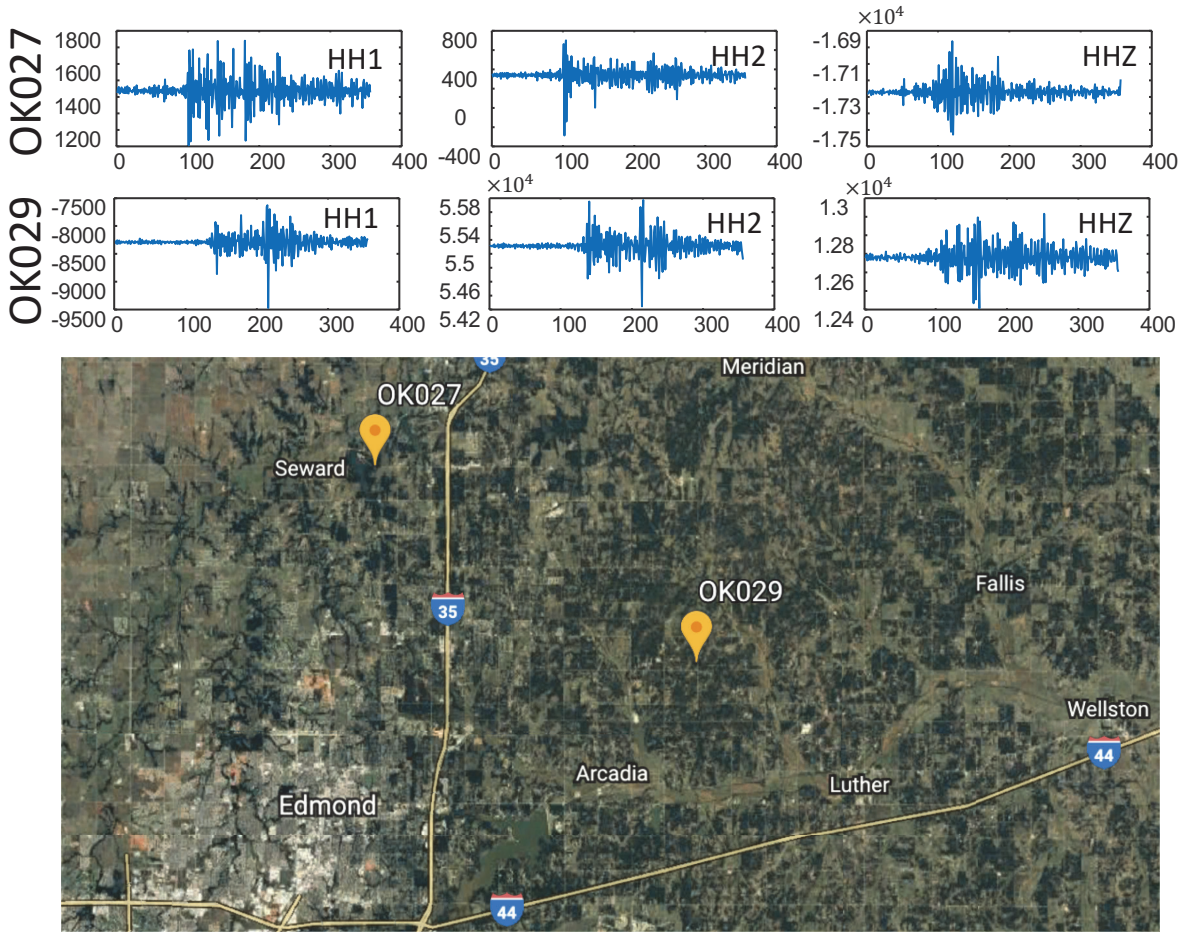

Figure 9: Example of validation using additional channels from the same and an additional station. *SeiSMo detects the signal at OK029-HH1. In this figure, We have other components (HH2, HH3) of OK029 station on the first row. Moreover, we picked another station, OK027 which is 13 miles from OK029 (exact position is shown in the map). We have shown waveforms from all three components from OK027 in the second raw. We observe the presence of the event in all six waveforms, thus, validating the detection.*

We apply our SeiSMo algorithm to discover hidden seismic events using existing cataloged events as labeled data. The Calaveras Fault in central California is known to have repeating earthquakes [62]. We have collected seven days (from January 8, 2011 to January 15, 2011) of single channel (horizontal north-south component) seismic data from a station (NC.CCOB) in the Northern California Seismic Network from the NCEDC [3]. We have collected 24 cataloged events that originated within 100 miles of the station. FAST [82] is a hash based method, which, in principle, is a radial search technique under a specific distance measure. We have augmented the catalog events with 87 events detected by FAST to create the seed set. Seismograms for each event were 20 seconds long, which is good enough to fit a single event and had a sampling rate of 20 samples per second. We applied a 4- to 10-Hz bandpass filter to remove any unwanted

Approved for public release; distribution is unlimited.

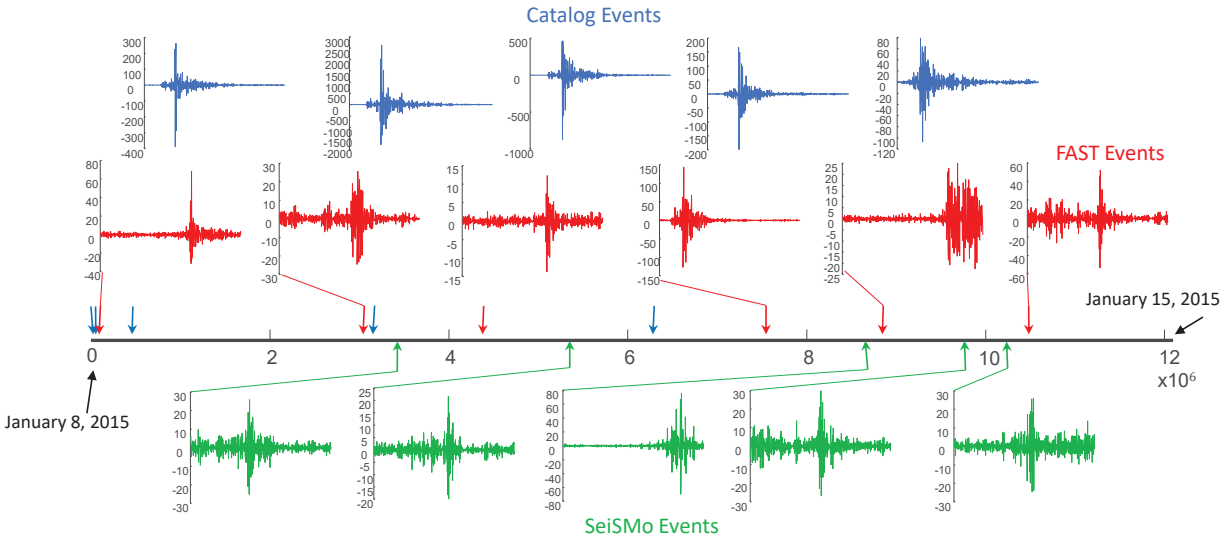

Figure 10. Randomly picked results of California dataset. *Top row of events is cataloged. Middle row of events is detected by FAST [82]. Bottom row of events is detected by SeiSMo.*

noise. We have discovered 18 new events that have neither been cataloged in IRIS nor been detected by FAST. We use only one channel (EHN) of data for detection. The events we have discovered are all confirmed by manually checking the other two channels as well as other nearby stations.

In Figure 10, we show some random events for further discussion. All the events show the presence of significant signals. The catalog events are generally high magnitude events. FAST detects higher magnitude events than those SeiSMo detects. Northern California is a very active region. Automated seismic signal discovery is the first step towards data driven understanding of seismic activity over time and space. Thus, SeiSMo contributes significantly towards that objective.

Southern California Seismic Network (SCSN)

SCSN recorded a good number of seismic activities after a 7.2 magnitude earthquake at Baja California, Mexico in April, 2010. We apply SeiSMo algo-rithm to discover hidden seismic events in a dataset collected from the SCSN using existing cataloged events as labeled data. We have collected seven days (from April 3, 2010 (00:00:00) to April 10, 2010 (00:00:00)) of single channel (vertical component, BHZ) seismic data from a station (CI.DRE) in the SCSN from the Southern California Earthquake Data Center (SCEDC) [5]. Unlike the *Northern California* experiment in the previous section, we perform a study on the effect of event magnitudes in the seed set using this dataset.

We have collected 60 cataloged events recorded at this station with a range of magnitudes from 0.3 to 7.2 in Richter scale. We order the events in increasing order of magnitudes and take sets of 30 consecutive seeds from the ordered list of events. For each of these sets of seeds, we run SeiSMo to detect new events and chart the performance in Table 2. We see an increase in detection performance as the magnitude of the seed events increases. High magnitude events show better contrast with respect to the baselines around them compared

Approved for public release; distribution is unlimited.

to low magnitude events. This corroborates to the observations in Table 2.

Table 2: Increasing performance with seed quality.

Maximum Magnitude	4.10	4.20	4.40	7.20
Minimum Magnitude	0.30	1.66	2.50	4.10
Number of Detection	2	17	30	35

1.4.3 Induced Seismic Events in Oklahoma

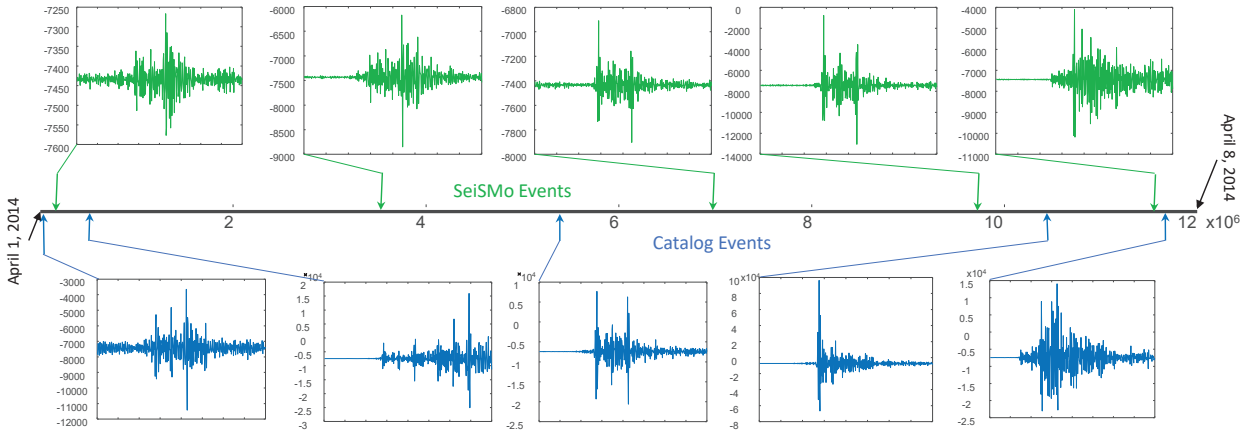

Figure 11. Results of Oklahoma dataset shows that, SeiSMo detects lower magnitude events than those in the IRIS catalog.

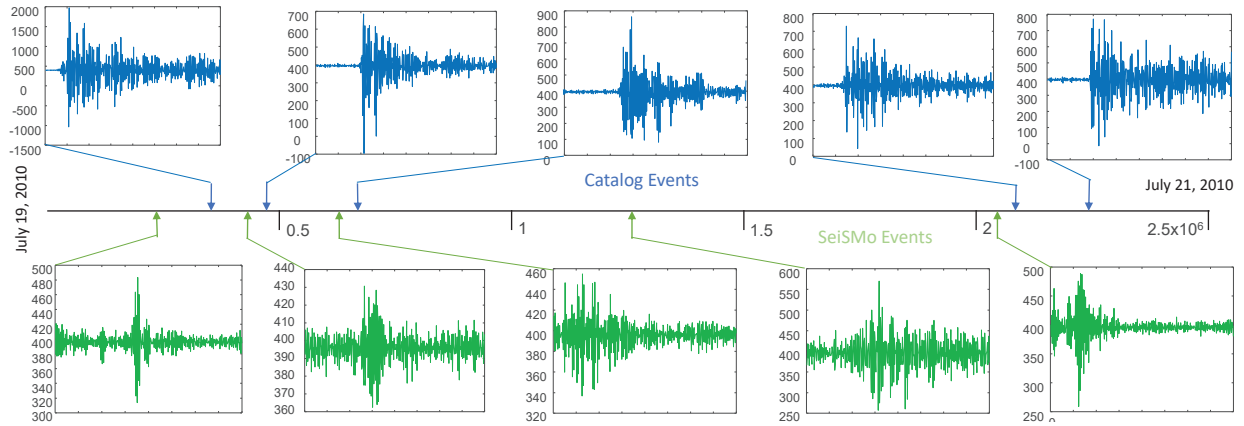

Figure 12. Some randomly picked events from the Wyoming dataset results. *Top row of events is known. Bottom row of events is detected by SeiSMo.*

Human-induced seismicity includes seismic activities that are not directly initiated by humans (e.g. nuclear tests), but rather are indirect outcomes of human activity, such as seismic activity due to waste water injection [71] [73]. In ConvNetQuake [50], authors have identified a recent increase in induced seismicity in Guthrie county of central Oklahoma.

We have applied SeiSMo to a single channel (HH1) dataset from the same station and for the same time duration of 7 days (April 1, 2014 (00:00:00) to April 8, 2014 (00:00:00)).

Approved for public release; distribution is unlimited.

We have collected 77 seed seismograms from IRIS [2]; each of the those was 20 seconds long with a sampling rate of 20 samples per second, just like the previous experiment. Also, we applied a 4- to 10-Hz bandpass filter to remove any unwanted noise. We run SeiSMo on this dataset and using these seeds we detect 39 more events. All of these 39 events are confirmed by manually checking all of the channels and few neighboring stations. Figure 11 shows five random cataloged and newly detected events on a time line. The cataloged events are higher in magnitude than those detected by SeiSMo.

1.4.4 Seismicity due to Controlled Explosions in Wyoming

The mining industry in Wyoming uses ripple fired mining technology [74], which creates a very low energy yet significant seismic footprint at nearby stations across the Bighorn Mountain [79] [46]. We collected a dataset from the experiment of [47], which contains 24 active source borehole events that were labeled by the authors. We used 16 borehole shot events over two consecutive days, on July 19 and July 20, 2010. SeiSMo identified 12 new signals (that do not include the eight remaining borehole shots that happened outside of these two days). We manually confirm that the signals *look* like arrival waveforms. Randomly picked SeiSMo detections and seeds are shown in Figure 12.

Consistent with the previous cases, the detected events are significantly lower in magnitude compared to catalog events. This suggest that SeiSMo is capable of discovering hidden low-magnitude seismic events.

1.5 Conclusion: Signal Discovery

We propose a semi-supervised motif discovery algorithm to discover motifs or hidden events in a time series when a small number of arbitrarily similar events are known a priori. Using six different datasets, We show that our method is more accurate and robust than a naive similarity search on large datasets. We apply the algorithm on four seismic datasets and discover novel seismic events that were unknown to experts. All of our code, data, and results used in this report are available [4] for reproducibility.

SeiSMo is good at spotting unique low magnitude events, which were not cataloged because analysts were not certain about their origin and characteristics. We plan on analyzing SeiSMo events to produce high quality information towards a fully automated cataloging system.

Approved for public release; distribution is unlimited.

Chapter 2

Filtered Lagged Correlation (FilCorr): Filtering, Correlating and Visualizing Seismic Waves in Real Time

2.1 Summary: Signal Monitoring

Monitoring systems have hundreds or thousands of distributed sensors gathering and transmitting real-time streaming data. The early detection of events in these systems, such as an earthquake in a seismic monitoring system, is the base for essential tasks as warning generations. To detect such events it is usual to compute pairwise correlation across the disparate signals recorded by the sensors. Since the data sources (e.g., sensors) are spatially separated, it is essential to consider the lagged correlation between the signals. Besides, many applications require processing of a specific band of frequencies depending on the event's type, demanding a pre-processing step of filtering before computing correlations. Due to the high-speed of data generation and large number of sensors in these systems, the operations of filtering and lagged cross-correlation need to be efficient to provide real-time responses without data losses. In this part of the project, we propose a technique named FilCorr that efficiently computes both operations in one single step. We achieve an order of magnitude speed-up by maintaining frequency transforms over sliding windows. Our method is exact, devoid of sensitive parameters, and easily parallelizable. Besides our algorithm, we also provide a publicly available real-time system named Seisviz that employs FilCorr in its core mechanism for monitoring a seismometer network. We demonstrate that our technique is suitable for several monitoring applications as seismic signal monitoring, motion monitoring, and neural activity monitoring.

2.2 Introduction: Signal Monitoring

Large monitoring systems such as a network of seismic stations or forest fire detection systems typically have hundreds of distributed sensors gathering and transmitting real-time and streaming data. An event in these systems, like an earthquake or an abnormal higher temperature, creates dynamic responses at these sensors. The responses can be arbitrarily lagged because of sensors' spatial separation and be limited to a specific band of frequencies depending on the type of event. For these systems, real-time event detection is an essen-

Approved for public release; distribution is unlimited.

tial task for decision making or alert generation. *We demonstrate an algorithm to correlate streaming data generated from distributed sensors in real-time in order to detect events.*

A concrete application where our algorithm can be employed is seismic monitoring. In this application, when an earthquake happens, seismometers (i.e., seismic sensors) across the region of the earthquake observe the wave at varying times for varying duration, while the signals recorded at these sensors are often correlated. For better understanding, in Figure 13, we show a magnitude 3.1 event in Yellowstone on May 29, 2020. Three waveforms recorded at three stations show a high correlation when the lag due to propagation delay is considered.

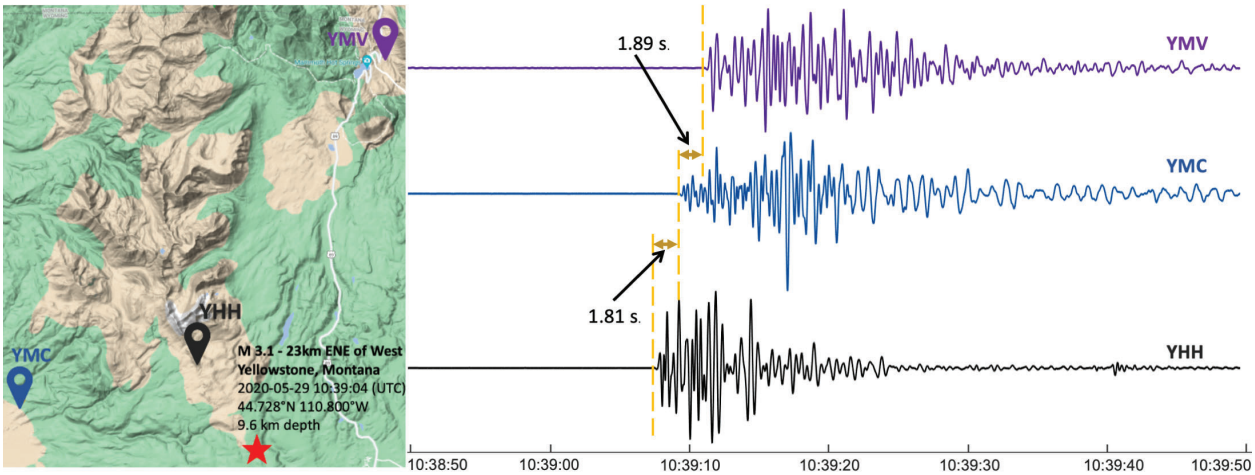

Figure 13: *(left)* A M3.1 earthquake in Yellowstone, on May-29, 2020. *(right)* The signals recorded at three different stations at different times due to the spatial separation of sensors. Data collected from IRIS [2].

In addition, filtering is a mandatory operation in seismic signal processing to remove undesired noise from data and extract the right frequencies for desired events. For many applications, mainly those involving digital signal processing, certain properties of data are made explicit when the signal is represented in the frequency domain [67]. Thus, filtering is also an essential task to extract descriptive features for machine learning models.

In a monitoring system that consumes streaming data from hundreds of sensors, computing lagged correlation (or asynchronous correlation) can be challenging because of the fast data-rate, a large number of stations, and the necessity for accurate correlation values. Although the problem has been studied for decades, none of the existing methods such as BRAID [60], COLR-tree [7], or StatStream [86]), can monitor one hundred seismometers at a 10Hz rate on a single workstation, a usual setting for many systems. The main reason for the failure of these methods is the data-dependent pruning, projection, or indexing technique. Seismic traces are mostly white noise (except when events happen), stressing the algorithms to fall behind the stream quickly. In contrast, none of these algorithms are amenable to data pre-processing requirements, such as filtering. Hence, pre-processing must always be done before correlation computation, resulting in a loss of efficiency.

We propose an algorithm, **FilCorr**, to merge filtering and lagged correlation computation in order to extract *data-independent* efficiency. Our algorithm efficiently maintains frequency information over the stream and calculates correlation in the frequency domain. The technique is *exact*, *devoid of sensitive parameters*, and *easily*

Approved for public release; distribution is unlimited.

parallelizable. This report gives a comprehensive technical description of its properties, implementation, and performance on various real-world problems. We experimentally show that our technique is suitable for monitoring applications where the lagged correlation of filtered time series is required, such as seismic event monitoring, motion monitoring, and neural activity monitoring. Specifically, for the seismic event application, we developed a real-time system named Seisviz that employs FilCorr in its core mechanism for monitoring a seismometer network in the Yellowstone region. Our implementation can monitor one hundred seismic stations at a 10Hz rate without any delay. A demonstration of Seisviz capturing past earthquakes is available on our website [84], and the system for real-time monitoring is available at `www.seisviz.com`.

The main contributions of this work are summarized below:

- We demonstrate a working system to cross-correlate hundreds of high speed streams in real-time at 10Hz speed using a conventional workstation;
- We merge digital filters and correlation computation in one combined step to achieve time and space efficiency;
- We show three case studies where such high-speed lagged correlation helps detect events of interest.

2.3 Methods, Assumptions, and Procedures: Signal Monitoring

2.3.1 Background and Notation

We define *time series* as a sequence of observations, and they are in the form of real numbers measuring a quantity at a fixed sampling rate. A *streaming time series* is an unbounded sequence of observations generated at a fixed rate. We use s^x to represent a particular streaming time series x. We define the *basic window* as a continuous segment of a time series. A basic window of size m from time series s^x at time t is denoted by w_t^x which contains observations from $s^x[t-m+1]$ to $s^x[t]$. We can extract at most $n-m+1$ basic windows of length m from a long time series of length $n >> m$. Adjacent basic windows are not independent of each other, overlapping basic windows are trivially close in the high dimensional space.

Filtering: An essential pre-processing task for time series required by many applications, filters are most commonly used to remove undesirable components from a time series. There are many kinds of filters useful in various domains. All filters have response functions that are convolved with a signal to apply the filter. Such response functions contain relative weights for each frequency. In an analytical form, the weights are non-zero for all frequencies. However, practically, many weights are significantly smaller than the rest, allowing us to cut off and keep only the essential frequencies. Most applications employ a band that focuses on the needs of the application. For example, seismic monitoring uses up to 10Hz [63], and EEG monitoring uses a gamma activity band between 30Hz-50Hz [12]. In this work, we assume that the given filter band is a contiguous set of frequencies, and all frequencies are equally

Approved for public release; distribution is unlimited.

important. This is also known as a box filter or ideal bandpass filter. Our proposed method is extendable to a non-contiguous set of frequencies with varying weights, i.e., other types of filters. We define *frequency window* W_t^x that contains all the frequency components of the basic window w_t^x, and each $W_t^x[k]$ where $k = 0, 1, 2, \ldots, m-1$ are defined in Equation 3. Note $i = \sqrt{-1}$, $W_t^x[k]$ is a complex number.

$$W_t^x[k] = \sum_{j=0}^{m-1} w_t^x[j] e^{\frac{-i2\pi}{m}kj} \tag{3}$$

We use f^x in Hz to represent the sampling rate of the streaming time series s^x and the *filtered basic window* $w_t^{x\prime}$ to denote the filtered version of w_t^x. If we apply a box filter with a band from f_s to f_t in Hz ($0 < f_s \leq f_t$), then $w_t^{x\prime}$ can be derived from Equation 4:

$$w_t^{x\prime}[j] = \frac{1}{m}\Big(\sum_{k=\lfloor \frac{f_s}{f} m \rfloor}^{\lfloor \frac{f_t}{f} m \rfloor} W_t^x[k] e^{\frac{i2\pi}{m}kj} + \sum_{k=m-\lfloor \frac{f_t}{f} m \rfloor}^{m-\lfloor \frac{f_s}{f} m \rfloor} W_t^x[k] e^{\frac{i2\pi}{m}kj} \Big); \tag{4}$$

There is a corner case when m is even and $f_t = \frac{f}{2}$ then $w_t^{x\prime}[j]$ equals $\frac{1}{m}(\sum_{k=\lfloor mfs/f \rfloor}^{k=m-\lfloor mfs/f \rfloor} W_t^x[k] e^{\frac{i2\pi}{m}kj})$.

The Fourier transform of $w_t^{x\prime}$ will only contain non-zero components that correspond to frequency from f_s to f_t in W_t^x. If $f_s = 0$ and $f_t = \frac{f}{2}$ (Nyquist frequency), then $w_t^{x\prime}$ is identical to the basic window w_t^x.

Correlation computation in the frequency domain: If we are given two basic windows $w_{t_1}^x$ and $w_{t_2}^y$, Pearson's correlation coefficient between them is defined in the time domain as Equation 5:

$$corr_{t_1 t_2}^{xy} = \frac{\sum_{j=0}^{m-1} w_{t_1}^x[j] w_{t_2}^y[j] - m\mu(w_{t_1}^x)\mu(w_{t_2}^y)}{m\sigma(w_{t_1}^x)\sigma(w_{t_2}^y)} \tag{5}$$

To compute the correlation in the frequency domain, we can exploit Parseval's theorem [49, 45], which is expressed as Equation 6 for the discrete Fourier transformation:

$$\sum_{j=0}^{m-1} |w_t^x[j]|^2 = \frac{1}{m} \sum_{k=0}^{m-1} |W_t^x[k]|^2 \tag{6}$$

Based on Equation 6, we can compute Equation 5 in the frequency domain: $\mu(w_t^x)$ equals $\frac{W_t^x[0]}{m}$ since $W_t^x[0] = \sum w_t^x[j]$. $\sigma(w_t^x)$ equals $\sqrt{\frac{\sum w_t^x[j]^2}{m} - [\mu(w_t^x)]^2}$ in the time domain and the $\sum w_t^x[j]^2$ term can be computed in the frequency domain by directly applying Equation 6. Since the discrete Fourier transform is a linear operation, Parseval's theorem can also be expressed as Equation 7 with two signals.

$$\sum_{j=0}^{m-1} |w_t^x[j] - w_t^y[j]|^2 = \frac{1}{m} \sum_{k=0}^{m-1} |W_t^x[k] - W_t^y[k]|^2 \tag{7}$$

Based on Equation 7, We can get Equation 8 to compute $\sum_{j=0}^{m-1} w_t^x[j] w_t^y[j]$. Thus we show how to compute each term in Equation 5 in the frequency domain.

Approved for public release; distribution is unlimited.

$$\frac{1}{2m}\Big(\sum_{k=0}^{m-1}|W_t^x[k]|^2+\sum_{k=0}^{m-1}|W_t^y[k]|^2-\sum_{k=0}^{m-1}|W_t^x[k]-W_t^y[k]|^2\Big) \tag{8}$$

Intuition: *Computing correlation values in the frequency domain is more efficient when we have a bandpass filter.* Based on Equation 4, the computation of correlation in the frequency domain only takes time $O(B)$, where B is the bandwidth in a number of frequency components.

Problem Statement: Given N streaming time series, a frequency band (f_s, f_t), and a maximum allowable lag l, compute the Pearson's correlation coefficients for all pairs of streaming time series over a sliding window up to the given lag l.

We argue with the empirical case study, this problem is very practical. In most domains, filtering can get rid of unwanted signals to compute correlation on right signals; and a reasonable maximum lag always exists, beyond which no correlation coefficient is meaningful.

2.3.2 Related Work

Lagged correlation is a problem that has been studied for decades [86, 60, 7]. However, none of the existing methods consider a set of features required by modern applications that monitor hundred of sensors in real-time. Based on these requirements and FilCorr properties, we categorize the related works to our proposal according to four groups, as follows.

Lagged Correlation Computation: Computing lagged correlation, or cross-correlation, on offline data is a fundamental operation that benefits from the Fast Fourier Transform. Researchers have proposed an online algorithm for lagged correlation computation named BRAID [60]. However, the method samples frequency coefficients in a logarithmic manner to approximate the correlation value. Moreover, the method computes correlations over the entire stream, unlike our method that computes over a sliding window in real-time. One may consider offline indexing methods such as iSAX [65] and COLR-tree [7], for lagged correlation computation. However, such offline indexes suffer from many modification (insert or delete) operations over correlation computation.

Real-time Correlation Computation: Efficient all-pair correlation computation for streaming data is no longer an active research problem. StatStream [86] is a technique that exploits a few Fourier coefficients to prune improbable pairs quickly. ParCorr [78] performs random projections to similarly prune improbable pairs without redundancy due to the sliding window. Hardware-based techniques are often used for computation at MHz to GHz rate [26]. However, it is important to note that none of these methods consider lagged correlation after filtering.

Frequency Domain Correlation Computation: It is very common in many application areas to calculate correlations in the frequency domain. However, most works consider the offline nature of computation [43, 63].

Filtered Correlation Computation: In this category, our method is unique. To the best of our knowledge, no work exploits filtering operations to extract efficiency in correlation computation. It is somewhat surprising considering the widespread usage of filtering in processing real-time data captures. The novelty in our technique is that the speedup is not data-dependent, unlike any of the aforementioned work.

Approved for public release; distribution is unlimited.

In Table 3, we present a comparison of the main capabilities of FilCorr concerning state-of-the-art algorithms for cross-correlation computation over multiple streaming time series. Some capabilities are not demonstrated but trivially achievable with simple extensions of the algorithms. FilCorr comprehensively covers capabilities across several existing works, making it unique in the suite of correlation computing algorithms.

Table 3. Capabilities of FilCorr and related work. ✓ represents a claimed capability; – represents extendable capability, and × represents unknown.

	FilCorr	ParCorr [78]	BRAID [60]	COLR-Tree [7]	StatStream [86]
Data-independent	✓	×	✓	×	×
Filtering	✓	–	×	×	–
Lagged correlation	✓	×	✓	×	×
Real-time	✓	✓	–	✓	✓
Exact	✓	✓	×	–	✓
Parallel	–	✓	–	–	–

2.3.3 FilCorr: Filtered Lagged Correlation

The primary motivation of FilCorr is the need for systems to compute lagged correlation of filtered high frequency streaming time series from distributed sensors, such as seismic event monitoring. These systems require monitoring hundreds of sensors responsible for generating data at high-speed. Surprisingly, none of the existing methods from literature can deal with all of these requirements.

In this section, we present our solution in detail. Table 4 shows the symbols and definitions considered in our discussions. The main variables are also illustrated in Figure 14. For simplicity but without loss of generality, we assume all streams have the same sampling rate f, no discontinuity (no data loss during the transmission) and the observations are all aligned, which means they all have timestamps in set $\{1, ..., t-2, t-1, t, ...\}$.

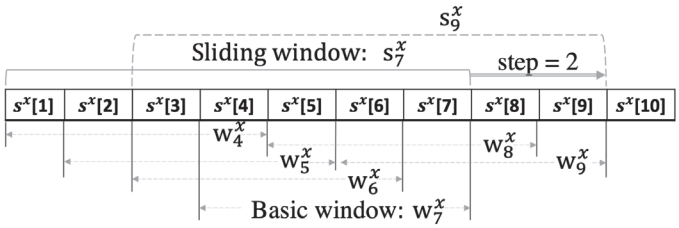

Figure 14. Examples of various windows on a stream, where $m = 4, l = 3$, and $step = 2$.

Lagged Correlation

We use $lcorr_t^{xy}$ to denote the lagged correlation value at time t between s^x and s^y. It is defined in Equation 9.

$$lcorr_t^{xy} = Max(corr_{tt_i}^{xy}, corr_{t_it}^{xy}); t_i \in [t-l, t] \tag{9}$$

Approved for public release; distribution is unlimited.

Table 4. Symbols and definitions.

Symbol	Definition
s^x	Streaming time series with ID x
N	Total number of streams that a system is processing
$s^x[t]$	The observation value of stream s^x at time t
f^x	Sampling rate of stream s^x
f_s	Lower bound frequency of the bandpass filter in Hz
f_t	Upper bound frequency of the bandpass filter in Hz
m	Length of basic windows in number of observations
l	Lag size in number of observations
w_t^x	The **basic window** in s^x at time t, includes m observations: $\{s^x[t-m+1] \dots s^x[t]\}$
$w\prime_t^x$	The **filtered basic window**, filtered version of w_t^x in the time domain
W_t^x	The **frequency window**, discrete Fourier transformed version of w_t^x
$W_t^x\|_{f_s}^{f_t}$	The **filtered frequency window** of W_t^x which only contains coefficients corresponding to frequencies from f_s to f_t
s_t^x	The **sliding window** in s^x at time t, covers observations in range $\{s^x[t-l-m+1]..s^x[t]\}$ includes $(l+1)$ basic windows: $\{w_{t-l}^x..w_t^x\}$
$w_t^x[j]$	jth element of the basic window w_t^x, $w_t^x[j] = s^x[t-m+1+j]$
$lcorr_t^{xy}$	Lagged correlation value between s_t^x and s_t^y at time t
$corr_{t_1t_2}^{xy}$	Correlation value of $w_{t_1}^x$ and $w_{t_2}^y$
$step$	Gap between two successive correlation computation in number of observations. If $step = 10$ and the first output is $lcorr_1^{xy}$ then the next output is $lcorr_{11}^{xy}$

$lcorr_t^{xy}$ is the largest Pearson's correlation coefficient value among the correlations between the most recent basic windows at time t and all the previous basic windows from time $t-l$ to t. Such a strategy can cover all possible cases when an event appears in different streams at different times.

Case 1: No lag. The event appears in both streams at the same time. $lcorr_t^{xy}$ equals to $corr_{tt}^{xy}$ which is the correlation value between basic window w_t^x and w_t^y.

Case 2: The event appears in s^y first then in s^x at time t, such scenario will be captured by computing correlation between w_t^x and $w_{t_i}^y$. $t_i \in [t-l, t-1]$.

Case 3: Similar to case 2, the event appears in s^x first then in s^y at time t. Then $lcorr_t^{xy}$ will be located among correlations between w_t^y and $w_{t_i}^x$. $t_i \in [t-l, t-1]$.

To combine all cases, we define the *sliding window* s_t^x and s_t^y. Each sliding window will cover all $(l+1)$ basic windows that are necessary to compute $lcorr_t^{xy}$. The relations are shown in Figure 14.

To represent this process with code, we have line 3 to line 10 in Algorithm 2 and line 5 to line 11 in Algorithm 3. It takes the last basic window from one sliding window and computes correlation with all basic windows in another sliding window and vice versa. It returns the largest value as the result of the lagged correlation between the two sliding windows.

Approved for public release; distribution is unlimited.

Lagged Correlation with Filtering

The following discussions are based on the ideal bandpass filter, which has frequency response values equal to 1 for frequencies in the band, and 0 for frequencies outside the band. For any other known frequency response, we can apply FilCorr trivially.

Naive approach: A straightforward approach to filter time series and compute lagged correlation. For the filtering, the naive approach will transform each basic window to the frequency domain, remove frequency components outside the filter band, and finally transform back to the time domain. This process shows up in the $filter$ function with lower cutoff frequency as f_s and higher cutoff frequency as f_t in Algorithm 2.

We define the *filtered basic window* $w\prime_t^x$, as the filtered version of w_t^x. $w\prime_t^x$ and w_t^x have the same length. Unlike two successive basic windows, $w\prime_t^x$ and $w\prime_{t+1}^x$ are independent with each other and no overlapping observations.

Once all the filtered windows in s_t^x and s_t^y are calculated, we can compute $lcorr_t^{xy}$ by calling the function $LagFilterCorr$ in Algorithm 2 with parameters (s_t^x, s_t^y, l, m, f, f_s, f_t). The $oneCorr$ function is based on Equation 10 derived from Equation 5 by substituting $w\prime_t^x$ for w^x.

$$corr_{t_1t_2}^{xy} = \frac{\sum_{j=0}^{m-1} w\prime_{t_1}^x[j]w\prime_{t_2}^y[j] - m\mu(w\prime_{t_1}^x)\mu(w\prime_{t_2}^y)}{m\sigma(w\prime_{t_1}^x)\sigma(w\prime_{t_2}^y)} \tag{10}$$

We can avoid some redundant computation by storing sums, means and standard deviations of all filtered windows which can be computed from Equation 11, 12, 13. Thus only $\sum_{j=0}^{m-1} w\prime_{t_1}^x[j]w\prime_{t_2}^y[j]$ needs to be computed for each pair of filtered windows. For our implementation, only one copy of a filtered window $w\prime_t^x$ and its statistics exist in the system.

$$\sum w\prime_t^x = \sum_{j=0}^{m-1} w\prime_t^x[j] \tag{11}$$

$$\mu(w\prime_t^x) = \frac{\sum w\prime_t^x}{m} \tag{12}$$

$$\sigma(w\prime_t^x) = \sqrt{\frac{\sum w\prime_t^x[j]^2}{m} - [\mu(w\prime_t^x)]^2} \tag{13}$$

Proposed approach: To speed-up the computation of lagged correlation on filtered time series, we propose combining the filtering and correlation computation in one step. Such an approach can bring both time and space efficiency.

We define the *frequency window* W_t^x, which is the Fourier transformed version of basic window w_t^x. $W_t^x|_{f_s}^{f_t}$ as the *filtered frequency window* which only contains half coefficients corresponding to frequencies from f_s to f_t in W_t^x. $W_t^x|_{f_s}^{f_t}$ will only keep one half of elements to save memory space since elements in W_t^x are complex conjugate symmetric. The main idea of our approach is to incrementally update the filtered frequency window $W_t^x|_{f_s}^{f_t}$ from the previous filtered frequency window $W_{t-1}^x|_{f_s}^{f_t}$ instead of calculating it from the basic window w_t^x. If all filtered frequency windows in a pair of sliding windows are computed then the lagged correlation coefficients can be directly computed using frequency components based on

Approved for public release; distribution is unlimited.

Algorithm 2 NAIVE

Function `LagFilterCorr`(s_t^x, s_t^y, l, m, f, f_s, f_t)

1 $w_t^x\prime \leftarrow$ `filter`(w_t^x, f, f_s,$2f_t$) $w_t^y\prime \leftarrow$ `filter`(w_t^y, f, f_s, f_t)

3 $curMax \leftarrow$ `oneCorr`($w_t^x\prime$, $w_t^y\prime$, m) `//case 1`

4 **for** $t_i = t - l : t - 1$ **do**

5 $w_{t_i}^x\prime \leftarrow$ `filter`($w_{t_i}^x$, f, f_s, f_t)

6 $w_{t_i}^y\prime \leftarrow$ `filter`($w_{t_i}^y$, f, f_s, f_t)

7 $tmp1 \leftarrow$ `oneCorr`($w_t^x\prime$, $w_{t_i}^y\prime$, m) `//case 2`

8 $tmp2 \leftarrow$ `oneCorr`($w_{t_i}^x\prime$, $w_t^y\prime$, m) `//case 3`

9 $curMax = max(curMax, tmp1, tmp2)$

end

10 **return** $curMax$

end

Function `filter`(w_t^x, f, f_s, f_t)

11 $LB = \lfloor mf_s/f \rfloor$

12 $UB = \lfloor mf_t/f \rfloor$

13 $W_t^x\prime \leftarrow FFT(w_t^x)$

14 $W_t^x\prime[0 \ : \ LB - 1] \leftarrow 0$

15 $W_t^x\prime[UB + 1 \ : \ m - UB - 1] \leftarrow 0$

16 $W_t^x\prime[m - LB + 1 \ : \ m - 1] \leftarrow 0$

17 **return** $IFFT(W_t^x\prime)$

end

Function `oneCorr`($w_{t_1}^x$, $w_{t_2}^y$, m)

18 **return** $[\sum_j w_{t_1}^x[j] w_{t_2}^y[j] - m\mu(w_{t_1}^x)\mu(w_{t_2}^y)]/[m\sigma(w_{t_1}^x)\sigma(w_{t_2}^y)]$

end

Approved for public release; distribution is unlimited.

Equations 6 and 8. Thus, our proposed method avoids computing repeated Fourier transforms and inverse Fourier transforms on every basic window and performs correlation computations directly in the frequency domain on fewer data. The length of $W_t^x|_{f_s}^{f_t}$ is $\lfloor \frac{f_t - f_s}{f} m \rfloor + 1$, where f is the sampling rate of the streams.

To maintain frequency components upon receiving a new observation, the algorithm removes the quantities for each frequency that contributed by the first observation $w_{t-1}^x[0]$ of basic window w_{t-1}^x and adds the quantities for each frequency that are brought by the new observation $w_t^x[m-1]$ of basic window w_t^x. To account for the sliding of the window, the algorithm updates the k^{th} coefficient by multiplying $e^{i\frac{2\pi k}{m}}$, for $\lfloor \frac{f_s}{f} m \rfloor \leq k \leq \lfloor \frac{f_t}{f} m \rfloor$. This process is applied to each of the frequencies from f_s to f_t. The steps are precisely represented by the function $filter$ in Algorithm 3.

Once filtered frequency windows $\{W_{t-l}^x|_{f_s}^{f_t}, \ldots, W_t^x|_{f_s}^{f_t}\}$ and $\{W_{t-l}^y|_{f_s}^{f_t}, \ldots, W_t^y|_{f_s}^{f_t}\}$ are ready, then the lagged correlation $lcorr_t^{xy}$ can be computed by calling Algorithm 3. The function $oneCorr$ in Algorithm 3 will be called to compute each $corr_{t_1 t_2}^{xy}$. Note that the correlation coefficients calculated from filtered frequency windows are *exactly* the same as the coefficients calculated from filtered windows in the naive algorithm. The exactness of our algorithm is directly derived from Parseval's theorem described earlier in Section 2.3.1.

To explain the function $oneCorr$ in Algorithm 3, we describe how Equation 10 is evaluated using a filtered frequency window instead of a filtered basic window. In the following we show how each term from Equation 10 can be expressed using frequency terms in detail.

Based on Equation 12, we can calculate the mean of the filtered basic window with Equation 14.

$$\mu(w\prime_t^x) = \begin{cases} \frac{W_t^x[0]}{m} & \text{if } f_s = 0 \\ 0 & \text{if } f_s > 0 \text{ (DC term is filtered)} \end{cases} \tag{14}$$

Based on Parseval's theorem in Equation 6, we can compute $\sum w\prime_t^x[j]^2$ in Equation 15. We multiply a constant 2 to include the symmetric part of $W_t^x|_{f_s}^{f_t}$. This also applies to Equation 16 and 17. There are two special cases, one case is when the $f_s = 0$ and another case is when the length of a basic window is an even number and the f_t equals to the Nyquist frequency which is one half of the sampling rate f. For both cases, $W_t^x[\lfloor \frac{m}{2} \rfloor]$ or $W_t^x[0]$ need to be subtracted after we multiply 2 since there is no symmetric value to them. Both values need to be subtracted if conditions for both cases hold.

$$\sum_{j=0}^{m-1} w\prime_t^x[j]^2 = \frac{2}{m} \sum |W_t^x|_{f_s}^{f_t}[k]|^2 \tag{15}$$

Then the standard deviations can be calculated as below if $f_s > 0$.

$$\sigma(w\prime_t^x) = \sqrt{\frac{2}{m^2} \sum |W_t^x|_{f_s}^{f_t}[k]|^2} \tag{16}$$

Lastly, the product term $\sum(w\prime_{t_1}^x[j] w\prime_{t_2}^y[j])$ can be expressed in Equation 17 based on Equation 8.

$$\frac{2}{2m}\Big(\sum |W_{t_1}^x|_{f_s}^{f_t}[k]|^2 + \sum |W_{t_2}^y|_{f_s}^{f_t}[k]|^2 - \sum |W_{t_1}^x|_{f_s}^{f_t}[k] - W_{t_2}^y|_{f_s}^{f_t}[k]|^2 \Big) \tag{17}$$

Approved for public release; distribution is unlimited.

Finally, we can derive the equation that appears in line 16 of Algorithm 3 when $f_s > 0$. For the case when $f_s = 0$, it only needs to include the $\mu(w_t^x)$ term in the computations since it no longer equals zero.

Algorithm 3 FilCorr

Function `LagFilterCorr`(s_t^x, s_t^y, l, m, f, f_s, f_t)

19 $LB \leftarrow m\lfloor f_s/f \rfloor$

20 $UB \leftarrow m\lfloor f_t/f \rfloor$

21 $W_t^x|_{f_s}^{f_t} \leftarrow$ `filter`($W_{t-1}^x|_{f_s}^{f_t}$, m, LB, UB, $w_{t-1}^x[0]$, $w_t^x[m-1]$)

22 $W_t^y|_{f_s}^{f_t} \leftarrow$ `filter`($W_{t-1}^y|_{f_s}^{f_t}$, m, LB, UB, $w_{t-1}^y[0]$, $w_t^y[m-1]$)

23 $curMax \leftarrow$ `oneCorr`($W_t^x|_{f_s}^{f_t}$, $W_t^y|_{f_s}^{f_t}$, m) `//case 1`

for $t_i = t - l : t - 1$ **do**

24 $W_{t_i}^x|_{f_s}^{f_t} \leftarrow$ `filter`($W_{t_i-1}^x|_{f_s}^{f_t}$, m, LB, UB, $w_{t_i-1}^x[0]$, $w_{t_i}^x[m-1]$)

25 $W_{t_i}^y|_{f_s}^{f_t} \leftarrow$ `filter`($W_{t_i-1}^y|_{f_s}^{f_t}$, m, LB, UB, $w_{t_i-1}^y[0]$, $w_{t_i}^y[m-1]$)

26 $tmp1 \leftarrow$ `oneCorr`($W_t^x|_{f_s}^{f_t}$, $W_{t_i}^y|_{f_s}^{f_t}$, m) `//case 2`

27 $tmp2 \leftarrow$ `oneCorr`($W_{t_i}^x|_{f_s}^{f_t}$, $W_t^y|_{f_s}^{f_t}$, m) `//case 3`

28 $curMax = max(curMax, tmp1, tmp2)$

end

29 **return** $curMax$

end

Function `filter`($W_t^x|_{f_s}^{f_t}$, m, LB, UB, d, a)

`//`d `is the element that will be deleted`

`//`a `is the element that will be added`

30 **for** q *from* $0 : UB - LB$ **do**

31 $k \leftarrow q + LB$ `//`k `is the index of` W_t^x

32 $W_{t+1}^x|_{f_s}^{f_t}[q] = e^{i\frac{2\pi k}{m}}(W_t^x|_{f_s}^{f_t}[q] - d \cdot e^{-i\frac{2\pi 0k}{m}} + a \cdot e^{-i\frac{2\pi mk}{m}})$

end

33 **return** $W_{t+1}^x|_{f_s}^{f_t}$

end

Function `oneCorr`($W_{t_1}^x$, $W_{t_2}^y$, m)

`//Assume` $0 < f_s < f_t < f/2$

34 **return** $\frac{\sum_q |W_{t_1}^x[q]|^2 + \sum_q |W_{t_2}^y[q]|^2 - \sum_q |W_{t_1}^x[q] - W_{t_2}^y[q]|^2}{2\sqrt{\sum_q |W_{t_1}^x[q]|^2 \sum_q |W_{t_2}^y[q]|^2}}$

end

We use the parameter *step* to control the output rate. FilCorr can output all-pair correlations upon receiving the next set of observations in the streams. However, the output rate does not necessarily have to be the same as the input rate. If the current sliding window is s_t^x, we can slide to s_{t+step}^x, where *step* is the number of observations in a stream the algorithm gathers before outputting the next set of pairwise correlations. When $step = 1$, the algorithm outputs at the same rate as the input. The larger the *step*, the slower the output rate.

Approved for public release; distribution is unlimited.

Computational Complexity

The time complexity of computing filtered and lagged correlation at any given time is composed of two parts: *i*) filtering and *ii*) correlation computation. For the filtering, FilCorr only takes $O(B)$ time based on the $filter$ function in Algorithm 3, B is the number of elements within the filter band, which is $1 + \lfloor \frac{f_t - f_s}{f} m \rfloor$. So the worst case is $O(m)$ for FilCorr when all the frequency components are kept. However, the $filter$ function in the naive algorithm will take $O(m \log m)$ due to the FFT algorithm.

As for the cost of computing lagged correlation coefficients, the for loop in both algorithms will be executed l times. Each iteration of the naive algorithm will take $O(m)$ based on the equation in line 18, while FilCorr only takes $O(B)$ time.

The final time complexity to compute one lagged correlation value is not the simple addition of the above two parts. It depends on how many pairs that a time series involves. Here we assume a time series will form a pair with all the other time series. Then the time to filter one basic window should be amortized among O(N) pairs, N is the number of streaming time series in the system. Thus the time complexity to compute one lagged correlation value for the naive algorithm is $O(\frac{m \log m}{N} + lm)$, and $O(\frac{B}{N} + lB)$ for FilCorr, $O(\frac{m}{N} + lm)$ in the worst case. In practice, the speedup is greater because the number of possible lags is much less than the window size, and the frequency band is much smaller than the number of observations in the signal. Finally, the total cost of the whole system has two factors; one is the number of computing units and the other is $O(N^2)$, which is the maximum number of pairs that N streaming time series can form.

The naive algorithm's space complexity is $O(Nlm)$ to maintain all the filtered basic windows in the most recent sliding window for all streams. The space complexity of FilCorr is $O(NlB)$.

Extensions to Our Implementation

Our proposal can be easily extended to execute in parallel and to employ different digital filters. In this section, we discuss the properties of FilCorr that provide such flexibility.

Parallelization: Since lagged correlation computation for one pair of streams is independent of other pairs, we can utilize multiple processing units (i.e., thread, core, processor, etc.) to expand the capacity to calculate multiple pairs in parallel. Each pair will maintain its own sliding windows for the streams s^x and s^y. In order to achieve the best real-time performance, one filtered frequency window with its statistics can be accessed by all pairs to save more memory and avoid redundant computation.

Different filters: Our method can adapt to other kinds of digital filters; for instance, the Butterworth filter in Figure 15. We can directly apply this filter on top of the box filter by changing each frequency weight. This way, we can still save time by not computing unnecessary frequencies and keep the filter property within the bandwidth. This operation will not change the overall time complexity since it is a linear operation to factor the weights in all frequencies. Note that incrementally updating DFT coefficients is also a linear time operation. Our method is not limited to the digital frequency-domain filters. A time-domain digital or analog filter can also be applied for streams passing through as pre-processing then passing into our method for correlation computation on the targeted frequencies.

Approved for public release; distribution is unlimited.

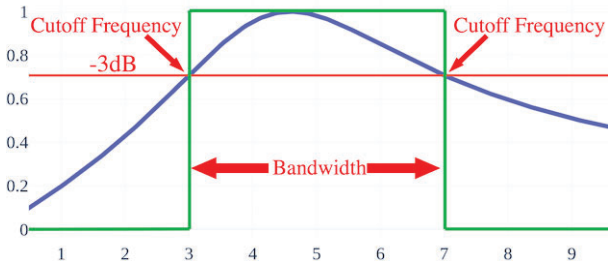

Figure 15. Butterworth second order bandpass filter 3-7Hz.

Discussion on Data Independence

The general idea of FilCorr is to exploit the use of digital filters to achieve faster pairwise correlation computation. Besides removing undesired noise, filters allow computing of the correlation between time series in a reduced dimensional space provided by the frequency domain. It turns the time cost of our method data-independent and will be affected by neither the sparsity nor the similarity. This cost is related to the lag and the bandwidth assumed by the filter, typically much smaller than the original series. It is also important to note that unlike other methods that speed up the correlation computation by pruning improbable pairs and provide approximate results, our approach efficiently calculates all possible lagged pairs and provides exact results.

Managing Spurious Correlation

Filtered correlation can occasionally be spurious because of the Ringing effect. It occurs due to spectral leakage when the length of the basic window is mutually prime with the period of the signal. For instance, in Figure 16(a), we show two uncorrelated signals (correlation = 0.14) obtained from different seismometers. If we filter such signals by a box filter, we obtain a false high correlation of 0.70 in the resulting signals, as illustrated in Figure 16(d). This high correlation is mainly caused by similar oscillations at the edges of both signals. This effect can be addressed by "windowing" (i.e., multiplying with a window function) the time series before converting them to the frequency domain. The resulting signals after the process of windowing using a Hamming window (Figure 16(b)) are illustrated in Figure 16(c). Finally, the filtered signals after the windowing process are illustrated in Figure 16(e). We note that the correlation of the original signals is reduced to 0.13, eliminating the previously observed false high correlation.

Windowing on each basic window data will change the filter process since we couldn't incrementally maintain the filtered frequency window. Here we need to apply the filter function from the naive algorithm.

2.4 Results and Discussion: Signal Monitoring

2.4.1 Experimental Evaluation

All our experiments are reproducible. The code, data, and additional results are available in our supporting website [84]. FilCorr is exact and deterministic, and the efficiency is not data-dependent.

Approved for public release; distribution is unlimited.

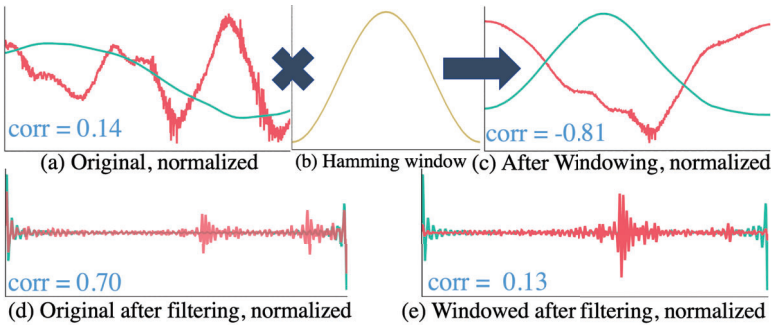

Figure 16: Example of the Ringing effect on two uncorrelated seismic signals and how the multiplication by a Hamming window can address the false high correlation between them.

Sanity Check

Before the experimental evaluation and comparisons with existing methods, we show a sanity check to demonstrate that our approach is fast enough to be employed in a real-time system that is capable of monitoring hundreds of sensors with a high sampling rate. To demonstrate this claim, we develop a system named Seisviz (`www.seisviz.com`) that will render the lagged correlation values computed by FilCorr in real-time. It has been used successfully for monitoring a seismic network at Yellowstone, WY, USA. This network has 30 stations, where each station has up to six channels. Each channel represents an individual stream of observations at 100Hz. We obtain 670 pairs if we pair the streams from different stations by the same channel type. We demonstrate videos of detected earthquakes in real-time by Seisviz on our website [84]. In Section 2.4.2, we return to the discussion about our findings on seismic data and a detailed system implementation description.

Setup

All experiments are performed on a desktop computer with an Intel i9-9900k (8 cores) processor, 32GB of memory, running a Linux operating system. As FilCorr is data-independent, we use synthetic data for various experiments. The performance on real-world data will be discussed in the case studies presented in Section 2.4.2.

We create two testing scenarios to evaluate the performance of naive and FilCorr algorithm: *offline* and *online*. In the offline scenario, all observations are available beforehand, so the system will use its full power to compute until it finishes computation on all data. For the online scenario, the observations are generated in a streaming fashion with the speed of sampling rate.

In the offline scenario, we measure the total execution time, including I/O operations. In the online scenario, we seek to find the maximum number of streams for which the system can compute their pair-wise lagged correlation without any delay. To measure this, we create an ideal environment where all the streams have an equal length and the same sampling rate specified by the parameter f. We consider the system capable of processing this number of streams if the finish time is ahead of the expected next computation time. There is no need to measure at each step because if the number of streams exceeds the system capability, then

Approved for public release; distribution is unlimited.

the extra time to finish the computation will accumulate at each step and be reflected in the finish time.

For both offline and online, we run ten trials to confirm an algorithm's capability on a certain number of streams to account for random events in the operating system.

Efficiency

In Figure 17, we show the execution time of FilCorr and the naive algorithm in the offline scenario. We consider three filter bandwidths 5Hz, 25Hz, and 50Hz, and five sampling rates between 100Hz to 300Hz with increments of 50Hz. The naive algorithm's execution time remains the same for the same sampling rate no matter the bandwidth size. This is because the number of observations used for correlation computation of the naive algorithm does not change along with the filter bandwidth in the time domain. On the contrary, the narrower the band the fewer frequency components for FilCorr to compute, so the less the execution time. The exact number of observations used for the correlation computation can be calculated by $1 + \lfloor \frac{f_t - f_s}{f} m \rfloor$. Another conclusion we can draw from the figure is that the growth rate for FilCorr is much slower than for the naive algorithm for a fixed bandwidth when the sampling rate is increasing. This is because the basic window length is calculated based on the time, which is 10 seconds here. For FilCorr, the number of frequency components in the filter band remains the same since B is defined as $1 + \lfloor \frac{f_t - f_s}{f} m \rfloor$, m equals $10f$ so B equals $1 + \lfloor 10(f_t - f_s) \rfloor$, The only extra cost for FilCorr is coming from the longer lag. However, for the naive algorithm, it is affected by both longer lag and more observations for computing the correlation; thus, it increases at a much higher rate than FilCorr.

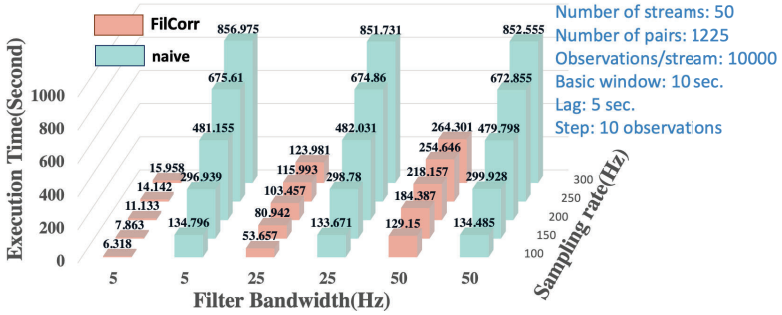

Figure 17. Offline performance of FilCorr and the naive algorithm.

In Figure 18, we show the number of pairs that each algorithm can process without delay in the online scenario. We vary the input sampling rate and output rate for both algorithms. In all experimental settings, FilCorr can process (up to 4×) more sensors than the naive algorithm. The performance gap increases with higher input or output rate. For other filter bands, the general performance trends hold.

Comparison to Existing Method

Based on our previous comparison shown in Table 3, we argue that FilCorr is a comprehensive method for streaming correlation computation. However, although not an ideal match in capabilities, we identify ParCorr [78] as the most recent baseline with state-of-the-art performance. ParCorr calculates pairwise correlation in parallel with the Apache Spark system

Approved for public release; distribution is unlimited.

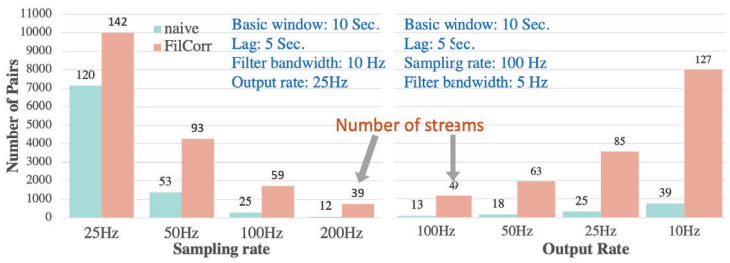

Figure 18. Online performance of FilCorr and naive algorithm.

based on randomly projected sketches. Note that ParCorr does not compute lagged correlation. The following experiments are conducted on the same setup as previous experiments.

In order to favor ParCorr's implementation in our comparison, we perform all the experiments in the offline scenario. To offset the Spark system's costs, we conduct another set of experiments as offsets. Each offset experiment will only process one time series with window size as 1, step size as 1, and the length of this time series depends on the actual corresponding experiment parameters. Our goal is to make sure the sliding window in both offset experiments and actual corresponding experiments move the same number of times. All the experiment results here are adjusted based on the results of the corresponding offset experiment.

Since the ParCorr algorithm is data-dependent, we use two sets of synthetic data with 2,000 observations in each targeted to emulate the best-case and worst-case scenarios for ParCorr. The cost of ParCorr depends on the number of pairs it can prune without computing the correlation coefficients. Our first synthetic dataset contains sequences of uniformly distributed random numbers, which is expected to contain only uncorrelated pairs. Thus, a random noise dataset is the best data for ParCorr where it can prune all possible pairs. To further boost ParCorr's performance, we use a high correlation threshold (*candThreshold*) for better pruning. The purpose of this is to guarantee that no two series will lead to actual correlation computation for ParCorr. Besides, we also set the parameter *candThreshold* with a high value in ParCorr as double insurance. On the contrary, the second dataset is a sinusoid that is expected to have all possible pairs of streams be highly correlated. In this case, ParCorr computes correlations for all possible pairs, failing to prune and demonstrating the worst-case performance. For FilCorr and the naive algorithm, the pairs are generated based on all possible combinations from all the streams.

We show the performance comparison in Figure 19. The light grey shaded area represents the range of performance by ParCorr. The worst-case performance (on sinusoid data) is illustrated by the superior grey line with solid circles, and the best-case (on random data) by ParCorr is illustrated by the inferior grey line with solid boxes. The actual performance of ParCorr on any other dataset should be in between the worst-case and best-case lines. Figure 19(zoom-out) shows that the time spent by FilCorr for various lags is well inside the shaded area. To be fair to ParCorr, when we consider the synchronous correlation (lag = 0), FilCorr is more efficient than the best-case of ParCorr up to around 700 streams as shown in Figure 19(zoom-in). Therefore, we recommend FilCorr on a single desktop computer when the number of streams is less than 700, instead of using a parallel system.

In the second experiment, we fix the total number of streams at 800 and vary the *step* from 5 to 20 observations, which correspond to output rates of 20Hz to 5Hz, respectively.

Approved for public release; distribution is unlimited.

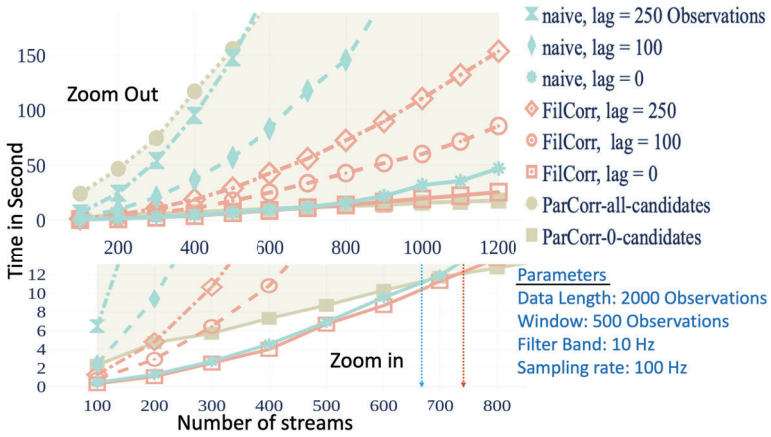

Figure 19: Comparison with ParCorr fixing *step* = 20. *The vertical red line shows the crossing point between ParCorr and FilCorr with lag = 0, and the vertical blue line shows the crossing point between ParCorr and naive algorithm with lag = 0.*

We note in the results shown in Figure 20 that the execution time for all methods increases when the *step* is getting smaller to compute more correlation coefficients for a higher output rate. However, ParCorr increases at a higher rate compared to FilCorr. The zoom-in figure shows that FilCorr with $lag = 0$ has better performance than the best-case of ParCorr when the output rate is higher than 6Hz.

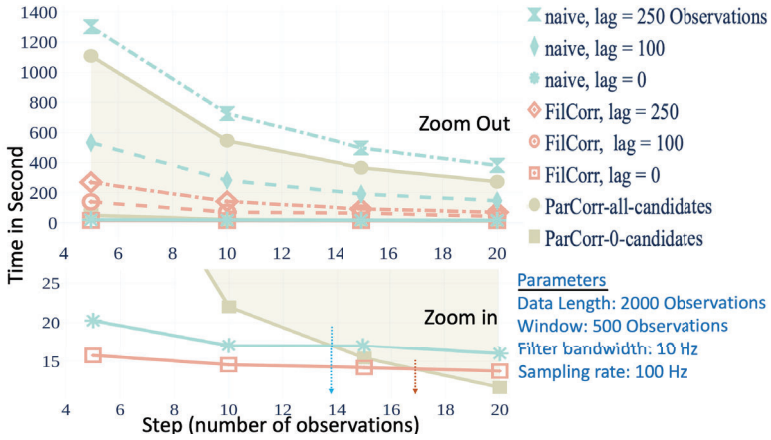

Figure 20. Comparison with ParCorr varying the *step* from 5 to 20 observations. *The vertical red line shows the crossing point between ParCorr and FilCorr with lag = 0; the arrow points to where the output rate is 6Hz. The vertical blue line shows the crossing point between ParCorr and the naive algorithm with lag = 0, and the arrow points to the output rate as 7Hz.*

Parameter Sensitivity

In this section, we discuss the algorithms' sensitivity to three design parameters: *i*) lag, *ii*) filter bandwidth, and *iii*) window size. We consider the online scenario and measure the performance between the naive algorithm and FilCorr. We fix the sampling rate at $f = 100$Hz and *step* = 10 observations for all experiments. The results are presented in Figure 21.

Approved for public release; distribution is unlimited.

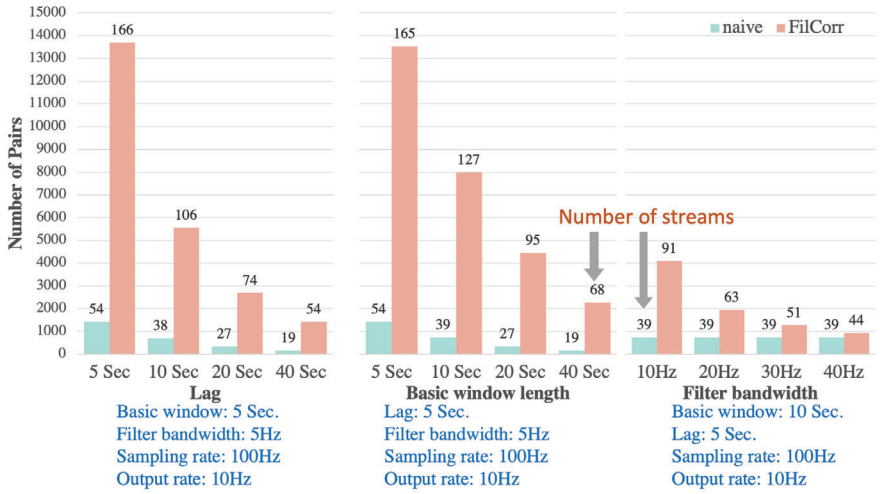

Figure 21: Results for the parameter sensitivity test considering the online scenario.

The results show that doubling either the window size or the lag size has similar effects, where the number of pairs the algorithm can handle shrinks by half for both FilCorr and the naive algorithm. The performance of FilCorr approaches the naive algorithm's when the bandwidth increases.

2.4.2 Case Studies

In this section, we present three case studies from diverse domains. We demonstrate that our technique is suitable for different monitoring tasks, such as seismic, body interactions, and patient monitoring through lagged correlation of filtered data.

Seismic Event Monitoring

We have deployed a system for monitoring seismic events in which FilCorr is one of the core components. We designed such a system, named Seisviz `www.seisviz.com`, by following the separation of concerns design principle (SoC) [48]. The main components of Seisviz are illustrated in Figure 22.

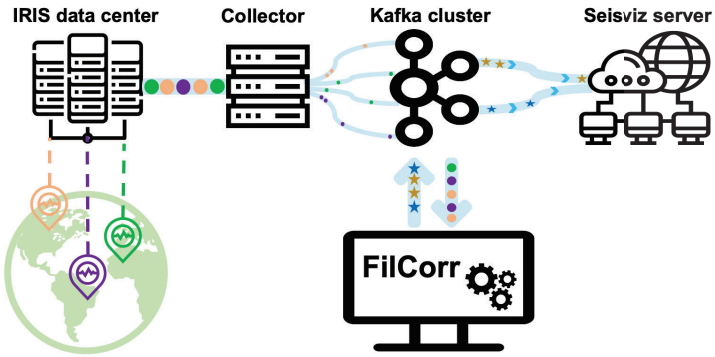

Figure 22. Main components of Seisviz, a real-time system for seismic event monitoring.

Approved for public release; distribution is unlimited.

The system has four modules: *i*) data collector, *ii*) the Kafka cluster, *iii*) the FilCorr computing unit, and *iv*) the Seisviz web server. Such a modular design is useful for developing and maintaining the system, while robustness and scalability are improved because failures can easily be tracked to one of the modules, and scaling each module is easier than scaling the whole system. There are two data pipelines in the system, one path originates at the seismic sensors and ends in the FilCorr computing unit. The other path originates at the FilCorr computing unit and ends in the Seisviz server. The seismic observations are fetched from the Incorporated Research Institutions for Seismology - IRIS data center [2] to the Kafka cluster, then consumed by the FilCorr computing unit. The correlation values are computed by the FilCorr computing unit and saved in the Kafka cluster, and finally consumed by the Seisviz web server.

The Kafka cluster was used for temporary data storage and data distribution. Kafka has essential features that meet our requirements, such as processing streaming data in real-time, storing a certain amount of historical data in a durable way, and allowing multiple consumptions by several applications and systems. Besides, it provides extra benefits, including fault tolerance and high availability [61], which can enhance the reliability and the scalability of our system.

A few key points are worth discussing when using Kafka for time series data. The first thing is the order of observations since it is natural to keep each observation in a timely order or index-based order for time series; however, no such order can be maintained if we use multiple Kafka topics with one partition or one topic with multiple partitions to store observations from a time series. In other words, there is no guarantee that the order for each observation arriving at the consumer is the same order when they are generated. So the downstream services need to restore the order. This is because Kafka can only guarantee the records from the same topic partition will arrive at the consumer in the same order as they were appended to the partition but not for the records across many partitions. This leads to another approach that uses a topic with only one partition to store observations from a time series. Such an approach can bring time efficiency to the downstream services as they no longer need to restore each observation order. However, this may cause a performance penalty when consuming a large number of records at a time compared with the approach using a topic with multiple partitions on several nodes in a cluster. We choose the latter approach for simplicity, and our system scale is not reaching that performance bottleneck.

Secondly, Kafka only supports millisecond precision for timestamps, generating losses when the period of a stream is less than a millisecond. To solve this, we first look at the structure of the Kafka record. Each record has three attributes: key, value, and timestamp. The key attribute is free in our case, so we can combine timestamp and key attributes to store the time information of an observation; time components after milliseconds can be saved in the key attribute. This approach can support the precision level up to nanoseconds, which is sufficient for virtually any streaming time series problem.

As previously stated, our system considers IRIS as a data source. IRIS provides a protocol called SeedLink[1] for users to receive real-time data. The streams of time series from IRIS are sent out in the form of batches containing a certain number of observations. Although the SeedLink is based on the TCP/IP protocol that guarantees packets transmission in order

[1] `https://ds.iris.edu/ds/nodes/dmc/services/seedlink/`

Approved for public release; distribution is unlimited.

and without any loss, the packets' order may not be consistent with observations' time order. To address this, we develop a collector module to save the streams in the Kafka cluster and, most importantly, recover the original data streams in time order. This process will encounter three different scenarios: 1) There is a time gap between data batches; in other words, a segment of observations of a time range is missing. 2) Two batches are overlapping. 3) There is a time shift among observations; if the time of one observation is shifted, then the time duration between this observation and the previous observation is no longer consistent with the sampling rate. This time shift has to be smaller than the sampling period to distinguish this scenario from the first scenario. Once we have all the necessary observations, we can start computing the filtered lagged correlation values using the FilCorr algorithm.

The back-end server will read the computed correlation value from the Kafka cluster, group the results based on the timestamp, and then send the results in a streaming manner to the front end. The front-end website will render the correlation value on a colored line between two points on the map, as illustrated in Figure 23. The points represent the locations of stations, which consist of various seismometers. The correlation value is depicted by the color and transparency of the line between two stations. Since there are usually multiple seismometers at one station, if there are multiple pairs of streams between two stations, then only the correlation with the highest value will be rendered at the moment.

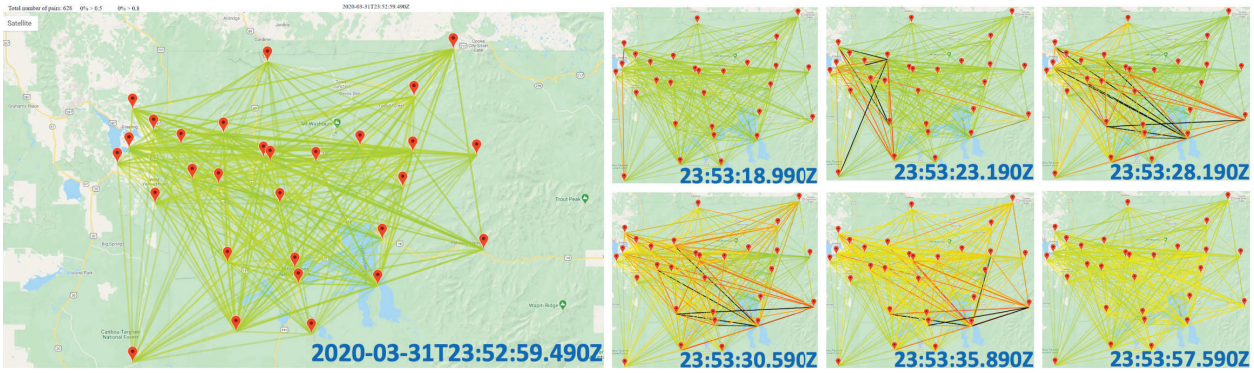

Figure 23. Pairwise correlation among 29 stations at Yellowstone, WY over different times given an M6.5 earthquake that occurred in Challis, Idaho, at 23:52:30 2020-03-31(UTC).

In Figure 23, we illustrate the propagation of a seismic event of magnitude 6.5 that occurred at Challis, Idaho, on March 03, 2020[2] and which was observed by our system about 300 miles away at the stations at Yellowstone. In this representation, each red location symbol represents a station, and a colored line represents the lagged correlation between two stations. Different colors and levels of alpha reflect the lagged correlation value. The correlation values from 0 to 0.5 to 0.9 are mapped from green to yellow to red. A black edge represents a correlation value greater than 0.9. We notice that the correlated edges are appearing between station pairs as the earthquake wave reaches them. Similarly, edges become uncorrelated when the wave has passed through.

In summary, our system is monitoring a seismic network with 30 stations in Yellowstone, Wyoming. There is a total of 98 streams, which can form a total of 670 pairs. Our system is computing the correlation of these 670 pairs at a 10Hz rate. Correlation coefficients can

[2] `https://earthquake.usgs.gov/earthquakes/eventpage/us70008jr5/executive`

Approved for public release; distribution is unlimited.

capture earthquake propagation through a region in real-time, which can easily be converted to a detector with the rule: *"If more than Q% of pairs of stations are highly correlated (> 0.8), an earthquake is propagating."* The utility of such a detector is massive for early warning systems because seismic waves propagate at 8 km/s or slower, which is much slower than electronic signals carrying the warning.

As we previously have shown filtering is essential for seismic data in order to remove noise and obtain a representation in the frequency domain that better describes the signals. For this case study, we consider a 20 seconds window size, a box bandpass filter with a cutoff frequency of 3Hz and 7Hz, and a lag of 10 seconds.

Although Seisviz website shows the results with a few minutes lag, we still claim our system is real-time since the delay occurs before the data arrive at our system. The delay is typically caused at the origin (i.e., seismometer) and during the transmission process. The delay time varies, and is beyond our control. Seisviz waits initially to accumulate enough data to be able to calculate the first set of correlations. After that, the system processes (i.e., flows data through the modules and computes the correlations) at a faster rate than the rate of streams.

2.5 Conclusion: Signal Monitoring

This part of the project demonstrates an algorithm, FilCorr, to compute filtered and lagged correlation over streaming time series. FilCorr combines filtering and cross-correlation computing operations in one step to obtain the lagged correlation between streaming time series efficiently. FilCorr is faster than the state-of-the-art ParCorr algorithm that computes correlation in parallel. We show three case studies where the algorithm achieves promising results towards greater societal impacts. We also provide a publicly available real-time system named Seisviz that employs FilCorr in its core mechanism for monitoring a seismometer network.

Approved for public release; distribution is unlimited.

Chapter 3

FASER: Seismic Phase Identifier for Automated Monitoring

3.1 Summary: Signal Classification

Seismic phase identification classifies the type of seismic wave received at a station based on the waveform (i.e., time series) recorded by a seismometer. Automated phase identification is an integrated component of large scale seismic monitoring applications, including earthquake warning systems and underground explosion monitoring. Accurate, fast, and fine-grained phase identification is instrumental for earthquake location estimation, understanding Earth's crustal and mantle structure for predictive modeling, etc. However, existing operational systems utilize multiple nearby stations for precise identification, which delays response time with added complexity and manual interventions. Moreover, single-station systems mostly perform coarse phase identification.

In this part of the project, we revisit seismic phase classification as an integrated part of a seismic processing pipeline. We develop a machine-learned model FASER, that takes input from a signal detector and produces phase types as output for a signal associator. The model is a combination of convolutional and long short-term memory networks. Our method identifies finer wave types, including crustal and mantle phases. We conduct comprehensive experiments on real datasets to show that FASER outperforms existing baselines. We evaluate FASER holding out sources and stations across the world to demonstrate consistent performance for novel sources and stations.

3.2 Introduction: Signal Classification

Real-time seismic signal processing is a key element of the geophysical monitoring required for early warning systems for earthquakes, underground mineral exploration and mining, and nuclear explosion monitoring. Seismic signal processing pipelines involve several sequential steps that start with signal (e.g., from an earthquake) detection from raw seismic signals recorded at a seismic station, and in the end, produce a formalized event bulletin for real-time alarm generation as well as future analysis. Figure 24 shows a typical pipeline. Phase identification is a key step in this pipeline subsequent to the signal detection step, which can

Approved for public release; distribution is unlimited.

be framed as a classification problem that takes a detected seismic signal as input, and outputs the phase label. Phase identification is required for proper utilization of the downstream steps of the pipeline, for example, earthquake location estimation, tomographic studies, and understanding of the Earth's crustal and upper mantle structure [21]. A successful phase classifier must classify a detected seismic waveform into transverse waves (ending with S in Figure 24.right) and compressional or longitudinal waves (ending with P in Figure 24.right), and all their subtypes.

Single Station vs. Array. At present, in operational systems, seismic phase identification is heavily dependent on the use of multiple close-by seismic monitoring stations, forming an *array* of stations [35]. High-quality arrays enable better detection and improved signal-to-noise-ratio, and estimation of phase velocity and direction of arrival; which greatly benefit both phase identification and association. Relative arrival times of seismic phases at different arrays and of different detections at the same array, together with their directions of arrival, are used to accurately classify and associate phases [64]. Unfortunately, most new stations added in dynamic response to changing monitoring needs, such as in oil fields or novel seismic sources, will be individual stations rather than arrays. This reduces the processing pipeline's performance because phase identification is less accurate for single stations than it is for arrays. In this work, we consider automated phase classification on data collected at a single station to enable rapid deployment addressing dynamic needs.

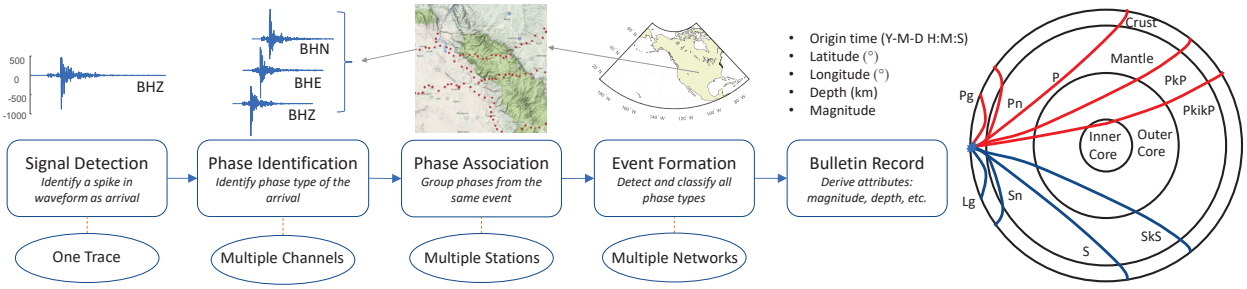

Figure 24. (left) Typical seismic data processing pipeline. Our objective is to develop a Machine Learning model for phase identification. (right) Travel times with respect to distance for various seismic phases. Phases ending with P can commonly be categorized as P, and phases ending with S can commonly be categorized as S.

Fine vs. Coarse Classification. In addition, most existing research work on automated seismic phase classification considers only the two high-level categories, while most monitoring applications require finer classification [58, 18]. For global monitoring, seismic signals are initially classified as teleseismic P (including more complex phases such as PkP and PkikP) or regional P or S (i.e., Pn or Sn). Refinements to the phase identification to add teleseismic S and crustal P and S phases (i.e., S, Pg, and Lg) are made much later in the processing pipeline. In this work, we consider classifying into finer phases at the initial identification step. In addition, phase detection and phase classification are usually sequential steps. Seismic signal detection algorithms are very well developed [51, 38, 66], and signal detection is a key module in the processing pipeline. In this work, we consider phase classification *only after* detection, unlike some recent works that address detection and classification jointly [57, 58].

Approved for public release; distribution is unlimited.

Global vs. Regional: Lastly, most existing research work focuses on local and near regional methods that only observe two major phases (local P and S, or occasionally regional P and S), and uses these phases to determine source location and origin times, mostly due to constrained focus. However, in a *global* monitoring application, all kinds of teleseismic (>1000km), regional (>200km), and local waves can arrive at various degrees of temporal overlap with arbitrary arrival order.

Figure 24 (right) illustrates some of the complexity of global seismic arrivals. Identifying the correct phase from a complex waveform containing multiple arrivals is difficult for global monitoring applications. In this work, we consider a global monitoring network, the International Monitoring System (IMS) network, that was established as part of the verification regime of the Comprehensive Nuclear-Test-Ban Treaty. IMS data are processed in near real-time at the International Data Centre (IDC) in Vienna, with initial detection, phase identification, and association performed automatically and then curated by human analysts. Our method can achieve significant classification accuracy even in such a complex application as presented by a global seismic network data.

Challenges to this research. First, there is no publicly available dataset of seismic signals categorized into six-phase classes. The main hindrance of curating such a dataset is the relative rarity of some phases compared to others. For example, in a continuous one-year time frame of the IMS catalog, 128K seismic waveforms are classified as P, and only 1300 are classified as S. Also, the manual labeling of these distinct phase-types requires a depth of knowledge and rigorous training. Second, the phases within broader hierarchical categories share highly similar spectral and amplitude properties. Also, seismic signals originating at different geolocations exhibit different propagation effects, resulting in dissimilar signal properties for the same phase-type. There are also differences among waveforms of a single type at different distances. The current bottleneck in processing is the association of seismic phases with their most likely sources, which could be improved by more accurate initial phase classification.

This work. To tackle these challenges, we focus on a few different aspects. First, we have curated a small-scale yet *balanced* seismic phase dataset collected from IMS network data. From an imbalanced collection of more than 200K seismic events, we have narrowed down to 16K events with finer and balanced phase labeling. We use Continuous Wavelet Transforms (CWT) to obtain a time-frequency representation from raw seismic time-series to utilize both temporal and spectral information. The CWT representation has been shown to be resilient to dynamic noise in waveforms [37]. We design an end-to-end deep neural network to perform phase identification using these CWT representations. We leverage the power of Convolutional Neural Network (CNN) to capture low-level features from the CWT representations that are invariant to frequency, scale, and position [59]. However, due to locally constrained receptive fields, CNNs are inadequate in modeling long-term temporal dependency, whereas seismic signals contain distinctive temporal patterns across different phase-types [57]. To mitigate this limitation, we incorporate Long-Short Term Memory networks (LSTM) on top of the CNN as LSTM can effectively model long term temporal patterns and dependencies.

Our proposed method FASER can perform fine-grained phase identification using single-station data from the global seismic network. Due to minimal preprocessing requirements

Approved for public release; distribution is unlimited.

and instantaneous output generation, it can be readily integrated into the existing real-time seismic signal monitoring pipeline. We show a comprehensive experimental evaluation of FASER using a real dataset in comparison with existing methods to validate improved performance. We justify the generalizability of FASER by demonstrating case studies for applications in novel operating conditions. To the best of our knowledge, this is the *first attempt* to perform *fine-grained phase identification* using single-station seismic signal data.

3.3 Methods, Assumptions, and Procedures: Signal Classification

3.3.1 Related Work

In general, the methods for seismic phase identification can be broadly categorized into two types, (1) heuristic template matching and statistical analysis based methods, and (2) deep learning based methods.

Statistical and Heuristic Methods. Since the early inception and development of seismic signal monitoring, several rule-based and physics-driven methods have been proposed for seismic phase identification. In [56], data-adaptive polarization filtering methods have been used for the phase detection task. In [8], the difference between the short-term average (STA) and the long-term average (LTA) of the seismic signal has been used for automated detection. Several methods have used higher-order statistics like kurtosis and skewness [33, 32]. Also, few methods have used frequency domain information [83, 37]. However, these methods perform poorly in the presence of noise and when the seismic events are of low magnitude. A few other works have proposed a similarity search based template matching method [55, 22]. But, such methods are heavily dependent on a prior collection of sample signal templates and often fail to generalize when used for phase detection at novel stations. Also, the pairwise similarity search with each sample template renders these methods computationally intensive and inefficient for real-time monitoring.

Deep Learning based Methods. Recently, multiple deep learning methods have been proposed to address the aforementioned shortcomings in the context of phase detection and identification [51, 38, 35, 17]. A deep learning based grid-free phase association method for phase identification has been proposed in [58]. In [57], a CNN based architecture has been used for phase identification from one-dimensional seismic signals. More recently in [18], the use of time-frequency representation and CNN has been explored. These methods have shown promising results compared to previously used statistical and heuristic-based methods as CNNs can effectively model low-level structured patterns into high-dimensional embedding. However, due to locally constrained receptive fields, CNN cannot capture the long-term temporal patterns. Thus, these models cannot fully exploit the higher-order temporal structures in seismic signals, distinctive across different phase-types. Moreover, none of these methods perform fine-grained phase identification.

Approved for public release; distribution is unlimited.

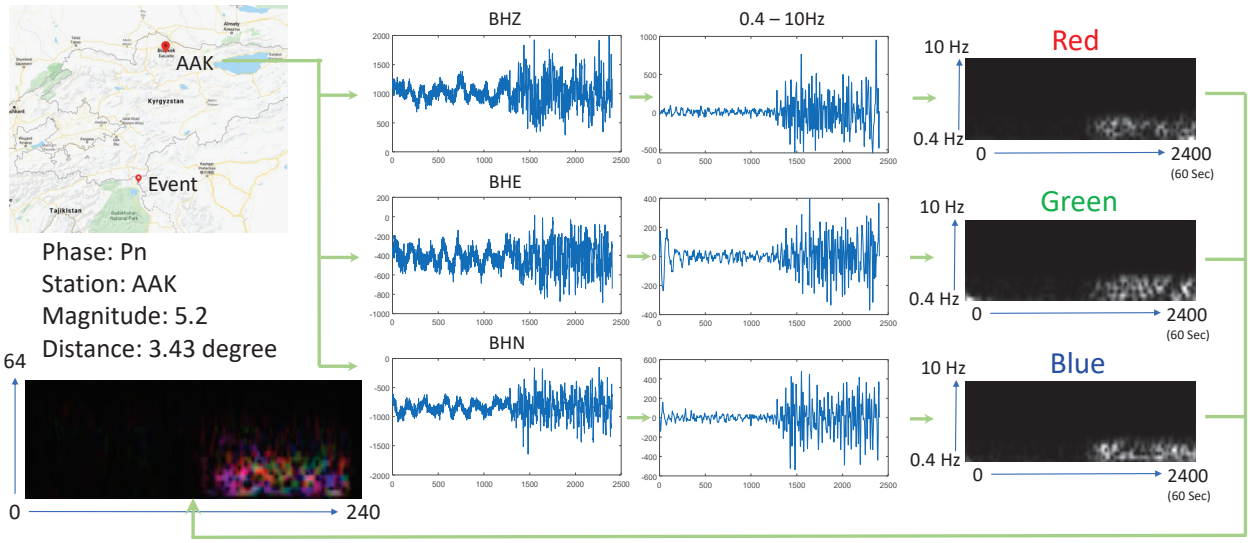

Figure 25. Input images of waveforms are created by taking Continuous Wavelet Transforms (CWT) of individual channels (i.e. BHZ, BHN, BHE).

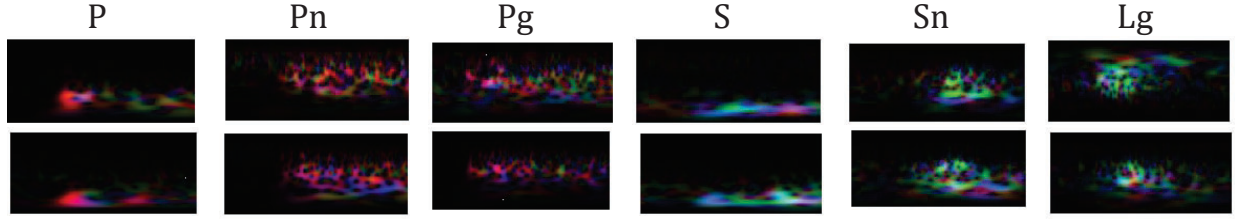

Figure 26. Continuous wavelet transforms of pairs of examples from all phase types. *Compressional or longitudinal waves (P, Pn, Pg) are dominantly red due to high vertical component amplitudes. Transverse waves (S, Sn, Lg) are dominantly green/blue due to high horizontal component amplitudes.*

3.3.2 Methods

In practice, the seismic phases are manually labeled by experienced analysts using multi-modal information, i.e., signal amplitude, frequency components in the signal, the distance between event origin and monitoring station, depth of the event origin, etc. However, for generalized and real-time phase classification, there exist a few challenges to producing such information a priori. Depth and source location estimation are intricate regression problems requiring complex analysis. Previously, the effectiveness of time-frequency representations has been demonstrated in a multitude of seismic signal processing tasks [68, 37]. Therefore, we use the Continuous Wavelet Transform (CWT) to obtain spectral-temporal features, as it produces higher spectral-resolution and more precise temporal-localization than other time-frequency transformations (e.g., Short-Fourier Transformation) [37]. We use a composite CWT image, where the vertical component CWT coefficients are represented by red brightness, and the two horizontal components are represented by the brightness of green and blue (See Figure 25).

The time-frequency representation of seismic events contains distinct structured features based on their phase-type. However, at a low-level, these features are highly overlapping.

Approved for public release; distribution is unlimited.

Convolutional Neural Networks (CNN) have been widely used in the domain of Computer Vision [31, 25], Natural Language Processing [19, 28], Speech recognition, [6] and other related domains to learn high-level features from raw structured input for better contrastive representation learning. As CNN's can model the local correlation of spatial and temporal patterns, it is highly suitable for our two-dimensional CWT feature maps, which contain incremental time information on one axis and frequency information on the other.

CNN's, however, are inadequate in learning long-range temporal dependencies due to their locally constrained receptive fields [59]. Nevertheless, the long-term temporal patterns in the seismic signals, which are well preserved in CWTs, are vital distinctive features across different phase types as showcased in Figure 26. To circumvent this limitation, Recurrent Neural Networks (RNN) have been instrumental in modeling the temporal dependencies by using the cyclic feedback mechanism from previous time-step inputs. Long Short-Term Memory networks (LSTM), an improved variant of vanilla RNNs, are capable of learning and modeling long-term temporal patterns and dependencies [29, 36]. It has been shown that higher-level features can be helpful in learning the underlying factors of variations within the input, which should make it easier to learn temporal structures between successive time-steps [77]. Thus, oftentimes CNNs have been successfully used as preceding layers before more complex sequential models to reduce the local temporal and frequency variations [59].

Motivated by these aforementioned successful use-cases, we use a combination of CNN and LSTM in an end-to-end network. First, we utilize CNN to identify low-level spectral-temporal features that are invariant to frequency, scale, and position. Afterward, we organize the output features obtained from CNN into sequential features preserving the temporal ordering. We feed these higher-level sequential representations of low-level structured patterns as input into the LSTM. By utilizing the cyclic feedback mechanism in between consecutive time steps, LSTM can better model the long-term temporal correlation in the seismic signal. Finally, we feed the output from each time-step into dense layers to make the final output prediction.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) [31] perform convolution operations on the input feature map using fixed-size kernels (learned during the training step) to produce higher-order representations. Convolution operations are usually followed by a non-linear activation function and max-pooling layers. The use of an activation function introduces non-linearity, and the max-pooling reduces sensitivity to temporal or spatial variation. CNN's are adept at learning local structural relationships and are invariant to feature scaling, which reduces the dependency on heavy data preprocessing and feature engineering [85].

Long Short-Term Memory Network.

Long Short-Term Memory (LSTM) networks are an improved variant of traditional Recurrent Neural Networks (RNNs) [27]. RNNs can model temporal dependencies in the data by utilizing feedback connections by considering both input at the current time step as well as output of the last time step's hidden state. However, vanilla RNNs suffer from the vanishing gradient problem, which prevents the model from learning long range dependencies. LSTM

Approved for public release; distribution is unlimited.

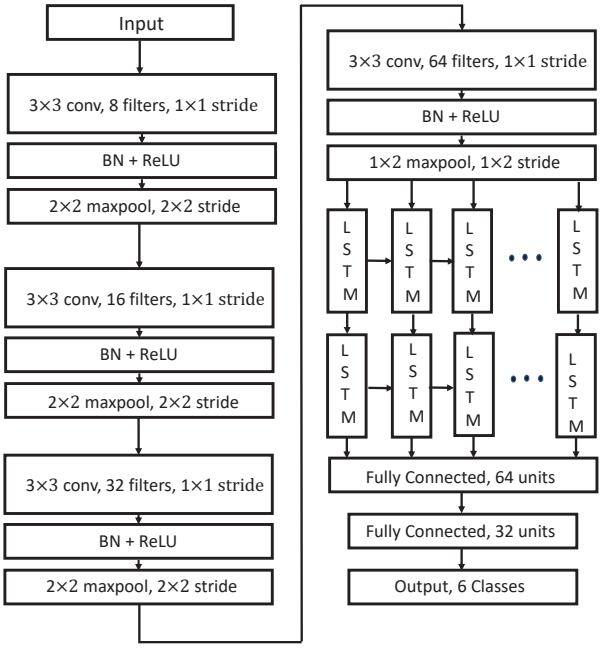

Figure 27. Proposed model architecture.

tackles this problem by introducing three gating mechanisms to update the memory cell c_t and hidden state h_t at each step t based on the current time step input x_t and the previous time step's hidden state output h_{t-1}. Each LSTM unit is composed of a memory cell and three main gates: input, output and forget. The input gate i_t, forget gate f_t, output gate o_t, memory cell c_t and hidden state h_t at step t are computed as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i \tag{18}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f \tag{19}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o \tag{20}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{21}$$

$$h_t = o_t \odot tanh(c_t) \tag{22}$$

Here, σ is the logistic sigmoid function, tanh is the hyperbolic tangent function, and $\odot$ denotes the element wise multiplication. By this architecture, the LSTM manages to create a controlled information flow by deciding which information it must forget and which information to remember. To understand the mechanism behind the architecture, we can view f_t as the function that controls to what extent the information from the old memory cell is going to be thrown away, i_t controls how much new information is going to be stored in the current memory cell, and o_t controls what to output based on the memory cell c_t.

Proposed Architecture

Our proposed architecture is comprised of four convolutional layers, followed by two LSTM layers and three fully connected dense layers. In Figure 27, we present our proposed architecture. The input to the networks is the Continuous Wavelet Transform (CWT) representations obtained from the bandpass filtered seismic signal waveforms. A detailed description

Approved for public release; distribution is unlimited.

of data preprocessing is presented in the following section. In each of the four convolutional layers, we use kernels of size 3×3 with a stride of 1×1 and zero paddings. Each convolution layer is followed by a Batchnormalization layer, a Rectified Linear Unit (ReLU), and a two-dimensional max-pool layer. In the first two max-pool layers, we use a kernel size of 2×2 with a stride of 2×2. However, in the last convolution layer, we perform max-pooling only along the temporal dimension with a kernel of size 1×2 and stride of 1×2. The first convolutional layer has eight filters, and we double the filter number on each subsequent convolutional layer to keep the number of parameters in each convolutional layer the same as we reduce the input image size by half after each convolution layer due to max-pooling.

The output from the final convolution layer is then passed into the LSTM layers preserving the temporal order. The first LSTM layer consists of 32 hidden units and the second LSTM layer consists of 16 hidden units. We use *sigmoid* and *tanh* as the recurrent and output activation function of the LSTM correspondingly. We use a 50% recurrent dropout in the LSTM layers. Both LSTM layers are unrolled for 15 steps as the input feature map to the LSTM has a temporal dimension of fifteen. Both LSTM layers return sequences in each unrolling step. These sequences are flattened before feeding into the dense layers. We stack three dense layers, each with 64, 32, and 6 hidden units consecutively. Each of the dense layers are preceded by Batchnormalization and ReLU activation functions with a 20% dropout rate. We use the softmax activation function in the final dense layer to obtain output probabilities for each phase-type.

3.4 Results and Discussion: Signal Classification

3.4.1 Dataset Description

The dataset is curated from 10 years of continuous seismic data collected at the 155 stations of the IMS. These consist of 46 primary stations, 24 of them were arrays, and 105 auxiliary stations, 98 of which are 3-component stations. These data, 80TB uncompressed, were obtained directly from the *United States National Data Center at the U.S. Air Force Technical Applications Center (AFTAC)*. This dataset includes the comprehensive IMS catalog, with arrival times and phase labels curated by human analysts for over 8 million seismic event detections. From these 8M seismic events, we filtered out 175K fine-grained seismic phase labeled data. However, among these, the P-phase was predominant, with 128,120 occurrences, while the S-phase had only 1,306 occurrences. To ensure a balanced dataset between crustal, regional, and teleseismic compressional and shear phase (i.e., Pg, Lg, Pn, Sn, P, and S), we used all labeled S-phases and randomly sampled around 2,500 waveforms from each of the other phases, for a total of 16,304 phases. As the spectro-temporal features of all the phases are highly nuanced, no data augmentation was performed to maintain integrity for practical application scenarios.

3.4.2 Data Prepossessing.

Our raw input data are 60 seconds long three-channel time-series, sampled at 40Hz. Following conventional seismic signal pre-processing techniques, we filter the waveform from

Approved for public release; distribution is unlimited.

each channel (0.4Hz to 10Hz). As these seismic signals were generated by events of different magnitude and recorded at stations spread across the globe, the amplitude is not relevant to phase identification in this first step. We first detrend each sample and remove the mean. We then max-normalize the data across each-channel, thus retaining the relative amplitudes among components of a station. Afterward, use the Continuous Wavelet Transform (CWT) to obtain spectral-temporal features representation.

Baseline Methods.

In order to validate the performance of our method we have compared our method with the following baseline methods.

- **XGBoost** (XGB) [13]: XGBoost is an optimized ensemble based model that has produced state-of-the-art methods for many classification tasks. We convert the multi-dimensional CWT representations into one-dimensional features as input for XGB. Afterward, we perform standardization across each feature dimension.
- **MLP** [24]: Multi Layer Perception (MLP) is a feed-forward neural network. We use a two-layer MLP with the same input features as XGB.
- **CNN** [18]: CNN based methods have been previously used in related seismic signal classification tasks. In [18], a CNN based method has been used for two-class phase classification. We use the same CNN architecture used in this paper to compare against our method.
- **LSTM** [27]: LSTM methods are highly suitable for temporal data modeling and have produced state of the art accuracy in many time series classification tasks. We use the most popular stacked LSTM architecture for comparison. The final output is fed into a fully connected layer to generate output labels.
- **CRED** [38]: In [38], a ResNet-BiLSTM architecture has been proposed for seismic event detection where it achieved state-of-the-art performance. We use the same architecture for our phase identification task.

3.4.3 Experimental Settings.

The hyper-parameters of the model were selected empirically by grid-search on the validation set. We use the Adam optimizer [30] with an initial learning rate of 0.01 and with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We apply $L2$ regularization with $\lambda = 0.001$, and we use *categorical cross-entropy* [23] as the loss function with a training batch size of 256. The XGBoost model was trained until convergence. The neural network models were trained for a maximum of 200 epochs with early stopping on the validation set.

We use 10-fold cross validation to measure the performance of our method, and report the average. In each fold, we use 80% of the data for training, 10% for validation and 10% for testing. We perform random stratification to ensure class balance in the training-validation-test split. All the experiments were performed on a core i5 2.70 GHz desktop computer with 8GB NVIDIA GeForce GTX-1070 GPU.

Approved for public release; distribution is unlimited.

Table 5. Performance metric comparison of FASER against baseline methods.

Method	PR	RL	F1	ACC
XGB	68.4	67.2	67.2	67.2
MLP	76.2	75.6	75.3	75.6
CNN	75.2	75.2	75.2	75.2
LSTM	75.7	74.3	75.0	75.3
CRED	81.3	80.2	80.7	81.5
FASER	**84.6**	**81.6**	**83.1**	**82.8**

FASER outperforms existing baseline methods in all four metrics: Precision(PR), Recall (RL), F1-Score(F1) and Accuracy (ACC).

Evaluation Metrics.

In our experiments, following conventional practices for classification tasks, we use accuracy as the primary performance metric. However, as there are minor class imbalances in the dataset, we also calculate macro (calculated individually for each class and averaged afterward) precision, recall, and F1-score [15]. Precision, recall, and f1-score are calculated based on true-positive(TP), false-positive(FP), and false-negative(FN) values using the following formulas.

$$precision = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN}$$

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

3.4.4 Results.

In Table 5, we report the performance of our method in comparison with the baseline methods. We observe that FASER consistently outperforms all the baseline methods across all performance metrics. To closely probe the performance of FASER across each class, in Figure 29(left), we show the confusion matrix for the test cases of a randomly split 80-10-10 train-validation-test scenario. It is noticeable that the majority of classification error is within the sub-classes of compressional (P, Pg, Pn) and transverse (S, Sn, Sg) waves. This performance is in coherence with the intuitive notion of similar spectral-temporal features within both broader classes.

In Figure 29(right), we plot the t-SNE visualization [34] of the same 10% test cases considering the activation values of the last layer before the prediction layer as deep embedding [76]. The compressional and transverse wave signal samples are well separated in the deep embedding space with high-margin with only a few mispositioned overlaps. However, as the spectral-temporal features within the sub-classes of translation and compressional waves are often overlapping, we notice soft-boundaries among the intra-sub-classes along with higher overlaps among the samples. Although the separation among finer-phase types is not well-established, it is evident from this embedding projection that our proposed method is adept at learning higher-order separable representations.

Approved for public release; distribution is unlimited.

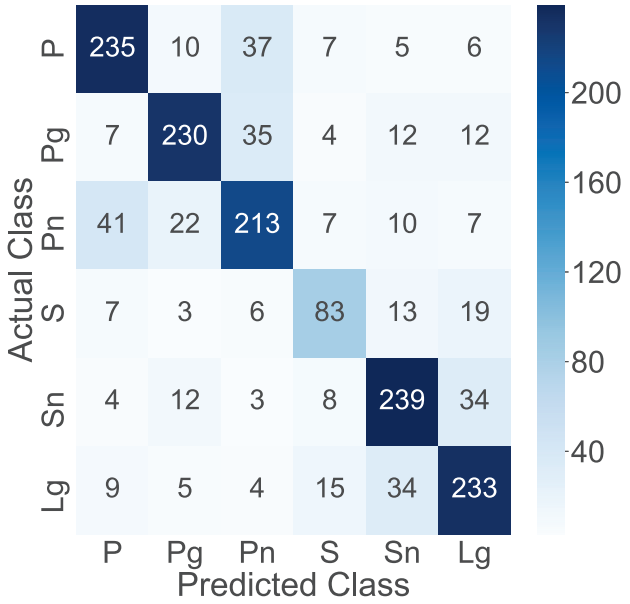

Figure 28. Confusion matrix for the test cases of a randomly split 80-10-10 train-validation test scenario.

3.4.5 Case Study: Novel Operating Conditions

In this section, we demonstrate practical use cases for our developed method in real-world applications when various novel scenarios emerge. In particular, we consider the following two novel scenarios: (1) If a monitoring agency adds a new station at a new location, will our method work without any calibration? (2) If new seismic sources occur in historically aseismic regions, will our method identify novel arrivals from a new source?

Performance at Novel Station

In our dataset, we have phase arrival signals recorded at 125 different stations across the world. To test how our method would perform at a novel station, we hold out signals at one station while training on signals at all other stations. We show empirical Cumulative distribution function (CDF) of stations for various accuracy levels in Figure 30. We compare two classifiers in this experiment. The nearest neighbor classifier that compares a test image with all training images to find the best match under the Euclidean norm, and labels the test image with the phase of the best match. The classifier achieves approximately default classification accuracy of 17% for the majority of the stations. This suggests that the nearest neighbor classifier cannot classify signals at a new station based on historical data at other stations. In contrast, FASER achieves an approximately 72% accuracy for the majority of the stations, suggesting single station analyses of seismic data may be useful in response to dynamically changing monitoring needs. The convolutional layers in our architecture extract local features from the images, unlike relying on a global one-to-one alignment of the images in the Euclidean space, as in the case of the nearest neighbor classifier.

The achieved accuracy of 72% for the majority of the station is significant for the IMS processing pipeline, as IDC analysts relabel 62% of the initial phases detected by the current

Approved for public release; distribution is unlimited.

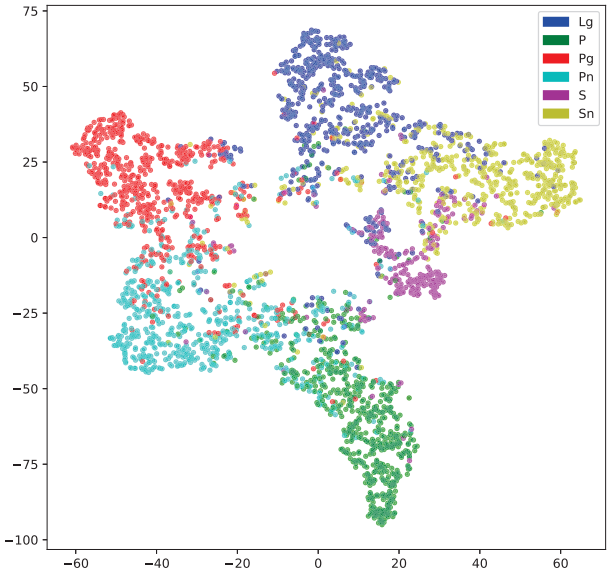

Figure 29. t-SNE visualization of the same 10% test cases considering the activation values of the last layer before the prediction layer as deep embedding [76]. The compressional and transverse wave signal samples are well separated.

automated algorithm. Moreover, only 38% of the initial phases remain the same, indicating that the initial phases are correct with 38% accuracy. FASER almost doubles the accuracy for a novel station of the current system's phase identification accuracy for an existing station. We show the held-out performance at each station in Figure 31(left). Most stations achieve higher accuracy (>0.7) when there are several closer stations. In contrast, isolated stations such as the one in the South Pacific suffer from a poorer performance.

Performance on Novel Sources

Most earthquakes originate along fault lines, while the rest of the earth is quieter. Novel seismicity in previously undocumented areas is intriguing. Hence, we evaluate our model by holding out regions of the earth for testing. For each held-out region, we train our model with data from the rest of the world and test the performance of our model on seismic events in that region. For this experiment, we divide the earth into 12× 12-degree grids. If a grid cell is not seismically active (i.e., not enough data), we exclude the region from testing. In Figure 31(right), we show the world map where the shaded grid cells are held out, one at a time. The average hold-out accuracy is 77.27%, with a standard deviation of 4.13%. More importantly, this suggests that our model is well suited for novel seismicity with little or no prior recorded events. We test on 46 cells of the 12×12 degree grids, which cover most of the known seismic events recorded at the NEIC (National Earthquake Information Center) for a three year period.

Approved for public release; distribution is unlimited.

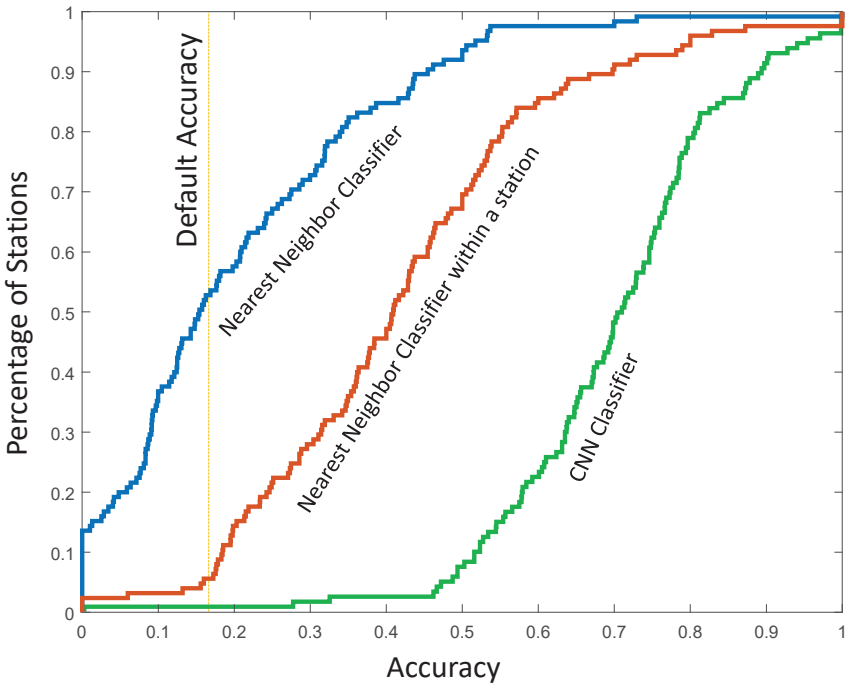

Figure 30. Performance comparison of FASER with naive nearest neighbor classifier for application in novel stations. *The plot shows cumulative distribution function (CDF) for percentage of stations having smaller than a given accuracy. FASER achieves an approxi-*
mately 72% accuracy for the majority of the stations in contrast to 17% for nearest neighbor classifier.

3.5 Conclusion: Signal Classification

In this part of the project, we present a method to perform fine-grained seismic phase identification, which can be readily integrated into existing seismic signal processing pipelines. As seismology evolves into a big-data-driven science, deep learning methods are becoming an indispensable part of next-generation seismic monitoring systems. This work shows a practical example of integrating deep-learning methods in an existing semi-autonomous system to achieve complete autonomy. We demonstrate empirical evaluation of our method with a real-world dataset where it outperforms existing methods. Our method reduces the de-pendency on using array-based methods, which inhibits precise monitoring for regions with limited monitoring stations. It also reduces the dependency on large-collections of manually curated template sets and presents the opportunity of using transfer-learning for stations with limited labeled data. Due to the minimal preprocessing requirements and faster prediction generation, it is highly suitable for a real-time monitoring pipeline. In the future, with the use of larger datasets, more complex models would produce higher accuracy as well as better generalizability. Moreover, the introduction of interpretable models would be highly suitable for downstream analysis.

Approved for public release; distribution is unlimited.

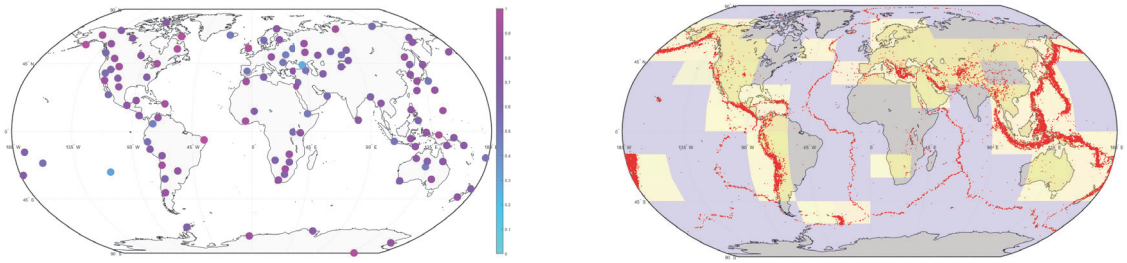

Figure 31. (Left) Testing with held-out stations. Each dot is an IMS station. The color bar shows held-out accuracy for a station. (Right) Testing with held-out regions. Each red dot is an event recorded at the NEIC (National Earthquake Information Center). We hold out shaded regions. Average hold-out accuracy across shaded regions is 77.27%, with a standard deviation of 4.13%, suggesting model efficacy at novel source regions.

Approved for public release; distribution is unlimited.

Chapter 4

Seismic Depth Prediction Using Attention Network

4.1 Summary: Depth Estimation

The depth of seismic events is an important feature to discriminate natural earthquakes from events induced or caused by humans. Depth is best calculated when a seismic station is located exactly above the event origin. However, estimating the depth of *any* event with a sparse set of stations is a daunting task, and there is no globally viable method. In this last part of the project, we focus on developing a not-so-deep machine learning model to accurately estimate the depth of arbitrary events. The main advantage of not-so-deep models as proposed in this work is the lower number of parameters to tune and better interpretability. Our model is crafted to retain interpretability for geophysicists while exploiting machine learning's power in generalizing to unreachable locations on earth. We use the SCEDC (Southern California Earthquake Data Center) catalog generated for events in California and demonstrate performance on these local and near regional events.

4.2 Introduction: Depth Estimation

Accurate depth estimation of a seismic event is critical to discriminating between man-made and natural seismic events, simply because nearly all man-made events are less than one kilometer deep. Distinguishing man-made events from natural events has several applications in nuclear non-proliferation, security of underground assets, such as optical fibre, among many others.

Theoretically, the depth of a seismic event is estimated by inverting the travel time equations to individual stations. However, the correctness (or uncertainty) in depth estimation largely depends on the locations of the stations and the number of them. In this project, we consider estimating depths directly from the wave form with the help of modern machine learning techniques.

Approved for public release; distribution is unlimited.

4.3 Methods, Assumptions, and Procedures: Depth Estimation

CNN-LSTM model

Convolutional Neural Networks (CNNs) are widely used deep networks. CNNs perform convolutions on input images with multiple fixed sized kernels. A convolution operation can be seen as sliding the kernel over the image and computing the dot product at each step. Each convolution extracts different higher order representations and form a feature map. A convolutional layer is usually followed by a nonlinear activation function (such as ReLU) and a max-pooling function. Long-Short Term Memory (LSTM) networks resolve vanishing and exploding gradient problems of vanilla RNNs and are able to capture long range temporal dependencies by introducing input and forget gates.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i \tag{23}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f \tag{24}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o \tag{25}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{26}$$
$$h_t = o_t \odot tanh(c_t) \tag{27}$$

Our CNN-LSTM network consists of four CNN layers followed by two LSTM layers followed by three dense layers. Each of the CNN layers is followed by a batch normalization layer and a ReLU activation layer. 30% dropout is applied after ReLU activation layers. We used 8, 16, 32 and 64 3×3-size kernels for convolution layers and the output from the fourth convolution layer is fed to the first LSTM layer after a time preserving transformation.

Attention Network with Hierarchy

Attention networks are widely used in Natural Language Processing to identify important regions of the input data. Such networks assign a many-to-many attention mapping on input and output sequences.

$$h_{it} = tanh(W_a c_{it} + b_a) \tag{28}$$
$$\alpha_{it} = \frac{exp(h_{it}^T v_a)}{\sum_t exp(h_{it}^T v_a)} \tag{29}$$
$$o_i = \sum_t \alpha_{it} c_{it} \tag{30}$$

Here, c_{it} is a time preserving input sequence which is fed through a single-layer perceptron with a hyperbolic tangent function to get hidden representation h_{it}. α_{it} is the normalized attention weight which denotes the weight of the current time-step, and o_{it} is the output computed by multiplying the input and attention.

Approved for public release; distribution is unlimited.

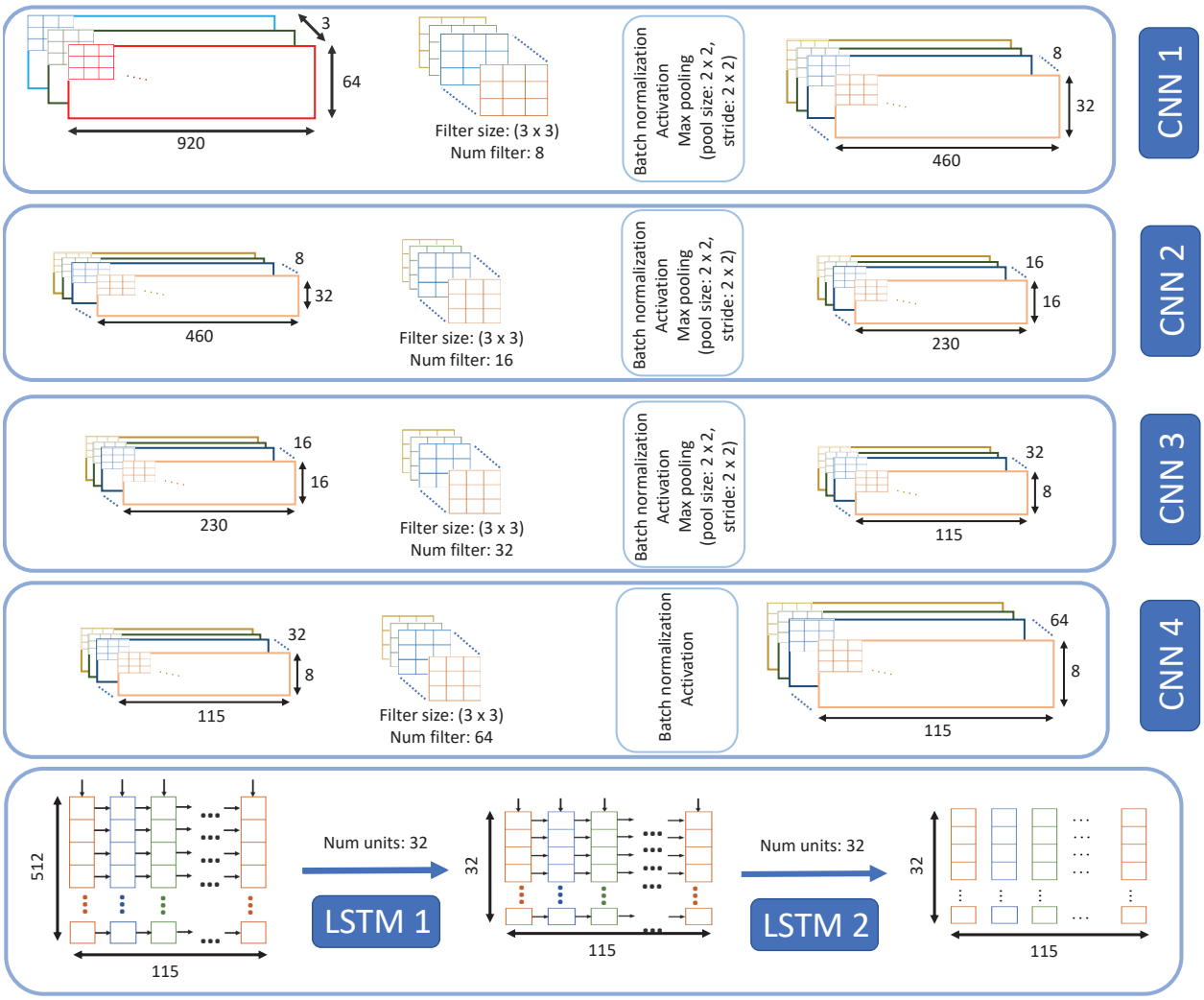

Figure 32. *Waveform encoder* consists of three CNN and two LSTM layers.

Our proposed attention network has two levels of hierarchy and two attention layers in each hierarchy. The first hierarchy is *waveform encoder*. The input to this level is linearly spaced continuous wavelet transforms (CWT) generated from three-channel waveforms. Similar to the CNN-LSTM model, four CNN layers followed by two LSTM layers are stacked in this level. Time preserved output from the last LSTM layer is fed into the attention layers. Output from multiple attention layers (each from a different station) are aggregated and fed into the *station encoder* which is second level in our hierarchical structure. Meta information for each corresponding station (i.e. station location, etc.) is aggregated to the attention vectors. The *station encoder* consists of one CNN layer, one LSTM layer, and an attention layer. The CNN layer in *station encoder* has 16 kernels of 3×3 size and is followed by a ReLU activation and a max pooling layer. Output from the CNN layer is fed into the LSTM layer which has 32 hidden units. The attention layer in *station encoder* captures the weights at each timestamp of the input data. The *station encoder* model ends with three dense layers with 64, 32 and 1 output units.

Approved for public release; distribution is unlimited.

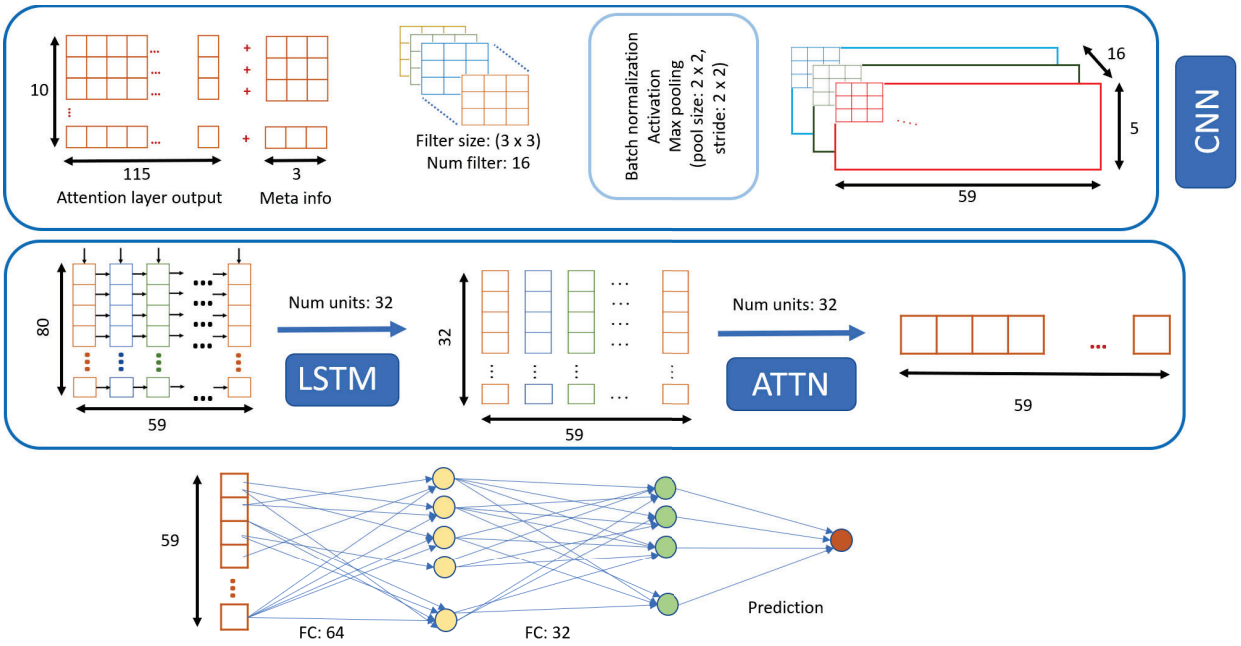

Figure 33: *Station encoder* consists of a CNN and an LSTM layers. *An attention layer is used to get weighted sequence. Three fully connected layers are used to get the final prediction.*

4.3.1 Data Description

We use a relocated earthquake catalog from the Southern California Earthquake Data Center (SCEDC). This earthquake catalog used 3D and Growclust algorithms for the relative relocation of earthquake hypocenters based on waveform cross-correlation data. The catalog includes uncertainty bounds for the depths and the reported median vertical uncertainty is only 0.4 km. Such uncertainty for depth is considered low, which makes this dataset a good candidate for our experimental evaluation. The catalog contains 75,000 earthquake events with magnitudes between 2.0 and 4.0 from the year 1981 to 2019. We collected more than 650K multi-channel waveforms recorded by 423 densely located seismic stations in Southern California.

We filter out any earthquake event for which we did not find a station within 1.2 times the reported depth. This filtering was needed as the depth related information in a waveform starts to lose resolving power once the distance between the epicenter and the recording station is higher than 1.2 times of reported depth. We also filter out any earthquake event for which the SDEDC data center does not have waveforms from at least 10 different stations. We finalize a total of 6,560 earthquake events; each having waveforms from three broadband channels recorded at 10 or more stations.

We collect 230 seconds of waveform to accommodate all information of a regional earthquake event. We start at 30 seconds before the first P arrival time and end at 200 seconds after the P arrival time. At a 40Hz sampling rate, the length of each waveform is 9,200 data points.

Approved for public release; distribution is unlimited.

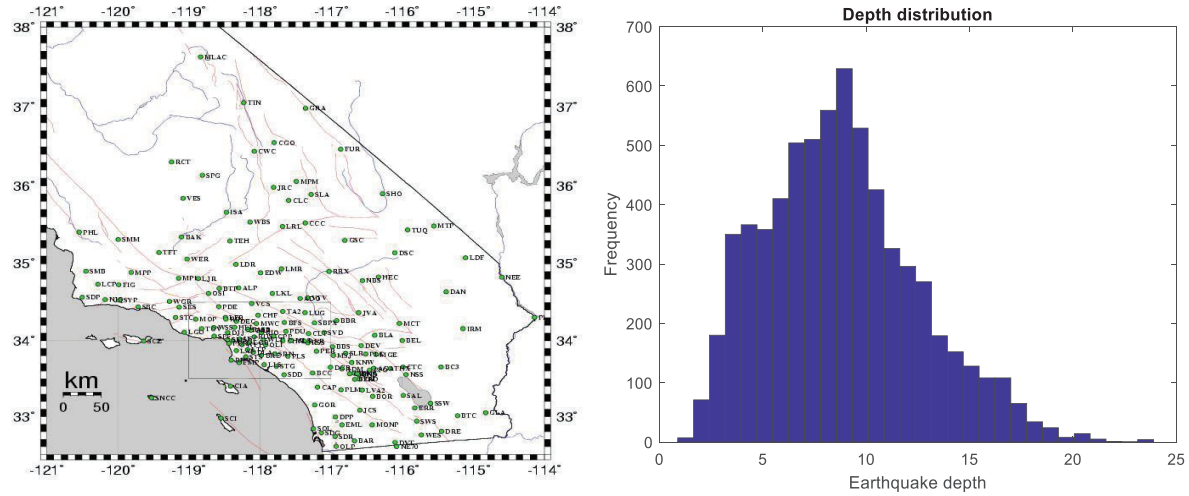

Figure 34. (left) Map of Broadband stations in Southern California shows a dense seismic network. (right) Distribution of earthquake depth of our dataset.

4.3.2 Waveform Preprocessing

Our raw input data are 230-second long three-channel time-series, sampled at 40 Hz. Following conventional seismic signal preprocessing techniques, we remove any instrument response and convert the horizontal, vertical north-south, and vertical east-west components (Z, N, E) to horizontal, radial and tangential components (Z, R, T). We filter the waveform from each channel (0.4Hz to 10Hz). We detrend each sample and remove the mean. We then max-normalize the data across each-channel, thus retaining the relative amplitudes among components of a station. We use the 64 scale Continuous Wavelet Transform (CWT) to obtain spectral-temporal feature representation in an RGB image. Each RGB image is 64×920×3, where the vertical component is represented in red, the radial component by green and the tangential component by blue. Figure 35 shows how a linearly spaced CWT captures spectral and temporal features from the 3-channel waveforms.

4.4 Results and Discussion: Depth Estimation

We consider Root Mean Square Error (RMSE) as a loss function and Stochastic Gradient Descent (SGD) as the optimizer to train our model. The learning rate is set to 0.01 with a linear decay of 0.1 per epoch. Both dropout ratio and recurrent dropout ratio for LSTM layers are set to 0.3 for all models. These values are set based on preliminary evaluations on training sets. We consider a dataset split of 80/10/10 for training, validation, and testing after random shuffling inputs to all experiments. The results are generated by taking the average of five separate training sets using five-fold cross-validation after 300 epochs of training. For our evaluation, we consider the RMSE of the predicted depth. We also calculate Pearson's correlation coefficient to show the relationship between the predicted and actual depth qualitatively. We use a GPU server with four Nvidia RTX2080 GPUs, 256 GB of RAM, and a 32-core CPU.

Our model achieves root mean squared error of 2.89km in predicting the depth of a

Approved for public release; distribution is unlimited.

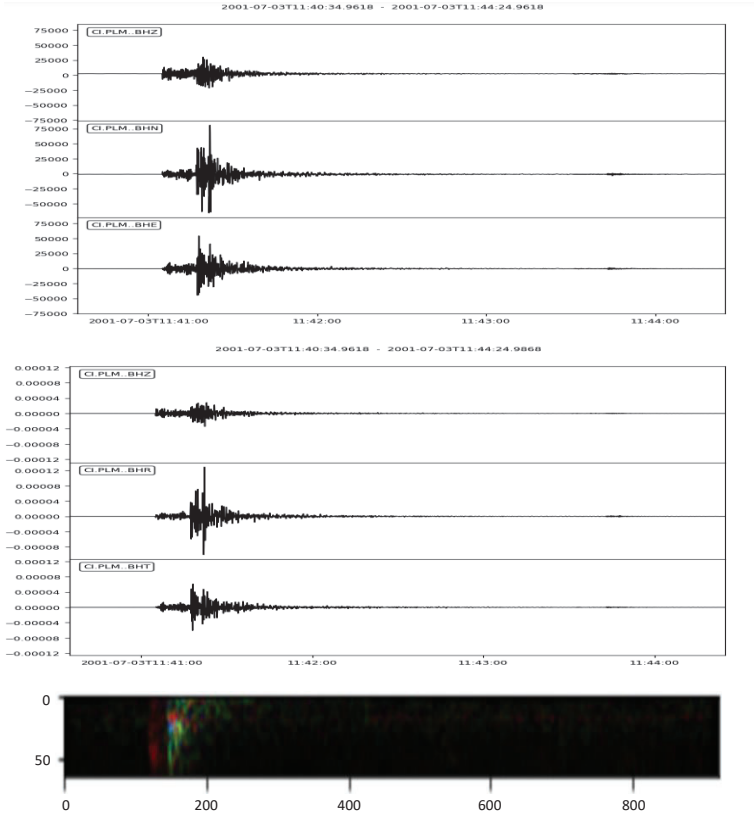

Figure 35. (a) 3-channel raw waveform collected from station CI.PLM (b) vertical, tangential and radial components generated from the raw waveforms (c) linearly spaced CWT image from the ZRT channels

regional earthquake, which is lower than one standard deviation of 3.86km in the training data. Pearson's correlation coefficient between predicted and actual depth is 0.62 which shows moderate positive correlation.

4.5 Conclusion: Depth Estimation

Depth estimation, even at regional level, is a difficult task. In this project, we have begun exploring machine learned models for depth estimation. Future work will involve extensive experiments by varying input conditions, model architectures, evaluation metrics and seismic networks.

Approved for public release; distribution is unlimited.

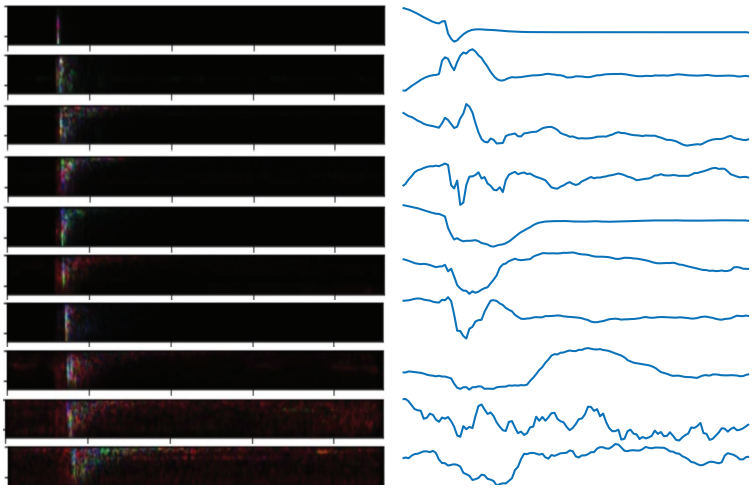

Figure 36. Linearly spaced continuous wavelet transforms and their corresponding waveform encoder attention vectors for a randomly picked earthquake

Approved for public release; distribution is unlimited.

References

[1] Deepdetect dataset public repository, https://sites.psu.edu/chasbolton/.

[2] Incorporated research institutions for seismology, https://www.iris.edu/hq/.

[3] Ncedc (2014), Northern California Earthquake Data Center, UC Berkeley Seismological Laboratory, dataset, doi:10.7932/ncedc. http://ncedc.org/.

[4] Public github repository to download code, spread- sheet, slides and datasets, 2019, https://github.com/aukdd/SeiSMo.

[5] Scedc, Southern California Earthquake Data Center, California Institute of Technology, dataset, http://scedc.caltech.edu.

[6] Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., and Yu, D., Convolutional Neural Networks for Speech Recognition, *IEEE/ACM Transactions on Audio, Speech, and Language Processing 22*, 10 (2014), pp. 1533–1545.

[7] Ahmad, Y. and Nath, S. Colr-tree, Communication-Efficient Spatio-Temporal Index- Ing for a Sensor Data Web Portal. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE),* Cancun, Mexico, 2008, IEEE, pp. 784–793.

[8] Allen, R. V., Automatic Earthquake Recognition and Timing from Single Traces. *Bulletin of the Seismological Society of America 68*, 5 (1978), pp. 1521–1532.

[9] Basu, S., Banerjee, A., and Mooney, R. J., Semi-supervised Clustering by Seeding, in *Proceedings of the Nineteenth International Conference on Machine Learning* , San Francisco, CA, USA, 2002, ICML '02, Morgan Kaufmann Publishers Inc., pp. 27– 34.

[10] Begum, N. and Keogh, E., Rare Time Series Motif Discovery from Unbounded Streams, *Proceedings of the VLDB Endowment 8*, 2 (10 2014), pp. 149–160.

[11] Breunig, M. M., Kriegel, H.-P., Ng, R. T., Sander, J., Breunig, M. M.,Kriegel, H.-P., Ng, R. T., and Sander, J. LOF, Identifying Density-Based Local Outliers, in *Proceedings of the 2000 ACM SIGMOD International Conference on Man-Agement of Data - SIGMOD '00,* New York, NY, USA, 2000), vol. 29, ACM Press, pp. 93–104.

[12] Cavanagh, J. F., Kumar, P., Mueller, A. A., Richardson, S. P., and Mueen, A., Diminished EEG Habituation to Novel Events Effectively Classifies Parkinson's Patients, *Clinical Neurophysiology 129*, 2 (2018), pp. 409–418.

[13] Chen, T. and Guestrin, C. Xgboost, A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 785–794.

Approved for public release; distribution is unlimited.

[14] Chen, Y., Hu, B., Keogh, E., and Batista, G. E. DTW-D, Time Series Semi-supervised Learning from a Single Example, In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '13,* New York, NY, USA, Aug. 2013, ACM Press, p. 383.

[15] Chowdhury, F. A., Suzuki, S., and Mueen, A. Structured noise detection: Application on well test pressure derivative data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 2952–2960.

[16] Dau, H. A. and Keogh, E., Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery, In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17,* New York, NY, USA, 2017, ACM Press, pp. 125–134.

[17] Dickey, J., Borghetti, B., Junek, W., and Martin, R., Beyond Correlation: A Path-Invariant Measure for Seismogram Similarity, *Seismological Research Letters* (Nov 2019).

[18] Dokht, R. M., Kao, H., Visser, R., and Smith, B., Seismic Event and Phase Detection Using Time–Frequency Representation and Convolutional Neural Networks, *Seis-mological Research Letters 90*, 2A (2019), pp. 481–490.

[19] Dos Santos, C. and Gatti, M., Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts, In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (2014), pp. 69–78.

[20] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, 1996.

[21] García, L., Álvarez, I., Benítez, C., Titos, M., Bueno, Á., Mota, S., De la Torre, A., Segura, J. C., Alguacil, G., Díaz-Moreno, A., et al., Advances on the Automatic Estimation of the P-Wave Onset Time, *Annals of Geophysics 59*, 4 (2016), 0434.

[22] Gibbons, S. J. and Ringdal, F., The Detection of Low Magnitude Seismic Events Using Array-Based Waveform Correlation, *Geophysical Journal International 165*, 1 (2006), pp. 149–166.

[23] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning,* MIT press, 2016.

[24] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,* Springer Science & Business Media, 2009.

[25] He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

Approved for public release; distribution is unlimited.

[26] Hickish, J., Razavi-Ghods, N., Perrott, Y. C., Titterington, D. J., Carey, S. H., Scott, P. F., Grainge, K. J. B., Scaife, A. M. M., Alexander, P., Saunders, R. D. E., Crofts, M., Javid, K., Rumsey, C., Jin, T. Z., Ely, J. A., Shaw, C., Northrop, I. G., Pooley, G., D'Alessandro, R., Doherty, P., and Willatt, G. P., A Digital Correlator Upgrade for the Arcminute MicroKelvin Imager, *Monthly Notices of the Royal Astronomical Society 475*, 4 (4 2018), pp. 5677–5687.

[27] Hochreiter, S. and Schmidhuber, J., Long Short-Term Memory, *Neural Computation 9* (1997), pp. 1735–1780.

[28] Hu, B., Lu, Z., Li, H., and Chen, Q., Convolutional Neural Network Architectures for Matching Natural Language Sentences, In *Advances in Neural Information Processing Systems* (2014), pp. 2042–2050.

[29] Karim, F., Majumdar, S., Darabi, H., and Chen, S., LSTM Fully Convolutional Networks for Time Series Classification, *IEEE access 6* (2017), pp. 1662–1669.

[30] Kingma, D. P. and Ba, J. Adam, A Method for Stochastic Optimization, *arXiv preprint arXiv, 1412.6980* (2014).

[31] Krizhevsky, A., Sutskever, I., and Hinton, G. E., Imagenet Classification with Deep Convolutional Neural Networks, In *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105.

[32] Küperkoch, L., Meier, T., Lee, J., Friederich, W., and Group, E. W., Automated Determination of P-Phase Arrival Times at Regional and Local Distances Using Higher Order Statistics, *Geophysical Journal International 181*, 2 (2010), pp. 1159–1170.

[33] Langet, N., Maggi, A., Michelini, A., and Brenguier, F., Continuous Kurtosis-Based Migration for Seismic Event Detection and Location, with Application to Piton De La Fournaise Volcano, La Réunioncontinuous Kurtosis-Based Migration for Seismic Event Detection and Location, *Bulletin of the Seismological Society of America 104*, 1 (2014), pp. 229–246.

[34] Maaten, L. v. d. and Hinton, G., Visualizing Data Using T-SNE, *Journal of Machine Llearning Research 9*, Nov (2008), pp. 2579–2605.

[35] McBrearty, I. W., Gomberg, J., Delorey, A. A., and Johnson, P. A., Earthquake Arrival Association with Back Projection and Graph Theory.

[36] Mohamed, A.-r., Hinton, G., and Penn, G., Understanding How Deep Belief Networks Perform Acoustic Modelling, In *2012 IEEE International Conference on Acoustics,Speech and Signal Processing (ICASSP)* (2012), IEEE, pp. 4273–4276.

[37] Mousavi, S. M., Langston, C. A., and Horton, S. P., Automatic Microseismic Denoising and Onset Detection Using the Synchrosqueezed Continuous Wavelet Transform, *GEOPHYSICS 81*, 4 (7 2016), pp. V341–V355.

Approved for public release; distribution is unlimited.

[38] Mousavi, S. M., Zhu, W., Sheng, Y., and Beroza, G. C., CRED: A Deep Residual Network of Convolutional and Recurrent Units for Earthquake Signal Detection, *Scientific Reports 9*, 1 (12 2019), p.10267.

[39] Mueen, A., Enumeration of Time Series Motifs of All Lengths, In *Proceedings - IEEE International Conference on Data Mining, ICDM* (2013), ICDM, pp. 547–556.

[40] Mueen, A. and Keogh, E., Online Discovery and Maintenance of Time Series Motifs, In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '10,* New York, NY, USA, July 2010, ACM Press, p. 1089.

[41] Mueen, A., Keogh, E., and Bigdely-Shamlo, N., Finding Time Series Motifs in Disk-Resident Data, In *Proceedings - IEEE International Conference on Data Mining, ICDM* (2009), pp. 367–376.

[42] Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B., Exact Discovery of Time Series Motifs, In *Proceedings of the 2009 SIAM International Conference on Data Mining* (2009), pp. 473–484.

[43] Mueen, A., Nath, S., and Liu, J., Fast Approximate Correlation for Massive Time-Series Data, in *Proc. ACM SIGMOD Int. Conf. on Management of Data* (2010), pp. 171–182.

[44] Mueen, A., Viswanathan, K., Gupta, C., and Keogh, E., The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance, Aug. 2015.

[45] Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, 1 ed. Prentice Hall, 1975.

[46] O'Rourke, C., Sheehan, A., Erslev, E., and Miller, K., Estimating B asin Thickness Using a High-Density Passive-Source Geophone Array, *Earth and Planetary Science Letters 402* (2014), pp. 120–126.

[47] O'Rourke, C. T., Baker, G. E., and Sheehan, A. F., Using P/S Amplitude Ratios for Seismic Discrimination at Local Distances Using P/S Amplitude Ratios for Seismic Discrimination at Local Distances, *Bulletin of the Seismological Society of America 106*,5 (2016), p. 2320.

[48] Ossher, H. and Tarr, P., Using Multidimensional Separation of Concerns to (Re) Shape Evolving Software, *Communications of the ACM 44*, 10 (2001), pp. 43–50.

[49] Parseval, M.-A., Mémoire sur les séries et sur l'intégration complète d'une équation aux différences partielles linéaires du second ordre, à coefficients constants, *Mém. prés. par divers savants, Acad. des Sciences, Paris,(1) 1* (1806), pp. 638–648.

[50] Perol, T., Gharbi, M., and Denolle, M., Convolutional Neural Network for Earthquake Detection and Location.

[51] Perol, T., Gharbi, M., and Denolle, M., Convolutional Neural Network for Earthquake Detection and Location, *Science Advances 4*, 2 (2 2018), p. e1700578.

Approved for public release; distribution is unlimited.

[52] Preparata, F. P. and Shamos, M. I., *Computational Geometry,* Springer New York, NY, 1985.

[53] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E., 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.

[54] Rakthanmanon, T. and Keogh, E., Fast shapelets: A Scalable Algorithm for Discovering Time Series Shapelets, In *Proceedings of the thirteenth SIAM Conference,* (2013), pp. 668–676.

[55] Rawles, C. and Thurber, C., A Non-Parametric Method for Automatic Determination of P-Wave and S-Wave Arrival Times: Application to Local Micro Earthquakes, *Geo- physical Journal International 202*, 2 (2015), pp. 1164–1179.

[56] Reading, A. M., Mao, W., and Gubbins, D., Polarization Filtering for Automatic Picking of Seismic Data and Improved Converted Phase Detection, *Geophysical Journal International 147*, 1 (2001), pp. 227–234.

[57] Ross, Z. E., Meier, M.-A., Hauksson, E., and Heaton, T. H., Generalized Seismic Phase Detection with Deep Learning, *Bulletin of the Seismological Society of America 108*, 5A (2018), pp. 2894–2901.

[58] Ross, Z. E., Yue, Y., Meier, M.-A., Hauksson, E., and Heaton, T. H., PhaseLink: A Deep Learning Approach to Seismic Phase Association, 2018.

[59] Sainath, T. N., Vinyals, O., Senior, A., and Sak, H., Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks, In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), IEEE, pp. 4580–4584.

[60] Sakurai, Y., Papadimitriou, S., and Faloutsos, C.,Braid: Stream Mining Through Group Lag Correlations, In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (2005), p. 610.

[61] Sax, M. J., *Apache Kafka,* Springer International Publishing, Cham, 2018, pp. 1–8.

[62] Schaff, D. P., Bokelmann, G. H., Beroza, G. C., Waldhauser, F., and Ellsworth, W. L., High-Resolution Image of Calaveras Fault Seismicity, *Journal of Geophysical Research: Solid Earth 107*, B9 (2002), ESE–5.

[63] Senobari, N. S., Funning, G. J., Keogh, E., Zhu, Y., Yeh, C. M., Zimmerman, Z., and Mueen, A., Super-Efficient Cross-Correlation (SEC-C): A Fast Matched Filtering Code Suitable for Desktop Computers, *Seismological Research Letters 90*, 1 (1 2019), pp. 322–334.

[64] Shearer, P. M., *Introduction to seismology,* Cambridge University Press, 2019.

Approved for public release; distribution is unlimited.

[65] Shieh, J., iSAX: Indexing and Mining Terabyte Sized Time Series, In *Work* (2008), vol. KDD '08, pp. 623–631.

[66] Siddiquee, M. A., Akhavan, Z., and Mueen, A., Seismo: Semi-Supervised Time Series Motif Discovery for Seismic Signal Detection, in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019), pp. 99–108.

[67] Silva, D. F., Souza, V. M. A., Ellis, D., Keogh, E., and Batista, G. E. A. P. A., Exploring Low Cost Laser Sensors to Identify Flying Insect Species, *Journal ofIntelligent & Robotic Systems 80*, 1 (2015), pp. 313–330.

[68] Sinha, S., Routh, P. S., Anno, P. D., and Castagna, J. P., Spectral Decom-Position of Seismic Data with Continuous-Wavelet Transform, *Geophysics 70*, 6 (2005), pp. P19–P25.

[69] Stefan, A., Athitsos, V., and Das, G., The Move-Split-Merge Metric for Time Series, *IEEE Transactions on Knowledge and Data Engineering 25*, 6 (June 2013), pp. 1425–1438.

[70] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E., Indexing Multi-dimensional Time-series with Support for Multiple Distance Measures, In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* New York, NY, USA, 2003, KDD '03, ACM, pp. 216–225.

[71] Walsh, F. and Zoback, M., Oklahoma's Recent Earthquakes and Saltwater Disposal: Science Advances, 1, e1500195, 2015.

[72] Wei, L. and Keogh, E., Semi-Supervised Time Series Classification, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '06,* New York, NY, USA, 2006, ACM Press, p. 748.

[73] Weingarten, M., Ge, S., Godt, J. W., Bekins, B. A., and Rubinstein, J. L., High-Rate Injection is Associated with the Increase in US Mid-Continent Seismicity, *Science 348*, 6241 (2015), pp. 1336–1340.

[74] Worthington, L. L., Miller, K. C., Erslev, E. A., Anderson, M. L., Cham-berlain, K. R., Sheehan, A. F., Yeck, W. L., Harder, S. H., and Siddoway, C. S., Crustal Structure of the Bighorn Mountains Region: Precambrian Influence on Laramide Shortening and Uplift in North-Central Wyoming, *Tectonics 35*, 1 (2016), pp. 208–236.

[75] Wu, Y., Lin, Y., Zhou, Z., Bolton, D. C., Liu, J., and Johnson, P., Deep Detect: A Cascaded Region-Based Densely Connected Network for Seismic Event Detection, *IEEE Transactions on Geoscience and Remote Sensing*, 99 (2018), pp. 1–14.

[76] Xie, J., Girshick, R., and Farhadi, A., Unsupervised Deep Embedding for Clustering Analysis, In *International Conference on Machine Learning* (2016), pp. 478–487.

Approved for public release; distribution is unlimited.

[77] Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c., Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems* (2015), pp. 802–810.

[78] Yagoubi, D. E., Akbarinia, R., Kolev, B., Levchenko, O., Masseglia, F., Valduriez, P., and Shasha, D., ParCorr: Efficient Parallel Methods to Identify Similar Time Series Pairs Across Sliding Windows, *Data Mining and Knowledge Discovery 32*, 5 (9 2018), pp. 1481–1507.

[79] Yeck, W. L., Sheehan, A. F., Anderson, M. L., Erslev, E. A., Miller, K. C., and Siddoway, C. S., Structure of the Bighorn Mountain Region, Wyoming, from Teleseismic Receiver Function Analysis: Implications for the Kinematics of Laramide Short- Ending, *Journal of Geophysical Research: Solid Earth 119*, 9 (2014), pp. 7028–7042.

[80] Yeh, C.-C. M., Kavantzas, N., and Keogh, E., Matrix Profile VI: Meaningful Multidimensional Motif Discovery, in *2017 IEEE International Conference on Data Mining (ICDM)* (11 2017), IEEE, pp. 565–574.

[81] Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A., and Keogh, E., Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets, in *2016 IEEE 16th International Conference on Data Mining (ICDM)* (12 2016), IEEE, pp. 1317–1322.

[82] Yoon, C. E., O'Reilly, O., Bergen, K. J., and Beroza, G. C., Earthquake Detection Through Computationally Efficient Similarity Search, *Science Advances 1*, 11 (12 2015), p. e1501057.

[83] Zhang, H., Thurber, C., and Rowe, C., Automatic P-Wave Arrival Detection and Picking with Multiscale Wavelet Analysis for Single-Component Recordings, *Bulletin of the Seismological Society of America 93*, 5 (2003), pp. 1904–1912.

[84] Zhong, S., Souza, V. M. A., and Mueen, A., Supporting website, 2020, https://sites.google.com/view/filcorr/.

[85] Zhou, C., Sun, C., Liu, Z., and Lau, F. A., C-LSTM Neural Network for Text Classification, *ARXIV Preprint ARXIV:1511.08630* (2015).

[86] Zhu, Y. and Shasha, D., StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time, in *Proceedings of the 28th International Conference on Very Large Data Bases* (2002), vol. 54 of *VLDB '02*, pp. 358–369.

[87] Zhu, Y., Zimmerman, Z., Senobari, N., Yeh, C.-C., Funning, G., Mueen, A., Brisk, P., and Keogh, E., Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins, in *Proceedings- IEEE International Conference on Data Mining, ICDM* (2017).

Approved for public release; distribution is unlimited.

List of Symbols, Abbreviations, and Acronyms

ACC	Accuracy
AFRL	Air Force Research Laboratory
AFTAC	Force Technical Applications Center
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
F1	F1 Score
FASER	Phase Identifier
FAST	Fingerprint and Similarity Thresholding
FFT	Fast Fourier Transform
FILCORR	Filtered Lagged Correlation
FN	False Negative
FP	False Positive
IDC	International Data Centre
IMS	International Monitoring System
IRIS	Incorporated Research Institutions for Seismology
KMP	Knuth-Morris-Pratt
LSTM	Long-Short Term Memory
LTSA	Long Term Average
NCEDC	Northern California Earthquake Data Center
NEIC	National Earthquake Information Center
NNG	Nearest Neighbor Graph
P	Compressional
PR	Precision
RELU	Rectified Linear Unit
RL	Recall
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RS	Radial Search
S	Shear
SeiSMo	Semi-supervised Motif
SeisViz	Seismic Visualization

Approved for public release; distribution is unlimited.

SCEDC	Southern California Earthquake Data Center
SCSN	Southern California Seismic Network
SGD	Stochastic Gradient Descent
SNR	Signal to Noise Ratio
STA	Short Term Average
TP	True Positive

Approved for public release; distribution is unlimited.

DISTRIBUTION LIST

DTIC/OCP
8725 John J. Kingman Rd, Suite 0944
Ft Belvoir, VA 22060-6218 1 cy

AFRL/RVIL
Kirtland AFB, NM 87117-5776 1 cy

Official Record Copy
AFRL/RVB/Dr. Frederick R. Schult 1 cy

Approved for public release; distribution is unlimited.