Online Amnestic Dynamic Time Warping to Allow Real-Time Golden Batch Monitoring

Chin-Chia Michael Yeh, Yan Zhu, Hoang Anh Dau, Amirali Darvishzadeh, Mikhail Noskov[†], Eamonn Keogh
University of California, Riverside, [†]Aspen Technology
{myeh003, yzhu015, hdau001}@ucr.edu, [†]mike.noskov@aspentech.com,{darvisha, eamonn}@cs.ucr.edu

ABSTRACT

In manufacturing, there is the Golden Batch concept. A golden batch is an idealized realization of the perfect process to produce the desired item, typically represented as a multidimensional time series of temperatures, pressures, flow-rates and so forth. The golden batch is sometimes produced from first-principle models, but it is typically created by recording a batch produced by the most experienced engineers on carefully cleaned and calibrated machines. In most cases, the golden batch is only used in post-mortem analysis of an unexpectedly inferior quality product as plant managers attempt to understand where and when the last production attempt went wrong. In this work, we make two contributions to golden batch processing. We introduce an online algorithm that allows practitioners to understand if the process is currently deviating from the golden batch in real-time, allowing engineers to intervene and potentially save the batch. This may be done, for example, by cooling a boiler that is running unexpectedly hot. In addition, we show that our ideas can greatly expand the purview of golden batch monitoring beyond industrial manufacturing. In particular, we show that golden batch monitoring can be used for anomaly detection, attention focusing, and personalized training/skill assessment in a host of novel domains.

KEYWORDS

Time Series, Dynamic Time Warping, Anomaly Detection

ACM Reference format:

Chin-Chia Michael Yeh, Yan Zhu, Hoang Anh Dau, Amirali Darvishzadeh, Mikhail Noskov, and Eamonn Keogh. 2019. Online Amnestic Dynamic Time Warping to Allow Real-Time Golden Batch Monitoring. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'19). ACM, New York, NY, USA, 9 pages.* https://doi.org/10.1145/1234567890

1 Introduction

Batch production is a process used in many industries, in which goods are produced in groups (batches). Each batch goes through one stage of the production process before moving on to the next stage. Some examples of batch production (informally, a *recipe*) are the manufacture of food and beverages, pharmaceutics, chemicals, inks, paints, petrochemicals, and adhesives. Depending on the items being produced, a single batch cycle may take from seconds to several days. Most production efforts are heavily monitored by sensors, for example some chemical productions have over a thousand time series continuously recorded [30].

To help in a post-mortem analysis of an inferior batch, engineers use the concept of the *Golden Batch*. The golden batch is the ideal batch against which all other batches are compared. It may be produced from first principle physics, or simply by having the best engineers scrupulously cleaning the apparatus and producing a batch under ideal conditions. We argue that confining the use of golden batch comparisons to *offline* considerations represents a lost opportunity. Comparing to the golden batch in *real-time* offers the possibility to:

- **Intervene**: Many batch productions last hours to days. For processes running at these time scales it may be possible to save a production run that is drifting towards an unacceptable product, by physically intervening in some way. One example of this intervention is seen when manually opening an axillary valve to cool an overheating ingredient.
- **Gracefully Abandon**: In the cases in which a batch is irretrievably lost, it would be useful to know this as soon as possible. Not only does this save time, it may prevent damage to equipment and raw material. For example, if a metal cutting robot chips a blade and is allowed to continue cutting, it may eventually damage the actuators that move it. Replacing the blades is a cheap and quick fix, but replacing the actuators will result in hours of costly downtime.

However, comparing the telemetry of two batch processes is difficult, since there are two sources of variability:

- Processes typically exhibit some *allowable* batch-to-batch variability. These can be caused by random variations in the ingredients, or in the weather (temperature, air-pressure, humidity). These variations may also be *systematic*. For example, a valve may slowly clog over time, requiring increasingly a longer time to fill a vat with some ingredients.
- The process may vary in an *unacceptable* way; for example, moving a dairy ingredient too fast through a thermiser process to allow it to become properly pasteurized [28], or a welding robot continuing to weld after tip-contact has reduced the ability of the welder to produce clean welds [12].

While one type of variation is inconsequential, and the other is catastrophic, a classic "lock-step," one-to-one comparison such as Euclidean distance is unlikely to be able to tell the difference. Consider the small synthetic example shown in Figure 1.

The result is unintuitive. While the bad batch has a clear semantic difference to the reference golden batch, it is much closer to it than the good batch is. It is only with very careful inspection that we can see why. The good batch has some "warping" in time. Most real-world realizations of a batch process will have such slight variations in time. For example, a pressurization step may take a little longer on a cold day, but the next day the humification step may go faster because it happened to be raining. Figure 2.*top* shows such example in a *delayed coker*, a machine used in refining petrochemicals.

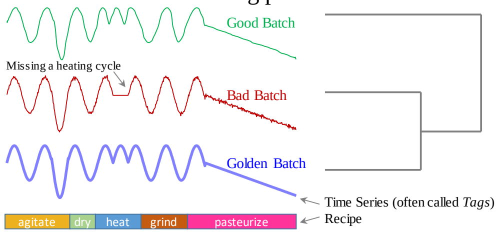

Figure 1: A toy example of a Golden Batch, compared to a good batch and a bad batch under the Euclidean distance.

One way to become less sensitive to slight distortions in time is to compare the batches using *global* statistical features, say the maximum and minimum values. However, if we do this, we would have no reason to reject the bad batch shown in Figure 1.

Given the above, we need a comparison mechanism that is invariant to small local shifts in time, but insists that the *right* processes happen, and in the *right* order. The reader may appreciate that Dynamic Time Warping (DTW) is potentially such a distance measure [18]. To see this, and to demonstrate that such timing differences often occur in real-word data, consider Figure 2.*bottom*.

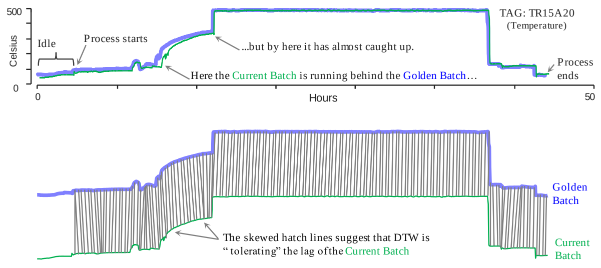

Figure 2: *top*) Two 44.4 hour runs of a delayed coker. While the runs are very similar, at about hour nine the current batch begins to drift behind the golden batch, eventually lagging by 27 minutes by around hour eleven, before moving back into phase. *bottom*) DTW is invariant to such local timing differences.

However, DTW is typically only defined for batch data and reports only a *global* difference score [23][24]. In this work, we will show how we can generalize DTW to work *online* and report a *localized* measure of compliance to an ideal template. Moreover, we will show that our *local DTW-based compliance measure* (hereafter DCM) is *amnestic*. That is to say, if a sensor reports that some measure is drifting out of compliance, and through a corrective action, it is brought back into tolerance, the compliance measure will reflect that. This contrasts with a classic incremental DTW score, which are cumulative and monotonically increasing [1].

Before delving into our proposed algorithms, we will take the time in the following two sections to discuss the actionability of golden batch monitoring, and to expand the purview of our ideas by explaining how they can used in non-industrial settings.

1.1 The Actionability of Golden Batch Monitoring

Given that we are proposing to move from an offline to an online analytics model, we need to consider the actionability of real-time monitoring. We see our method as allowing the following:

- **Attention Focusing Algorithms**: For video surveillance, it is often the case that there are thousands of cameras, but only a single human to monitor them all. To bridge this gap, *attention focusing algorithms* are algorithms designed to produce an everchanging prioritized "top ten" type list of views to monitor. We envision doing the same with golden batch monitoring [5]. As shown in Figure 3, there are many commercial tools for monitoring time series. While these tools have various built-in automatic analytics, they also rely on human inspection of the evolving process. However, there are many industrial processes that have 1,000+ time series. We envision using the current value of *DCM* as an index to prioritize human inspection.

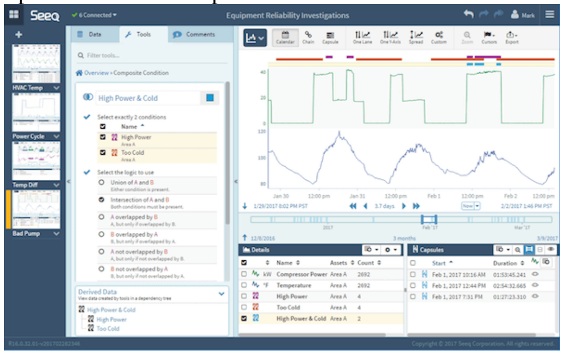

Figure 3: A commercial tool for monitoring industrial process data. Here the user has chosen to monitor just two time series, out of perhaps thousands.

- **Anomaly Detection**: In some cases, it may be possible to learn the maximum acceptable levels for the DCM to deviate, and then sound an alarm if that threshold is reached. We test this idea in great detail in Section 5.2.
- **Personalized Training and Skill Assessment:** The idea of golden batch comparison has been adapted or rediscovered by several researchers for the task of personalized training in fields as diverse as music, sports, and surgery [8]. However, most of this work is only used for after-the-fact analysis. The ideas introduced in this paper allow us to do this in real-time.

The rest of this paper is organized as follows. In Section 2, we consider related work. Section 3 introduces all the necessary definitions and notation. In Section 4, we explain our methodology, including some variants that may be useful, depending on the domain of interest. We perform a comprehensive evaluation in Section 5 before offering conclusions and directions for future research in Section 6.

2 Related Work

Our review of related work is brief. To the best of our knowledge, there is no commercial tool that offers the capability that we are proposing. This, in spite of the fact that there are at least twenty major companies offering software products in this space, including Seeq (see Figure 3, Rockwell, Emerson Automation Experts, AspenTech, Honeywell, Trendminer, and the like. Many of these products *do* have a golden batch product, however these products are mostly limited to editing/creating the golden batch. Once created, the user is invited to monitor by simply visualizing the incoming batch. For example, a training video for Trendminer suggests the user ask themselves, "*as this batch is going on, how does it visually compare to my* (golden batch)" [27] (our emphasis). AspenTech's Aspen ProMV multivariate analysis product does have extensive batch monitoring capabilities, but does not support pattern comparisons based on a golden batch [1].

The problem of *early classification in time series* is superficially similar to the task at hand, see [15] and the references therein. However, that problem is a supervised problem, and requires copious training data in at least two classes. In contrast, we may have only a single positive exemplar available.

Many principles of golden batch comparison have been rediscovered by [8] for the task of personalized training for surgical skills, moreover, they also use DTW for this purpose. The feedback provided is claimed to be real-time, however this claim needs to be qualified. It is real-time in the sense that during a single operation, shortly *after* the surgeon completes an atomic action such as suturing or knot-tying, feedback can be provided. However, they cannot provide feedback *during* an atomic action. In contrast, our proposed approach can do exactly that.

3 Definitions and Notation

We begin by defining the data type of interest, *time series*:

Definition 1: A *time series* $T \in \mathbb{R}^n$ is a sequence of real-valued numbers $t_i \in \mathbb{R} : T = [t_0, t_1, \ldots, t_{n-1}]$ where n is the length of T.

A specific type of time series is defined for the golden batch monitoring problem which is the problem of interest.

Definition 2: A *golden batch* $G \in \mathbb{R}^n$ is a time series that stores the "ideal" outputs from a process sampled by a sensor [1].

Note that we use the term "process" here in the most general sense. While we are mostly interested in industrial processes, our ideas may have implications for medical processes [8][26][34], sports performances, artistic performances [8] and so on.

With the term "golden batch" defined, we are ready to introduce the research problem addressed in this work.

Definition 3: The *golden batch monitoring problem* requires a real-time system that provides quantitative measurement on how a monitored time series differs with a set of golden batches at each point in time. Formally, given a time series that is being monitored $T \in \mathbb{R}^n$ and k golden batches $\boldsymbol{G} = [G_i | 0 \le i < k, G_i \in \mathbb{R}^m]$, a golden batch monitoring system should return a vector $E \in \mathbb{R}^n$ where $E[i]$ stores the difference/error between $T[0{:}i]$ and $\boldsymbol{G}$.

Note that we speak of a *set* of golden batches. This set is often of size one. However, sometimes the process may have polymorphic behaviors that are best modeled by multiple golden batches. For example, for any large-scale industrial process the machinery is typically exposed to the elements, thus the plant engineers may have a *hot-day* golden batch and a *cold-day* golden batch. In one extreme, the plant engineers may add *all* successful batches to the golden batch set. We will further address both the single and the polymorphic cases in Section 5.2.

In our definition, the golden batches are all of length *m*, but each such discovered golden batch may have slightly different lengths. We can fix this by simply interpolating all batches to the same length. No information is lost by this; each batch is really encoding information about local ordering and local timing of events, the DTW itself will give invariance to the global timing differences, which are typically quite small (on the order of a few percent).

As noted in Definition 3, we need to monitor the error between the most recently arrived data points in current batch with the golden batch under optimal alignment. Such information can be computed by DTW, and we store the result errors in an *accumulated error matrix*. Note, such matrix is frequently referred to as *warping matrix* in literature, but we name it the *accumulated error matrix* for clarity. Figure 4 shows the optimal alignment between a golden batch and the time series in current batch.

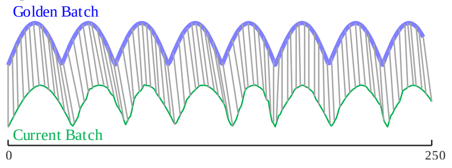

Figure 4: The optimal alignment between the most up-to-date time series in current batch with the golden batch.

Definition 4: An *accumulated error matrix* $\boldsymbol{M} \in \mathbb{R}^{n \times m}$ is a matrix stores the amount of deviation (or error) accumulated over time when comparing all prefix of two given time series under optimal alignment. Formally, given two time series $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$, $\boldsymbol{M}[i,j] = DTW(X[0{:}i], Y[0{:}j])$.

This accumulated error matrix $\boldsymbol{M}$ stores the classic incremental DTW scores between all pairs of prefix from two given time series.

Before moving on, we note that while we use the term "error" here for consistency with the literature [12][29], it may sound unnecessarily pejorative in this context. Most of the "error" is just natural and allowable variability.

As noted in [30], "it may be important to control some parameters tightly, while other measurements may vary significantly without affecting the produce quality." We need a way to represent this variable adherence constraint to golden batch. The idea of warping constraint window is a classic concept

[1]There is no standard terminology for this concept, it is also called Golden Fingerprint [27], Golden Profile, Ideal Batch etc.

in DTW that can be co-opted for this task [6]. As shown in Figure 5.left.top, the fixed warping constraint window of 4 is visualized with respect to an accumulated error matrix $\boldsymbol{M}$. We only need to compute the highlighted elements (gray) of $\boldsymbol{M}$.

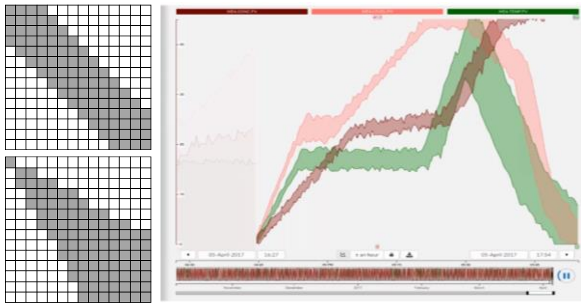

Figure 5: DTW can be constrained by only allowing it to visit the cells marked in gray. In *left.top* we show a classic *fixed* warping constraint window, and in *left.bottom* we show a *growing* warping constraint window, that is designed to model accumulating "drift" over time. The need for this growing constraint is reflected in this commercial dataset (*right*).

Moreover, consider Figure 5.*right* which shows a screen dump of Trendminer's golden batch tool monitoring three variables. The "envelopes" show the variability from six successful runs. Note that the timing uncertainty/variability increases over time. This is to be expected, as the amount of misalignment between two time series can accumulate over time as they drift apart. As shown in 0.*left.bottom*, by using a suitable warping constraint window, we can represent such a linearly growing temporal drift. Once again, the warping constraint window is visualized with respect to an accumulated error matrix $\boldsymbol{M}$, and we only need to compute the highlighted elements (gray) of $\boldsymbol{M}$.

To summarize or "flatten" the matrix $\boldsymbol{M}$ into a vector or time series, we define the accumulated error profile:

Definition 5: An *accumulated error profile* $E \in \mathbb{R}^n$ is a time series which summarizes an accumulated error matrix $\boldsymbol{M}$ by extracting the classic incremental DTW score between all prefix of one of the given time series pair with the best aligned prefix of the other time series within warping constraint window. Formally, given an accumulated error matrix $\boldsymbol{M} \in \mathbb{R}^{n\times m}$ (of time series $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$) and a warping constraint window w, $E[i] = \min(\boldsymbol{M}[i, i-w: i+w]) = \min_{i-w\leq j\leq i+w} DTW(X[0:i], Y[0:j])$.

Without loss of generality, for the remainder of this work we assume a fixed warping constraint window for clarity of presentation. In the case of dynamic warping window, the warping constraint window used for the i^{th} element in E should be computed by a warping constraint window function $w(i)$. Figure 6 shows an example of accumulated error profile between a golden batch and the time series in current batch. The error in accumulated error profile increases dramatically when the current batch deviates from the golden batch.

Because the values in E are classic incremental DTW scores, the values in the time series are cumulative and thus monotonically increasing over time. However, we want a score that is "forgiving" or amnesic. Because batches can take many hours, if we signal a problem and corrective action is taken, we want the score to be allowed to return within normal limits. Thus, we define the localized error profile:

Definition 6: A *localized error profile* $E_l \in \mathbb{R}^n$ is a time series that stores the *local DTW-based Compliance Measure or* DCM, which is non-cumulative and non-monotonically increasing.

Figure 6.*bottom* shows an example of localized error profile between a golden batch and the time series in current batch. Note the error caused by the missing rectified half sine wave only induces error *locally* in localized error profile.

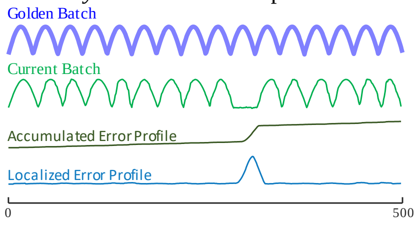

Figure 6: *top to bottom*) A golden batch. A (bad) current batch which misses a rectified half sine wave. The accumulated error profile. The localized error profile returns to a low value after peaking during the anomalous period.

The golden batch monitoring problem can be considered as a special case of the anomaly detection problem [17] with the following properties:

1. The training data is available but only contains example(s) of the "normal" class.
2. The training data is extremely scarce.
3. The test data arrives in a streaming fashion.

To interpret golden batch monitoring as an anomaly detector, we can simply set some maximum permissible value for the error profile. This value could be set from first principles, or as we show in Section 5.2, learned directly from training data. Our method can be considered as a spiritual successor to one of the most competitive anomaly detection method [4][13][17] tailoring toward anomaly detection problem with aforementioned properties.

However, it is important to note that golden batch monitoring is more general than classic anomaly detection as it is typically understood, as it allows for the possibility to take corrective action before the situation becomes dire.

4 Methodology

We are finally in a position to formalize our algorithms. We begin with the *single* golden batch case, before generalizing to the *polymorphic* golden batch case.

4.1 Online DTW Algorithm: Single Batch Case

Given a golden batch G and a time series T that we are monitoring, the localized error profile E_l is computed using the online DTW algorithm (see Algorithm 1) modified from Ψ-cDTW [25]. Note the pseudocode is described as a batch algorithm (opposed to online algorithm) for clarity of presentation; we will discuss the modifications that allow Algorithm 1 to be computed in an online fashion while introducing it.

Algorithm 1: The *OnlineDTW* algorithm

```
Procedure OnlineDTW($G, T, w$)
Input: a golden batch $G$, a time series $T$, and warping constraint window $w$
Output: localized error profile $E_l$
1   $n_G \leftarrow length(G), n_T \leftarrow length(T)$
2   $G \leftarrow normalization(G)$
3   $\boldsymbol{M} \leftarrow infinity\_matrix\_of\_size(n_T, n_G)$
4   $\boldsymbol{M}[0{:}w,:] \leftarrow 0, \boldsymbol{M}[:,0{:}w] \leftarrow 0$
5   for $i \leftarrow 0$ to $n_T$
6       $T[i] \leftarrow normalization(T[i])$
7       for $j \leftarrow max(0, i-w)$ to $min(n_G, i+w)$
8           $\boldsymbol{M}[i,j] \leftarrow abs(T[i] - G[j]) +$
9                   $min(\boldsymbol{M}[i-1,j], \boldsymbol{M}[i,j-1], \boldsymbol{M}[i-1,j-1])$
10      end for
11      $E[i] \leftarrow min(M[i, max(1, i-w){:}min(nG, i+w)])$
12      $E_l[i] \leftarrow E[i] - E[i-1]$
13  end for
14  return $E_l$
```

In line 1, the lengths of both input time series T and G is stored. Since the length of T is unavailable in the online scenario, we can simply initialize it as $n_G + w$. The maximum meaningful length of T is limited by the warping constraint w as any point in T beyond $n_G + w$ cannot be matched with any point in G. Next, in line 2, G is normalized using a normalization function designed specifically for the data domain in question (We will expand on this in Section 4.4). In line 3, the accumulated error matrix $\boldsymbol{M}$ is initialized as a matrix of infinity values. Note in the pseudocode the matrix is initialized as a full matrix with space complexity of $O(n_G\ n_T)$. Because of the warping constraint w, majority of the off-diagonal entries are not used. Only $O(min(n_G, n_T)w)$ space is needed to store the relevant information. In line 4, $\boldsymbol{M}$ is prepared based on [25] to relax the prefix endpoint constraint.

From line 5 to line 13, the loop is advanced by one iteration each time a new data point in T is received. In the case where the length of T is unknown in advance, the for-loop can be replaced with a while-loop which breaks when receiving the end-of-time-series signal. In line 6, the received value is normalized using appropriate normalization function. From line 8 to line 11, we update $\boldsymbol{M}$ using the standard DTW recurrence relation [11][24].

In line 11 and line 12, the accumulated error profile E and the localized error profile E_l are updated. Under online scenario, we can yield/emit the newest element of E_l here. We do not use the relaxed suffix endpoint constraint [25] as it has the freedom to choose either the i^{th} point in G and/or the i^{th} point in T to compute the latest value in E. Because the purpose of Algorithm 1 is monitoring T, the most recent (i^{th}) point in T should be considered to allow the earliest possible anomaly detection. In other words, we are using the suffix endpoint constraint proposed in [26] instead of the one proposed in [25]. See Figure 7 for an illustration of the adopted suffix endpoint constraint. The localized error profile is computed simply by taking the difference between current value in E (i.e., $E[i]$) with the previous value in E (i.e., $E[i-1]$). One may smooth E_l with moving average for better visual representation. Finally, the result localized error profile E_l is returned in line 14.

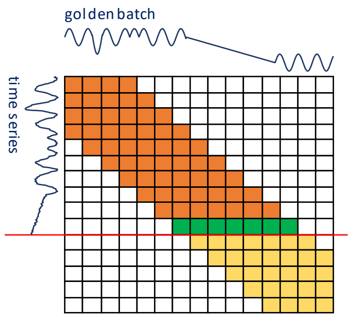

Figure 7: Given the golden batch and the first *i* points of a time series, the i^{th} term in the accumulated error profile *E* is computed by locating the minimal value from the i^{th} row of accumulated error matrix $\boldsymbol{M}$. The i^{th} row of $\boldsymbol{M}$ is labeled green in the figure.

4.2 Online DTW Algorithm: Polymorphic Golden Batch Case

Given a golden batch set $\boldsymbol{G}$ and a time series T that we are monitoring, the localized error profile $E_{l,out}$ is computed using Algorithm 2, which calls Algorithm 1 as a subroutine.

Algorithm 2: The *OnlineDTW_P* algorithm

```
Procedure OnlineDTW_P($\boldsymbol{G}, T, w$)
Input: a set of golden batch $\boldsymbol{G}$, a time series $T$, and warping constraint window $w$
Output: localized error profile $E_{l,out}$
1   $n_T \leftarrow length(T), k \leftarrow size(\boldsymbol{G})$
2   for $i \leftarrow 0$ to $n_T$
3       for $j \leftarrow 0$ to $k$
4           $\boldsymbol{E}[j,i], \boldsymbol{E}_l[j,i] \leftarrow OnlineDTW(\boldsymbol{G}[\boldsymbol{j}], T[i], w)$
5       end for
6       $idx \leftarrow argmin(\boldsymbol{E}[:,i])$
7       $E_{l,out}[0{:}i] \leftarrow \boldsymbol{E}_l[idx, 0{:}i]$
8   end for
9   return $E_{l,out}$
```

In line 1, the length of input time series T and the number of golden batch time series in $\boldsymbol{G}$ are stored in n_T and k respectively. The variable n_T is only used for cleaner presentation and is not required in real implementation. From line 2 to line 8, we process each newly received data point of T in each iteration. From line 3 to line 5, we update both $\boldsymbol{E}$ and $\boldsymbol{E}_l$ using the latest point of T. Note the call to the $OnlineDTW$ only advances the outer loop (i.e., line 5 to line 13 in Algorithm 1) by one. In line 6, we identify the best (i.e., most similar in terms of DTW score) golden batch, and in line 7, we point the current localized error profile $E_{l,out}$ to the $\boldsymbol{E}_l$ computed using the best golden batch. Lastly, we return the localized error profile $E_{l,out}$ in line 9. The overall time complexity of Algorithm 2. is $O(\min(n_G, n_T)\, wk)$.

4.3 Generalization of Online DTW Algorithm

The algorithm we outlined in Algorithm 1 computes the accumulated error E for a univariate time series. As with the original DTW algorithm [11][24], our algorithm can be trivially modified for other types of sequential data such as multivariate time series or symbolic sequences by 1) changing the normalization function at line 2 and 6 (e.g., see Section 4.4,) and 2) changing the distance function at line 8 (i.e., replace abs($T[i]-G[j]$) with appropriate distance function (e.g., norm of the elementwise difference between vector $T[i]$ and $G[j]$ or sub-dimensional distance [31]).

In addition to the trivial modification outlined above, one nontrivial modification which turns the point-versus-point warping distance computation into subsequence-versus-subsequence warping distance computation is introduced here. The subsequence-versus-subsequence version is useful when the user is interested in aligning local shapes between golden batch G and time series T instead of the values of each individual points (i.e., shape-based alignment is amplitude/offset invariant). The naïve implementation of the subsequence-by-subsequence idea is simple. Given a subsequence length m, we first apply a sliding window of size m with hop 1 to extract all subsequence of G and T then store the extracted subsequences in respective list G_m and T_m (where $G_m[0]$ contains the first m sized subsequence of G). Then, we input G_m and T_m to a modified version of Algorithm 1 where 1) line 2 and 6 are removed, and 2) the distance function at line 8 is changed to the function that computes z-normalized Euclidean distance.

It is easy to understand the correctness of the naïve implementation, but the time complexity (i.e., $O(min(n_G,n_T)wm)$) would be suboptimal because we ignore the fact that there are $m-1$ points overlapped between consecutive subsequences. We can reduce the time complexity to $O(min(n_G,n_T)w)$ by using the same computational scheme which accelerates STOMP [32][33]. Due to space limitations, we will not describe the computational scheme in this paper. Those interested can refer to [32][33] for details about the computational scheme, and the optimized source code for online DTW using this scheme can be download from [21].

Another generalization that can be made to Algorithm 1 is replacing the fixed-width warping constraint window with a dynamic warping constraint window (see Figure 5 and its accompany paragraphs). The modification is identical to the standard DTW algorithm [23].

Because the evaluation of the last two aforementioned modifications is beyond the scope of this paper, we leave such considerations for future work.

4.4 Normalization

Normalization is crucial in the case of multidimensional time series when each dimension is measured by different types of instruments (e.g., thermometer, pressure gauge, or flow-rates gauge) [32][33]. Because the measurements from different instruments carry different units (e.g., kelvin, pascal, or cubic meters per second), numbers from different dimensions are not commeasure. If we sum the error from different dimensions without normalization, the result localized error profile would be dominated by the dimension with largest range. For example, in the Wafer database [16], the range of the six dimensions are `{9, 234, 10, 4, 1, 61}`; the first, third, fourth, and fifth dimension have little to no effect to the localized error profile compared to the second dimension. Such problem is usually addressed by applying z-normalization [18] or 0-1 normalization to each dimension of the multidimensional time series.

5 Experimental Evaluation

To ensure that our experiments are reproducible, we have built a website which contains all data/code/raw spreadsheets for the results [21]. In addition, some experiments are augmented by videos, showing the results being computed in real time.

We had access to datasets from a major oil and gas industrial analytics company to bootstrap and motivate this research effort. However, this industrial sector is notoriously secretive and unwilling to share data. This conflicts with our academic policy of complete transparency and reproducibly. Thus, we only evaluate on proxy datasets, and existing public datasets that we can archive and share in perpetuity [21].

5.1 A Case Study in Oil and Gas Processing

We begin with a simple demonstration of our ideas, before considering a more qualitative evaluation in subsequent sections.

As the reader will appreciate, the oil and gas industry is reluctant to share real data, particularly of faults. In this section, we revisit the real dataset introduced in Figure 2 and *induce* a synthetic fault. While the fault is synthetic, it is based on a real problem communicated to us by a petrochemical engineer. In Figure 8 we show the *DCM* computed for the example introduced in Figure 2.

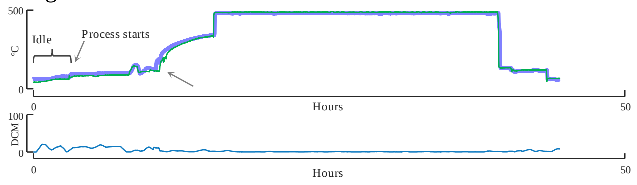

Figure 8: The DCM computed between the golden batch (blue/bold) and the current batch (green/fine) shown in Figure 2.

We used warping constraint window of 30 minutes, which was suggested by the petrochemical engineer. Note that the *DCM* never rises above 1.5, in spite of the fact that the beginning of the heat-up phase is a little slower in the current batch.

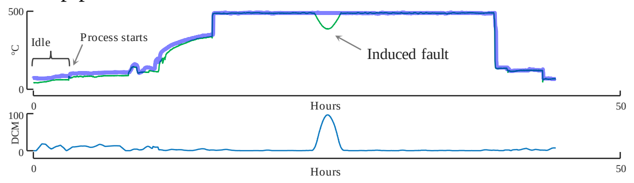

Figure 9: The DCM computed using the same golden batch (blue/bold) shown in Figure 8 and the current batch (green/fine) under the effect of recreated freak rainstorm.

Delayed cokers are typically so large that they are not housed in a building, but left outside exposed to the elements. A petrochemical engineer recalled a fault he had seen a handful of times in his career. In most of the Middle East’s oil processing regions, the summer days are uniformly hot with temperature between 45° C and 50° C and near zero chances of precipitation, ideal for oil processing. However, every few years, some regions can see a freak rainstorm lasting for just a few minutes. These brief rainstorms can quickly cool the unprotected boiler, as we have recreated in Figure 9.

The sudden cooling can cause foaming and heater fouling, which may require "pigging" (mechanical coke removal), which is extremely expensive. While it is not possible to heat up the boiler fast enough to mitigate the rain's cooling effect, it may be possible to reduce the infeed rate to reduce the damage. Here, the short lag between the cooling event and the dramatic rise of the *DCM* would allow such corrective interventions. As shown in Figure 9.*bottom*, the *DCM* score significantly increases at the relevant location. Moreover, as the temperature is brought back into compliance, the *DCM* score decreases to normal levels.

5.2 Large Scale Experiments

To allow large-scale experiments with stochastic element, we repurpose the eight-class Mallat data set [14]. As shown in Figure 10, this data set is at least, by visual inspection, a good proxy for a four-stage process in petrochemical processing [22][27][30].

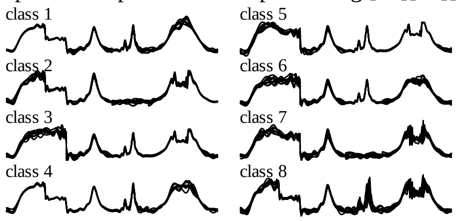

Figure 10: Samples of data from the Mallat dataset, showing multiple instances in each class to hint that the natural variability. While a synthetic dataset, they are highly reminiscent of real industrial traces.

We selected the k^{th} class from the Mallat data set and denoted all the time series from it as "normal" and the other time series as "abnormal" ($k \in \mathbb{Z}: 1 \leq k \leq 8$]). We created the training data set by randomly including eight time series from the normal class, and we added all the excluded time series (both normal and abnormal) to the test data set.

We repeated this procedure 32 times for each class. In other words, we use the Mallat data set 256 times to generate training and test data splits for *each* trial of experiment.

Given that we have eight exemplars to act as the golden batch, but plan to only use one (for evaluating the single batch system), we need a policy to obtain our *single* true golden batch. We consider the follows two options:

- We construct the golden batch set by randomly *selecting* a time series from the training data set (i.e., the naïve method).
- We construct the golden batch set by *averaging* time series from the training data set. It has been demonstrated that DTW averaging can be very useful for producing idealized exemplar for classification and clustering [18][19], and as such, it is natural to ask if it has utility here.

For each time series in the test data set, we compute the localized error profile E_l using Algorithm 1, and then we use the *Maximum DCM* (MCM) (i.e., the maximum value in E_l) as the anomaly score.

As the objective is to detect anomalies, we need a threshold for the anomaly score. We set the threshold as $MCM_{\text{T}} = mean(MCM) + 3\,std(MCM)$ where the $mean(MCM)$ and the $std(MCM)$ are estimated from the MCMs computed using the training data set. This idea is adapted from the classic Three-sigma Limit in statistical process control [22]. Note, in the case where multiple time series are presented in the golden batch set we estimated the threshold for each golden batch time series separately. The learned threshold MCM_{T} can also be used to normalize E_l (i.e., $\bar{E}_l = E_l / MCM_{\text{T}}$), so the $\bar{E}_l$ can be loosely interpreted as the probability of a time series being abnormal.

We summarize the experimental results in Table 1 using two different performance measurements: F-score and AUC-ROC [20]. The presented values are computed by averaging the experiment results over 256 trials of experiment. The DTW averaging technique [18][19] improves the system significantly over the naïve method (tested with paired sample t-test with 5% significance) under both performance measurements. In contrast, the F-score of the naïve method is even worse than the random guess baseline.

Table 1: Naïve Method Versus DTW Averaging

	F-score	AUC-ROC
Random Guess	0.935	0.500
Naïve	0.878	0.968
DTW Averaging	0.964	0.990

As neither the F-score nor the AUC-ROC provides the whole picture for the capability of our method, we also report the confusion matrix. The confusion matrix is shown in Table 2. Because we are averaging over the 256 runs, the values are real numbers; however we round to integers for clarity.

Table 2: Averaged Confusion Matrix

n = 2,392	Predicted Good	Predicted Bad	Total
Actual Good	TN = 259	FP = 33	292
Actual Bad	FN = 106	TP = 1,994	2,100
Total	365	2,027	

Because the Three-sigma Limit rule is just a heuristic "rule-of-thumb", the optimal threshold may be a value other than three standard deviation above the mean [22]. In a post-hoc analysis, we varied the threshold from mean minus three standard deviation to mean plus six standard deviation. Figure 11 shows the averaged F-score as we vary the threshold from conservative (i.e., $mean(MCM) - 3\,std(MCM)$/lower FP rate/higher FN rate) to liberal (i.e., $mean(MCM) + 6\,std(MCM)$/higher FP rate/lower FN rate). Note positive is associated with the anomaly class while negative is associated with normal class. Although this shows that the threshold defined using the Three-sigma Limit rule is not optimal, the performance gained by using the optimal threshold is very small (i.e., from 0.965 to 0.971), and any values between -1 and 5 will produce a result better than random guessing (i.e. the *default rate*).

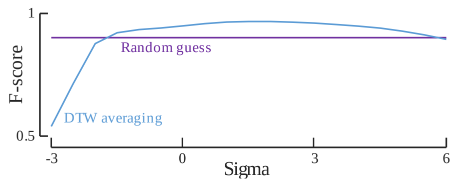

Figure 11: The optimal threshold is $mean(MCM) + 1.5\ std(MCM)$ based on this post-hoc analysis.

It is common to have a polymorphic "normal" class in industrial processes. For example, the input material (infeed) for a chemical processing factory may have a different composition for different seasons (i.e., Summer and Winter [29]) or different origins of raw material, and this difference in the infeed causes the process to generate distinct time series [29]. In this case, we require multiple time series to represent an acceptable run in the golden batch set.

To test out our method under polymorphic settings, we repurpose the Mallat data set in a similar fashion as our previous experiment, but instead of just selecting samples from a *single* class as the normal samples, we select samples from *multiple* classes to form the training set. At each trial, we first generate a random integer $n_c \in \mathbb{Z}: 1 \leq n_c \leq 4$ which gives us the number of Mallat classes we draw our samples from. Next, we randomly choose n_c classes from the Mallat class label set $\{k \in \mathbb{Z}: 1 \leq k \leq 8\}$ and add the samples associated with the selected class into the normal class set. Finally, we randomly select 16 samples from normal class set as the training set and leave the rest as the test set. As with the previous experiment, we repeat this process 256 times.

As shown in the previous experiment, DTW averaging is significantly better than the naïve method. To apply the same idea to this problem, we adopt the k-means with DTW averaging to produce the golden batch set from the training data. The k in k-means is set to four as it is the minimum number of samples in golden batch to represent the normal class when n_c is four (i.e., k is fixed to four even if n_c is less than 4). For each time series in the test data set, we compute the localized error profile $E_{l,out}$ using Algorithm 2, then we use the *Maximum DCM* (MCM) (i.e., the maximum value in $E_{l,out}$) as the anomaly score.

Table 3: Naïve Method Versus DTW Averaging

	F-score	AUC-ROC
Random Guess	0.575	0.500
Naïve (1 batch)	0.343	0.758
DTW Averaging (1 batch)	0.564	0.806
Naïve (4 batches)	0.760	0.858
DTW Averaging (4 batches)	0.875	0.896

As in the previous experiment, we use the Three-sigma Limit heuristic to determine the threshold; we use F-score and AUC-ROC [20] to measure the performance. Table 3 shows the performance of both naïve method and DTW averaging with 1/4 batches.

The DTW averaging technique with multiple batches improves the system significantly over the others (tested with paired sample t-test with 5% significance) under both performance measurement. Both naïve and DTW averaging with one batch's resulting F-scores are worse than random guessing which further reinforces the need for multiple batches in polymorphic scenario.

Again, as neither the F-score or the AUC-ROC provides the complete picture for the capability of our method, we also report the confusion matrix. Averaged over 256 runs, the result confusion matrix is reported in Table 4. Note that as in Table 2, we again rounded to integers for clarity.

Table 4: Averaged Confusion Matrix

n = 2,384	Predicted Good	Predicted Bad	Total
Actual Good	TN = 507	FP = 212	719
Actual Bad	FN = 139	TP = 1,526	1,665
Total	646	1,738	

We performed the same post-hoc analysis as before to reexamine the threshold established by Three-sigma Limit rule. Figure 12 shows the averaged F-score as we vary the threshold. Although the threshold defined using the Three-sigma Limit rule is not optimal, the performance gained by using the optimal threshold is slim (i.e., from 0.875 to 0.877).

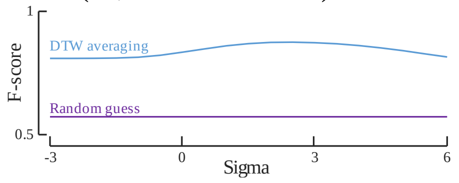

Figure 12: The optimal threshold is $mean(MCM) + 2.5\ std(MCM)$ based on this post-hoc analysis.

In addition to the polymorphic cases, it is also important to test the method on multidimensional or multivariant time series as multidimensional time series is ubiquitous in domains such as industrial and human activity monitoring. For this purpose, we have tested our method on the Wafer database [16] which is a real dataset consisting of readings from six different sensors (e.g., pressure, emission, and power) during a wafer manufacturing process. The yield of the process is either good (or normal) or bad (or abnormal).

The Wafer database consists of 1,067 normal runs and 127 abnormal runs. To format the data for our experiments, we have randomly selected n_t examples from the normal runs as the training data and left the rest as the test data. We repeat the experiment 32 times for each of the four different n_t settings (i.e., 8, 16, 32, and 64). Since there is only one type of normal pattern, we only use one batch in this set of experiment. Figure 13 shows the AUC-ROC as we vary the number of training data. DTW averaging constantly outperforms the naïve method and the performance does not change substantially as we vary the number of training data.

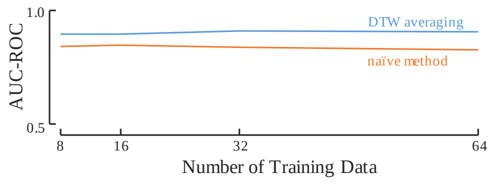

Figure 13: The DTW averaging technique improves the performance in all cases disregard the number of training data.

As with the previous experiments, we also wish to evaluate the three-sigma limit rule with a post-hoc analysis under different numbers of training data. Figure 14 shows the F-score as we vary both the sigma and the number of training data. The three-sigma limit rule produces near optimal results when the number of training data is small. However, when we increase the number of training data, the chance of hitting the optimal (or near optimal) threshold decreases. The plateau around the optimal threshold decreases as the number of training data increases. This observation hints the possibility of setting the threshold as a function of training data set size. This is somewhat similar to the observation that the DTW's warping window width should be decreased as increase the training set size [6]. Nevertheless, we leave such investigation for future work as it is beyond the scope of the current paper.

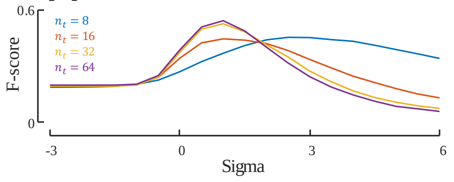

Figure 14: The F-score as we vary both the sigma and the number of training data.

5.3 Expanding the Purview of Golden Batch Monitoring

In our introduction, we noted that one use of a golden batch monitoring system is *attention focusing*. This idea is most commonly associated with video monitoring [5], but applies in any circumstances where multiple evolving data sources compete for human attention. In this section, we show that our algorithms allow us to expand the notion to of *attention focusing* to a novel domain.

The Taipei Mass Rapid Transit (MRT) system consists of 108 stations and carries an average 2.10 million passengers per day. As shown in Figure 15, the number of passengers using each station is monitored by faregates and other sensors at a fine temporal resolution[2]. Note that the three time series shown in Figure 15 look quite different, as their use is dictated by their relative proximity to centers of education, entertainment, shopping, residence, and the like. The metro-master is stationed at the headquarters in Taipei Main Station. One of her tasks is to monitor the demand at the stations and respond by redirecting trains. For example, if she notes an unexpectedly high demand, she reduces the train interval. Because of the lag time needed to prepare trains for service, timeliness is of the essence, and she must monitor 108 time series at once.

[2] The data can be download from http://data.taipei/, however we have cached the exact version we used at the supporting website [21].

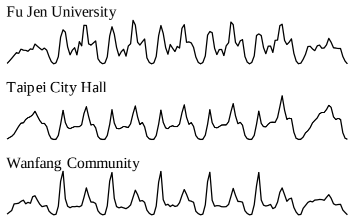

Figure 15: The passenger counts over the first week of November 2015 for three different stations.

We envision casting this task as a *Golden Batch Attention Focusing* problem. Concretely, we take the time series recorded in each station in the *previous* week and denote it as the golden batch for that station. We then proceed to monitor each station's *current* week time series with the golden batch and rank them based on the result DCM ranking from high DCM (more "different" to previous week) to low DCM (as expected based on previous week).

Figure 16 shows the time series from the top ten stations from December 27, 2015 to December 31, 2015 around midnight.

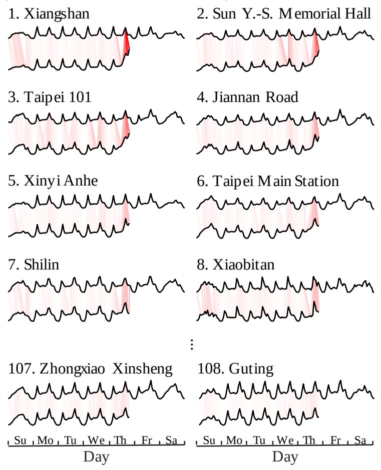

Figure 16: The New Year's Eve firework around midnight creates a surge in ridership, especially the stations near the popular firework watching spot (i.e., the first five stations) and major transportation hub (i.e., Taipei Main Station).

For each station, the top time series is the golden batch, and the bottom time series is the time series being monitored. The red lines linking both time series show the alignment between the two weeks, with the boldness of the red line in proportion to the DCM value (perfect alignments are effectively invisible).

The ridership surges in many stations due to the Taipei 101/Miramar Entertainment Park New Year's Eve fireworks event particularly for the stations near popular firework watching spots (i.e., the first five stations) and major transportation hub (i.e., Taipei Main Station). For contrast, note that the last two stations (i.e., Zhongxiao Xinsheng and Guting) consist of almost invisibly faint lines, suggesting that they are unaffected by the festivities.

To further consider the utility of our ideas in this domain, we simulated an unexpected event, and measured how much time it would take to force the effected stations to rise to the top of the list of 108. We imagine that the area around Presidential Office Building unexpectedly closes due to a spontaneous demonstration event that happens on the last Sunday of March 2016 from 2 pm to 8 pm (assuming people start to gather around 1 pm). This forces many people leaving the NTU Hospital (which is very close to the Presidential Office Building) to use the Shandao Temple Station instead of the NTU Hospital Station during the demonstration. In particular, we simulate an 80% reduction in the volume of traffic at NTU Hospital Station and corresponding increase in the volume of traffic at Shandao Temple Station. Note the percentage change is substantially less in Shandao Temple Station as it is a busier station compared to the NTU Hospital Station.

Here we use the previous Sunday as the golden batch. As Figure 17 shows, this simulated event does result in the two effected stations quickly become the top two at 1:51 pm, which gives the control center 9 minutes to react before the demonstration begin. For contrast, note the last two stations (i.e., Nanjing Fuxing and Taipei Main Station) consist of almost invisibly faint red lines suggesting that they are unaffected by this disturbance.

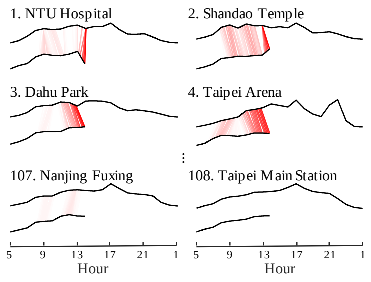

Figure 17: Oppose to Figure 16, we only monitor in a daily fashion instead of weekly fashion to better show the minutely update (upsampled from hourly update).

5.4 Human Performance Coaching

The idea of personalized training and skill assessment by comparing time series telemetry of an idealized performance to subsequent attempts has been attempted in a host of domains, including music [9], surgical skills [8][34] and sports. However, in virtually all these efforts, the coaching is done offline. In a handful of domains, it may be possible to give the user feedback in real time. For example, when practicing calligraphy, the feedback can come in the form of visual cues (if using pen-based computing), or tactical feedback (if using a haptic pen [3][7]). With this in mind, we consider an application of our system to calligraphy coaching and instrument playing skill assessment.

Calligraphy

One common practice for learning the art of writing Hanzi/Kanji (i.e., Chinese characters) as a beginner is to carefully imitate a given example over a 3x3 grid. Figure 18 shows an example of such practice for the character "tea". A beginner can first observe/trace the example shown in the left grid, then practice recreating the character in an empty grid like the one shown in the right. Note, for Hanzi/Kanji writing, the order of writing each stroke and the way to initialize/end each stroke[3] is standardized.

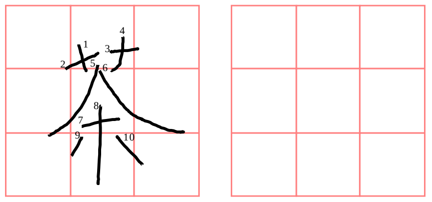

Figure 18: An example for the Hanzi/Kanji "tea" on a 3x3 grid. A beginner is encouraged to learn to write the character by either tracing the given example in the left grid or redraw the character in the right grid. The numbers shown in left grid are the order of writing each stroke.

We have formulated the Hanzi/Kanji writing practice process as the golden batch problem in which the example provided by an experienced writer is the golden batch and a naïve writer is asked to imitate the golden batch while our golden batch method provides real-time feedback. That is, the *DCM* between the naïve writer's current pen location and the golden batch can be used to drive a haptic feedback system to correct the naïve writer's writing in real-time [7]. In our experiment, we have asked an experienced writer to provide 15 examples of the character "tea," then we have asked a naïve writer to try his best recreating the writing of the experienced writer.

Both writers utilized Livescribe Echo Smartpens to transcribe the character "tea" on a dotted pattern paper. When a user writes, an infrared camera at the tip of the smartpen detects a subtle (near invisible to the human eye) dot-matrix pattern on the paper. The dot-matrix pattern provides information about where on the page the writing is occurring. This information is stored as a series of data points containing Cartesian XY coordinates and timestamp. The tip of the smartpen is equipped with a switch that is sensitive to pressure. The switch records the timestamps at which the smartpen is writing or has been lifted up from the paper. This data is utilized to split collected data points into separate pen strokes. Finally, these pen strokes are transformed into 2D time series.

We learn the golden batch from the five best attempts of the experienced writer (as she self-judged) with DTW averaging. Figure 19.*left* shows the learned golden batch. By computing the *DCM* between a naïve writer's writing with the golden batch, we can acquire the amount of correction required for the naïve writer to improve himself at each individual time instance. Figure 19.*right* shows the naïve writer's writing over the golden batch. The thin lines shown in the figure indicate the alignment between the golden batch and the naïve writer's writing where the darkness of the line is proportional with the amount of the correction (i.e., *DCM*).

[3] For example, a stroke is typically written in a top-down fashion for vertical stroke.

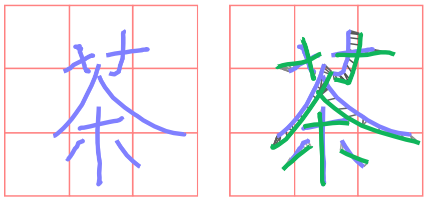

Figure 19: *left*) The golden batch learned from the experience writer's examples. *right*) A beginner's writing (green) over the golden batch. The thin lines indicate the alignment between the beginner's writing and the golden batch where the strength of the line indicates the strength of correction needed.

In Figure 20, we have replaced the complete writing with a half-completed example to showcase our method's real-time output in the middle of writing. Compared to the one shown in Figure 19.*right*, the character is written by a more experienced writer. Because the last stroke in Figure 20.*right* deviates from the golden batch, the DCMs of this stroke are stronger comparing to the prior written part of this character. Given only that our system is able to provide either a visual or haptic feedback, the writer can correct his/her writing in real-time.

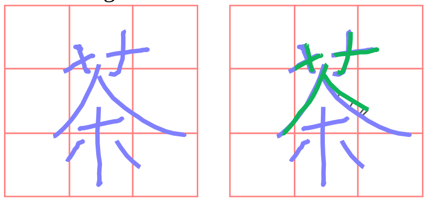

Figure 20: *left*) The golden batch learn from the experience writer's examples. *right*) A half-done writing (green) over the golden batch. The thin lines indicate the alignment between the beginner's writing and the golden batch where the strength of the line indicates the strength of correction.

Music

Real-time performance feedback is important for learning musical instruments [9]. For example, if a musician plays the wrong note in the first chorus during practice, it is very likely that she will make the same mistake in the second chorus of the same playthrough because she may misread the music script in the same fashion. However, if the golden batch system notifies the musician in the first chorus about the misplayed note, it is possible the same mistake can be avoided in the second chorus.

We use the piano arrangement of *Stairway to Heaven* by Led Zeppelin. The musical script was downloaded in MIDI format from [10]. We use the downloaded MIDI as the golden batch, then generate the batch being monitored with the following errors, which were suggested by a music teacher as being common mistakes by amateurs:

1. We vary the start time and end time of each piano keystroke with a lead/lag of $t \sim N(0, 0.025)$ seconds.
2. Each piano keystroke is given a 0.5% chance of being wrong.

A wrong key is generated by randomly ascending/descending the pitch $k \sim N(0, 5)$. Figure 21 shows a snippet of the modified midi visualized as piano roll. The wrongly played key (circled in red) can be higher or lower in pitch comparing to the golden batch.

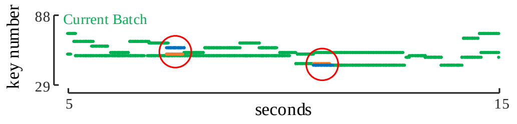

Figure 21: A ten second snippet piano roll of the modified MIDI starting from the 5th seconds of the song. The musician made two mistakes, which are circled in red. The correct key is shown in blue and the incorrect key in orange.

The outputted DCM of our golden batch algorithm of the same ten seconds snippet is shown in Figure 22.*bottom*. The DCM raised prominently when the wrong note is played. Although our method only shows the temporal location of the wrongly played note, it is trivial to identify the specific key which the musician wrongly played using an XOR operation given the temporal alignment (which is part of the gold batch's output). A demo video, which shows the DCM being computed just-in-time as the musician plays the instrument, can be found in [21].

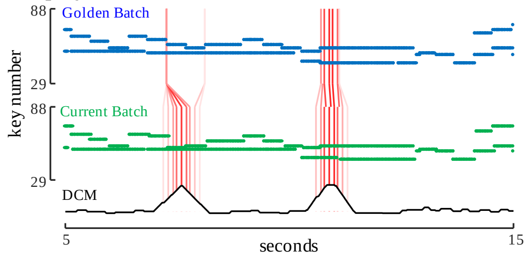

Figure 22: *top*) The golden batch for the same section of the song. *middle*) The same snippet shown in 0. *bottom*) The DCM computed by our method. The red lines indicate the alignment between the current batch and the golden batch.

6 Conclusion

In this work, we have introduced an online amnestic dynamic time warping for solving the golden batch problem. The proposed algorithm is *online* because it evaluates the correctness of each data point as it arrives, and it is *amnestic* as any deviation in the monitored data will only induce error locally, and if the deviation is corrected the error will decrease appropriately. We have showcased the *utility* of our algorithm by using it as an attention focusing algorithm, anomaly detection algorithm, and a personalized training and skill assessment on a set of *diverse* data sets from domains including: industry, transportation systems, and human handwriting.

In this work, we have introduced an online amnestic dynamic time warping for solving the golden batch problem. The proposed algorithm is *online* because it evaluates the correctness of each data point as it arrives, and it is *amnestic* as any deviation in the monitored data will only induce error locally, and if the deviation is corrected the error will decrease appropriately. We have showcased the *utility* of our algorithm by using it as an attention focusing algorithm, anomaly detection algorithm, and a personalized training and skill assessment on a set of *diverse* data

sets from domains including: industry, transportation systems, and human handwriting.

ACKNOWLEDGMENTS

We would like to thank all the donors of data. Thanks to Dr. Thomas Stahovich for his help with the pen-based computing apparatus, and the petrochemical engineer (who asked to remain anonymous) for his help and advice. This work was funded by NSF awards 1510741 and 1544969, by NIH R01-HD083431-01A1 and by a gift from MERL labs.

REFERENCES

[1] Aspen ProMV Brochure, https://www.aspentech.com/en/resources/brochure/aspen-promv-brochure.

[2] I. Assent, M. Wichterich, R. Krieger, H. Kremer, T. Seidl, "Anticipatory DTW for Efficient Similarity Search in Time Series Databases." PVLDB 2(1): 826-837 (2009).

[3] O. Bau, I. Poupyrev, A. Israr, and C. Harrison. "TeslaTouch: electrovibration for touch surfaces." In Proceedings of the 23nd annual ACM symposium on User interface software and technology, pp. 283-292. ACM, 2010.

[4] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41, no. 3 (2009): 15.

[5] S. Chiappino, L. Marcenaro, and C. S. Regazzoni, "Selective attention automatic focus for cognitive crowd monitoring," Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on. IEEE, 2013.

[6] H. A. Dau et al. "Optimizing dynamic time warping's window width for time series data mining applications." DMKD (2018): 1-47.

[7] M. Ewerton et al. "Assisting Movement Training and Execution with Visual and Haptic Feedback." Frontiers in Neurorobotics 12 (2018): 24.

[8] M. J. Fard, S. Ameri, and R. D. Ellis. "Toward Personalized Training and Skill Assessment in Robotic Minimally Invasive Surgery." In Proceedings of the World Congress on Engineering and Computer Science, vol. 2. 2016.

[9] S. Giraldo et al. "Automatic assessment of violin performance using dynamic time warping classification." In 2018 26th Signal Processing and Communications Applications Conference (SIU). IEEE, 2018.

[10] https://musescore.com/user/7639766/scores/2847181.

[11] F. Itakura, "Minimum prediction residual principle applied to speech recognition." IEEE Transactions on Acoustics, Speech, and Signal Processing 23.1 (1975): 67-72.

[12] P. Kah, M. Shrestha, E. Hiltunen, and J. Martikainen. "Robotic arc welding sensors and programming in industrial applications." International Journal of Mechanical and Materials Engineering 10, no. 1 (2015): 13.

[13] E. Keogh, J. Lin, and A. Fu. "HOT SAX: Finding the most unusual time series subsequence: Algorithms and applications." In Proc. ICDM, pp. 440-449. 2004.

[14] S. Mallat. A wavelet tour of signal processing. Elsevier, 1999.

[15] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano. "Early classification of time series by simultaneously optimizing the accuracy and earliness." IEEE Transactions on Neural Networks and Learning Systems (2017).

[16] R. T. Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. No. CMU-CS-01-108. Carnegie Mellon University, Pittsburgh, PA. 2001.

[17] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. "A review of novelty detection." Signal Processing 99 (2014): 215-249.

[18] F. Petitjean et al. "Dynamic time warping averaging of time series allows faster and more accurate classification." In ICDM on, pp. 470-479. IEEE, 2014.

[19] F. Petitjean, A. Ketterlin, and P. Gançarski. "A global averaging method for dynamic time warping, with applications to clustering." Pattern Recognition 44, no. 3 (2011): 678-693.

[20] D.M.W. Powers. "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation." Journal of Machine Learning Technologies, 2(1), 37-63. 2011.

[21] Project website: https://sites.google.com/view/gbatch.

[22] P. Qiu. Introduction to statistical process control. CRC Press, 2013.

[23] C. A. Ratanamahatana and E. Keogh. "Making time-series classification more accurate using learned constraints." Proceedings of the 2004 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2004.

[24] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." IEEE transactions on acoustics, speech, and signal processing 26.1 (1978): 43-49.

[25] D. F. Silva, G. E. A. P. A. Batista, and E. Keogh. "Prefix and suffix invariant dynamic time warping." Data Mining (ICDM), 2016 IEEE 16th International Conference on. IEEE, 2016.

[26] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli. "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation." Artificial intelligence in medicine 45.1 (2009): 11-34.

[27] Trendminer. Video retrieved on 4/15/2018 www.trendminer.com/use-case-compare-current-batch-to-golden-batch-fingerprint/.

[28] US Department of Health and Human Services, Food and Drug Administration. "Grade A Pasteurized Milk Ordinance 2015 Revision." 2015.

[29] E. Wang, Z. Yu, and P. Collings. "Dynamic control strategy of a distillation system for a composition-adjustable organic Rankine cycle." Energy 141 (2017): 1038-1051.

[30] R. Wojewodka and T. Blevins. "Data Analytics in Batch Operations." Control Global. May 4, 2008. Accessed April 14, 2018. https://www.controlglobal.com/articles/2008/164/.

[31] C.-C. M. Yeh, N. Kavantzas, and E. Keogh. "Matrix profile VI: Meaningful multidimensional motif discovery." In Data Mining (ICDM), 2017 IEEE International Conference on, pp. 565-574. IEEE, 2017.

[32] C.-C. M. Yeh et al. "Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile." DMKD 32, no. 1 (2018): 83-123.

[33] Y. Zhu et al.. "Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins." KIS (2018): 1-34.

[34] A. Zia, Y. Sharma, V. Bettadapura, E. L. Sarin, and I. Essa. "Video and accelerometer-based motion analysis for automated surgical skills assessment." International journal of computer assisted radiology and surgery 13, no. 3 (2018): 443-455.