Instance-Based Counterfactual Explanations for Time Series Classification

Eoin Delaney*, Derek Greene and Mark T. Keane
University College Dublin, Dublin, Ireland
Insight Centre for Data Analytics, UCD, Dublin, Ireland
VistaMilk SFI Research Centre
eoin.delaney@insight-centre.org, derek.greene@ucd.ie, mark.keane@ucd.ie

Abstract

In recent years there has been a cascade of research in attempting to make AI systems more interpretable by providing explanations; so-called Explainable AI (XAI). Most of this research has dealt with the challenges that arise in explaining black-box deep learning systems in classification and regression tasks, with a focus on tabular and image data; for example, there is a rich seam of work on post-hoc counterfactual explanations for a variety of black-box classifiers (e.g., when a user is refused a loan, the counterfactual explanation tells the user about the conditions under which they would get the loan). However, less attention has been paid to the parallel interpretability challenges arising in AI systems dealing with time series data. This paper advances a novel technique, called Native-Guide, for the generation of proximal and plausible counterfactual explanations for instance-based time series classification tasks (e.g., where users are provided with alternative time series to explain how a classification might change). The Native-Guide method retrieves and uses native in-sample counterfactuals that already exist in the training data as "guides" for perturbation in time series counterfactual generation. This method can be coupled with both Euclidean and Dynamic Time Warping (DTW) distance measures. After illustrating the technique on a case study involving a climate classification task, we reported on a comprehensive series of experiments on both real-world and synthetic data sets from the UCR archive. These experiments provide computational evidence of the quality of the counterfactual explanations generated.

1 Introduction

In recent years, the predictive success of machine learning systems has been undermined by their lack of interpretability, and there have been growing public calls for fairness, accountability, and transparency in the decisions of intelligent systems (Gunning 2017). These challenges have led to major efforts in Explainable AI (XAI) where a raft of techniques have been developed to shed light on opaque predictions. Many of these explanation methods provide users with post-hoc, example-based justifications such as factual explanations (i.e., "You were refused a loan because your profile is the same person X who was also refused"; see e.g., (Keane and Kenny 2019)) or post-hoc counterfactual explanations (i.e., "If your salary was higher you would have received the loan"; see e.g., (Byrne 2019; Mittelstadt, Russell, and Wachter 2019)). Most of this XAI research focuses on tabular and image data, with less attention being given to the explanation of time series data (Nguyen, Le Nguyen, and Ifrim 2020). Indeed, we have found no work showing how post-hoc counterfactual explanations (Miller 2019), could be computed for time series predictions. Hence, in this paper, we propose *Native-Guide*, a novel model agnostic post-hoc explanation technique for time series classifiers that provides factual and counterfactual explanations for their predictions.

Paper Outline: In the remainder of the introduction, we review related work on explaining time series models, before considering why counterfactual explanations might be especially useful. Then, in Section 2, we outline some of the problems that arise in finding and evaluating counterfactual explanations, before presenting *Native-Guide* as a solution and describing how to evaluate explanations (see Section 3). In Sections 4 and 5, we present a case study, involving a climate prediction task, and then report results from tests on 38 benchmark data sets from the UCR archive (Dau et al. 2019). Section 6 concludes with a consideration of issues arising from this work and opportunities for future research.

1.1 Related Work

From one perspective, given the flexibility, accuracy, and simplicity of k-NN techniques for time series prediction, the use of post-hoc, example-based explanations have often been a preferred option (Mueen and Keogh 2016; Cunningham and Delany 2020; Rudin 2019; Sokol and Flach 2020); any time series classification or prediction can easily be explained using a nearest neighbouring instance (e.g., Figure 1 which asserts "the climate in Rome should be like that of Athens because they have very similar weekly high-temperatures"). So, the success of baseline methods, using 1-NN-Euclidean and 1-NN-DTW (Dynamic Time Warping) techniques for time series classification (TSC) have made factual-example-based explanations the preferred choice (Bagnall et al. 2017). These white-box models are comprehensible and, since the explanations are generated directly from the underlying model, they are faithful by definition (Rudin 2019; Fauvel, Masson, and Fromont 2020). However, with the emergence of less transparent, deep learn-

*Corresponding Author

ing methods (Fawaz et al. 2019), the need to find some way to explain predictions has re-emerged as a challenge. For example, methods such as COTE and LSTM-FCN boast impressive predictive performance, but limited interpretability (Le Nguyen et al. 2019; Karim et al. 2017). Beyond factual, nearest-neighbour explanations, it is not at all clear how contrastive explanations for time series prediction might be computed, a new class of post-hoc explanations that is now attracting much attention (Miller 2019; Byrne 2019; Mittelstadt, Russell, and Wachter 2019).

Recent work has explored model-agnostic methods, such as Saliency Maps, LIME (Ribeiro, Singh, and Guestrin 2016), and SHAP, to explain time series classification (Schlegel et al. 2019). Class activation maps, shapelets, and dictionary-based approaches have been linked to interpretable prediction as they highlight the most influential sub-sequences of the time series at the time of classification (Wang et al. 2019; Karlsson et al. 2018). Instance-based approaches using prototypes have also been advanced for black-box deep learning classifiers, where the *prototype* instances are representative of a whole class (Gee et al. 2019; Molnar 2020). Class prototypes have also proved useful in guiding the formation of counterfactual explanations for image and tabular data (Van Looveren and Klaise 2019). However, it is not at all clear how these approaches can be applied to time series data where very different similarity measures (e.g., Dynamic Time Warping as opposed to weighted Manhattan distance) and feature-dimension partitioning methods (i.e., k-d trees (Van Looveren and Klaise 2019)) are used (Tan, Webb, and Petitjean 2017). As such, the problem of how to explain time series classifications largely still remains to be solved, especially with respect to the development of counterfactual explanations.

1.2 Why Counterfactual Explanations?

The best way to understand how counterfactual explanations might be used for time series classification is to consider how they differ from factual explanations. Consider a binary classification system which predicts that "the number of new people infected by Covid-19 in a certain region is set to increase", based on an analysis of the time series for case numbers over the last week. The system might explain its decision *factually* with the following statement: "The number of cases next week will increase because in the past a region with similar characteristics (population, restriction measures etc.) reported an increase in cases". In contrast, the system might explain its decision *counterfactually* with the following statement: "If you tightened the regulations using a local lockdown, then the number of cases would go down". There is a growing consensus that such counterfactual explanations are causally more informative (Lipton 2018; Pearl and Mackenzie 2018), psychologically effective (Miller 2019; Byrne 2019; Dodge et al. 2019; Keane and Smyth 2020; Molnar 2020), and legally acceptable with respect to GDPR, (Wachter, Mittelstadt, and Russell 2017). Some have specifically argued that counterfactuals provide more robust and informative explanations when compared to other model agnostic methods, such as LIME or SHAP (Guidotti et al. 2019). Finally, although it is difficult to visualise counterfactuals for tabular data (Mothilal, Sharma, and Tan 2020), for time series such visualisations are quite straight-forward. That is, if one can find a way to compute the counterfactual alternative, the visual presentation of that alternative is a given.

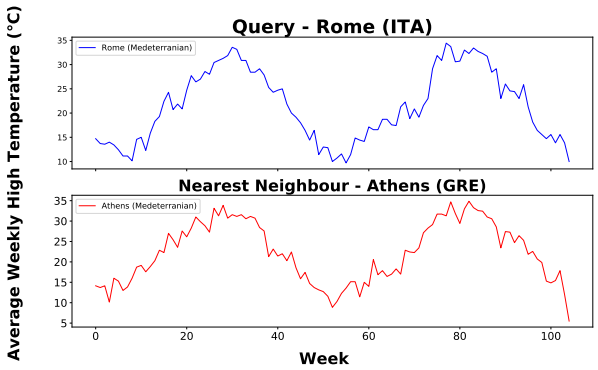

Figure 1: Factual Explanation of a 1-NN Euclidean Time Series Classification: The climate of Rome was unknown. It was queried to the training data. Athens was retrieved as Rome's nearest neighbour. Athens has a Mediterranean climate. Therefore we (correctly) predict that Rome also has a Mediterranean climate.

1.3 Contributions of this Work

This work introduces a novel model-agnostic method for generating counterfactual explanations for time series classifiers. Specifically, we show that:

- Counterfactual explanations can be built by retrieving existing in-sample instances that bear counterfactual-relationships to one another (i.e., *Native* counterfactuals) and then perturbing these to explain the predictions of a query posed to a k-NN time series classifier (hence, it is called the *Native-Guide* method)
- This Native-Guide method is flexible enough to be used with either Euclidean and DTW distance measures.
- This method consistently generates "good" counterfactuals for a wide range of data sets from the UCR repository.

2 Native-Guides for Counterfactuals

In this section, we motivate and describe *Native-Guide*, a method for generating good counterfactual explanations for time series classification. Many existing methods produce counterfactuals by implementing "blind" perturbations without referencing the existing data (Keane and Smyth 2020). This strategy tends to result in large numbers of candidate explanations (i.e., the so-called Rashoman effect (Rudin 2019)), many of which will be invalid observations (i.e., they are actually misleading as they fall outside the sample). Furthermore, given the number of possible feature dimensions in time series data, this solution quickly becomes intractable. Instance-based solutions for counterfactual generation often rely on the strategy of finding/generating the closest instance

to the to-be explained prediction that involves a class change (Laugel et al. 2018; Nugent, Doyle, and Cunningham 2009; Keane and Smyth 2020). Hence, for time series data, the current method relies on using instances from the existing data set (so-called "native in-sample guides"), to generate counterfactual candidates to explain time series predictions. In the following sub-sections, we summarise the prerequisite definitions adopted in defining the method, before describing the algorithm and how it retrieves and perturbs native guides.

2.1 Prerequisite Definitions

The following definitions are adopted in the current method (see also Figure 2):

Definition 1 ***Time Series Data Set.*** *A time series* $T =< t_1, ..., t_L >$ *is an ordered set of real values, where* L *is the length. A time series data set* $\boldsymbol{D} = T_1, ..., T_N$ *is a collection of such time series.*

Definition 2 ***Time Series Classification Task.*** *A time series data set* $\boldsymbol{D}$ *contains* N *time series, each of which has a class label* p*. The goal of time series classification is to distinguish to which class an unlabelled query time series belongs.*

Definition 3 ***Distance Function*** *A distance function* $d(T_a, T_b)$ *takes two time series* T_a *and* T_b *as inputs and returns a non negative value as an output which we refer to as the distance between the two time series. We require that this function is symmetric i.e.* $d(T_a, T_b) = d(T_b, T_a)$

2.2 Retrieving Native-Guides

A Native-Guide is a counterfactual instance that already exists within the data set (i.e., the nearest-neighbouring time series of the query time series which involves a class change). In Figure 1, the queried time series is represented by an X and the corresponding native -guide is labelled accordingly. We can retrieve this instance using a 1-NN search (see Algorithm 1).

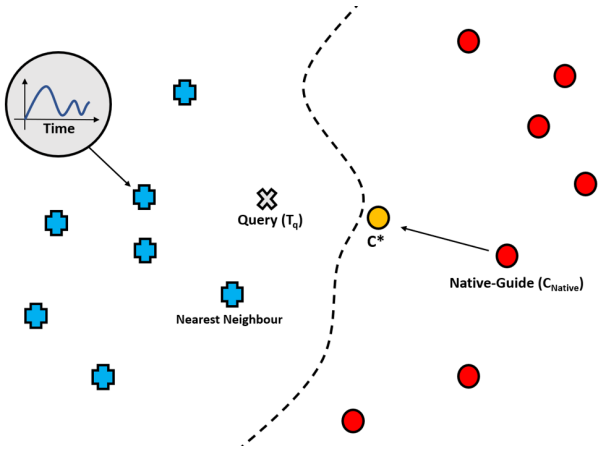

Figure 2: A time series data set for a binary classification task with two class labels. A query time series T_q (represented as X) and it's Native-Guide (C_{Native}). The generated counterfactual $(C*)$ is represented in yellow.

Algorithm 1 Native-Guide Retrieval

Input: T_q: Query time series with predicted class p
Input: d: A distance function (e.g. Euclidean, DTW)
Input: $D_{p'} \subseteq D$: The set of time series $\{T_1, ..., T_n\}$ in the training data with class label p'
Initialize: $d(T_q, C_{Native}) = \infty$, $C_{Native} = \emptyset$
for $i = 1, 2, ...n$: **do**
    compute $\delta = d(T_q, T_i)$
    if $\delta < d(T_q, C_{Native})$: **then**
        Set $C_{Native} = T_i$
        Set $d(T_q, C_{Native}) = \delta$
    end if
end for
Return: C_{Native}

2.3 Perturbing Native-Guides

The retrieved native-guide is then perturbed towards the query until just before there is a class change in the base classifiers prediction (yellow instance in Figure 2). We outline how to generate such instances with reference to Figure 2;

1. **Inputs**
   - T_q: The queried time series
   - C_{Native}: The retrieved in-sample counterfactual Native-Guide
2. **Initialize**
   - Start at Native-Guide (C_{Native})
3. **Perturbation**
   - Perturb C_{Native} towards T_q using a suitable weighted average function (see arrow in Figure 2)
   - Classify the newly generated instance
4. **Stopping-Condition**
   - Terminate process just before we cross the decision boundary
   - **Return** $C*$: Generated counterfactual instance (Instance in yellow Figure 2.)

This method ensures that, rather than blindly perturbing the time series to generate a counterfactual explanation, we are perturbing from the existing data. The approach is designed to ensure that the generated counterfactual is close to the query (i.e., closer than the native-guide), while it remains within the distribution of the data. That is, formally:

$$d(Q, C*) < d(Q, C_{Native}) \tag{1}$$

Of course, the above allows for different distance measures d to be used. Next, we elaborate on such measures which are commonly employed in conjunction with time series data: Euclidean distance and Dynamic Time Warping.

Euclidean Case. The simplest approach is to implement a weighted perturbation strategy, defined as

$$C* = (\beta \times Q) + ((1 - \beta) \times C_{Native}) \tag{2}$$

where $\beta \in [0, 1]$ is a weight controlling the resemblance of the generated counterfactual $C*$ to the native counterfactual C_{Native}. As we increase the weight on the guided perturbation to inherit properties of the query, the counterfactual should generate better explanations as it will be in closer proximity to the query.

Dynamic Time Warping (DTW) Case. Although Euclidean distance is a popular baseline distance metric for time series classification tasks, a 1-NN classifier can achieve state-of-the-art performance when using Dynamic Time Warping (DTW) as a distance measure (Mueen and Keogh 2016) DTW is based on Levenshtein distance, and was originally introduced for application in speech recognition (Sakoe and Chiba 1978). It finds the optimal alignment between two sequences of numerical values, and captures flexible similarities by aligning the coordinates inside both sequences (Petitjean, Ketterlin, and Gançarski 2011), which can be appropriate if two time series are out of phase. In order to compute an average or weighted time series using DTW we are tasked with aligning similar sub-sequences instead of element wise matching as in the Euclidean case. This adds a layer of abstraction and complexity to the process and invites a different form of perturbation. Here we apply a global averaging technique known as Dynamic Barycenter Averaging (DBA) (Petitjean, Ketterlin, and Gançarski 2011). This approach aims to compute an "average" sequence, called the *barycenter*, which ensures that the sum of squared DTW distance between the barycenter and the set of considered sequences is minimized. This technique has been applied to generate synthetic time series to augment sparse data sets and significantly improve classification accuracy (Forestier et al. 2017): defined as follows:

Definition 4 ***Weighted average of time series under DTW.*** *Given a weighted set of time series* $\boldsymbol{D} = (T_1, \beta_1), ..., (T_N, \beta_n)$*, the average time series under DTW,* $\overline{T}$*, is the time series that minimizes:*

$$argmin\, \overline{T} \sum_{i=1}^{N} \beta_i \cdot DTW^2(\overline{T}, T_i) \tag{3}$$

This technique generates in-distribution time series data, suggesting that it should tend to generate feasible explanations. By adjusting the weights on the query and in-sample counterfactual instance, we can generate a new time series that offers a better explanation when compared to the in-sample, native counterfactual.

3 Evaluating Counterfactual Goodness

Although a common consensus on the best approach to evaluate counterfactual explanations is still emerging; there is a general agreement that the explanations should be (i) Similar to the query (Mittelstadt, Russell, and Wachter 2019; Keane and Smyth 2020) and (ii) Within the distribution of the data (Sokol and Flach 2020). Hence, in our evaluations we use a Relative Counterfactual Distance (RCF) measure (Keane and Smyth 2020) and an Out-of-Distribution count (OOD) measure, which are described below.

3.1 Relative Counterfactual Distance (RCF)

It is generally assumed that good counterfactuals are ones in which the original query and the explanation-instance are close on some distance measure. Typically, a counterfactual that is in close proximity to the query offers the best explanation (Mothilal, Sharma, and Tan 2020). A relative counterfactual distance (RCF) measure was proposed by (Keane and Smyth 2020) to assess if a generated counterfactual is closer to the query than the native in-sample counterfactual.

$$RCF = \frac{d(Q, C*)}{d(Q, C_{Native})} \tag{4}$$

If RCF > 1 the in-sample counterfactual is closer to the query than the generated counterfactual and if RCF < 1 the generated counterfactual is closer. We use the distance metric used in the underlying classifier (i.e., it can be either Euclidean, or DTW , but could be any distance metric).

3.2 Measuring Out of Distribution (OOD)

Counterfactual instances are not necessarily representative of the underlying data distribution and may be based on invalid data-points (Poyiadzi et al. 2020). We cast the evaluation of if a counterfactual is in-distribution as a novelty detection problem. When we have a data set of n observations from the same distribution described by p features and we add another observation to that data set, novelty detection methods can tell us if the new observation is within the distribution of the original data or an outlier (Pedregosa et al. 2011). Common novelty detection algorithms include one class Support Vector Machines (Schölkopf et al. 2000; Ma and Perkins 2003), Isolation forests (Liu, Ting, and Zhou 2008) and Local Outlier Factors (Breunig et al. 2000). As we are already using instance-based methods, we implement Local Outlier Factor (LOF) to evaluate if the generated counterfactual explanations are plausible and representative of the existing data. LOF evaluates if the generated counterfactuals "make sense" or fit the distribution of the existing data. This is done by measuring the local deviation of a given data point with respect to its neighbours. LOF quantifies how isolated a point is with regard to the density of its neighbourhood. LOF depends on a single parameter *k* which indicates the number of nearest neighbours to consider. Recent work has shown that LOF is scalable for large data sets, and can leverage modern distributed infrastructures making it suitable for our task. A good technical summary of LOF can be found in (Yan, Cao, and Rundensteiner 2017). In the next section, we provide a case study using the Native-Guide method and evaluate it using these measures (see Section 4).

4 Case Study: Climate Classification

As a case study we examine a climate prediction task in some detail. There is evidence that AI advances will support the understanding of climate change and the modeling of its possible impacts (Vinuesa et al. 2020). Under the UNs *AI for Good Platform*; Goal 13 targets the understanding and

Query	NN-Medeterranian	NN-Oceanic	Actual	Predicted	$d(Q, C*)$	OOD	RCF
Amsterdam	Salamanca (57.587)	London (23.193)	0	0	36.280	0	0.620
Rome	Athens (26.379)	Milan (33.126)	1	1	5.300	0	0.160

Table 1: Subset of case study results (Euclidean).

prediction of climate change. The task in this case-study is to explain the predictions of a 1-NN time series classification-system; specifically, it is a binary classification-task where system classifies a city as having an Oceanic climate (class: 0) or a Mediterranean climate (class: 1) based on the average weekly high temperatures over a period of two years. The system uses Weather data from 40 cities over a two-year period from 2017–2019, to capture seasonal effects and trends. The WorldWeather API was used to collect the data. The true labels are from the Koppen-Geiger classification system (Kottek et al. 2006). Explaining the systems predictions can provide us with insights into the conditions that would cause the classifications to change. The main goal of the case study is to portray the intuition behind and method of using in-sample counterfactual instances as Native-Guides for counterfactual generation. All code, data, and results are included in Supplementary Material.

Experimental Setup

Twenty cities were used for training data and twenty for testing data, with an equal representation of each class present in each of the splits to mitigate class imbalance. Nearest neighbours are retrieved along with the in-sample counterfactual and the corresponding distance: $d(Q, C_{Native})$ is recorded. We generate counterfactual explanations for instances and record the distance between the generated counterfactual and the query $d(Q, C*)$. The RCF is calculated to determine the proximity of the counterfactual to the query relative to the in-sample counterfactual. The LOF (parameter k set to $\sqrt{len(X_{train} + X_{test})}$) is used to determine the OOD measure which evaluates if the generated counterfactuals are within the distribution of the data. A subset of results for instance retrieval can be found in Table 1 with a full collection of data, results and code to be found in Supplementary Material A.

Explaining Climate Classifications

In this section, we will look at explanations for specific instances. We also show how contrastive explanations can be visualized for time series data. In this example, Amsterdam (NED) was queried (see Figure 3) and three different explanatory instances generated by the system are shown:

- *A. Factual explanation:* The true label for the query, Amsterdam (NED), according to Koppen Climate Classification is a Oceanic climate. In the training data its nearest neighbour is London (UK), where $d(Amsterdam, London) = 23.193$, which also has an Oceanic climate. Therefore, the system correctly classifies Amsterdam's climate to be Oceanic (Figure 3A). Hence, this factual explanation could be summarised as: "Amsterdam has an Oceanic climate because it is most similar to London, which has an Oceanic climate too".

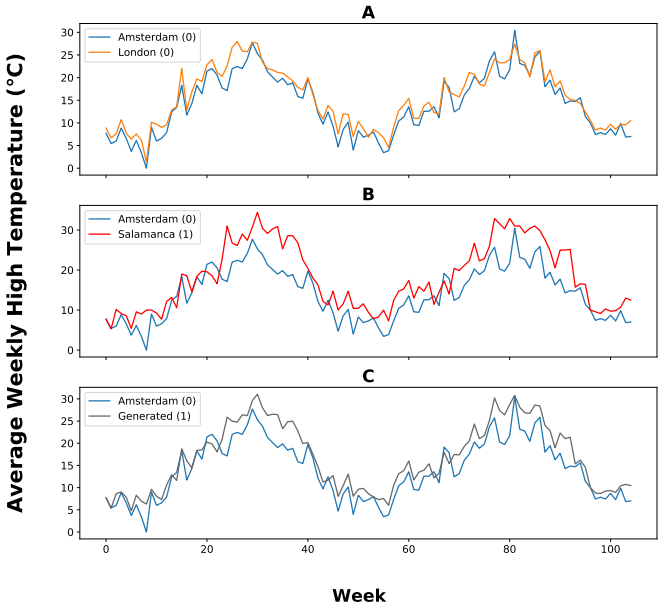

Figure 3: Classification explanations for Amsterdam (NED) predictions, in the climate change case study.

- *B. In-sample counterfactual explanation:* The system can also find an in-sample counterfactual to the "correct" Oceanic classification, in the form of the city of Salamanca (ESP), which is in the training data, where $d(Amsterdam, Salamanca) = 57.587$, which is classed as having a Mediterranean climate (Figure 3B). This counterfactual explanation could be glossed as saying: "If Amsterdam had the same weather as Salamanca the system would classify it as having a Mediterranean climate".

- *C. Generated counterfactual:* Finally, the systems can generate a counterfactual (that should be better than the in-sample case), that is a perturbation of the Salamanca data, but closer to the query (i.e., Salamanca's summer highs are noticeably hotter than Amsterdam's). As the underlying classifier predicts this generated counterfactual to be in the Mediterranean class, this explanation can be glossed as saying "If Amsterdam had a weather profile like the Generated-Instance then system would classify it as as having a Mediterranean climate" (Figure 3C). This generated explanation is much closer to the query $d(Q, C*) = 36.28$. It is also within the distribution of the existing based off LOF.

Data set	Train	Test	CF	β-mean	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
BeetleFly	20	20	5	0.800	25.734	5.009	0	0.195
BirdChicken	20	20	9	0.739	18.528	5.014	0	0.271
Chinatown	20	343	19	0.766	712.065	203.432	0	0.286
DistalPhalanxOutlineCorrect	600	276	78	0.783	1.350	0.307	7	0.227
Earthquakes	322	139	40	0.963	30.217	1.115	0	0.037
ECG200	100	100	12	0.757	5.071	1.444	0	0.285
ECGFiveDays	23	861	175	0.797	6.321	1.361	0	0.215
FreezerRegularTrain	150	2850	556	0.615	3.180	1.384	8	0.435
FreezerSmallTrain	28	2850	924	0.559	6.763	3.044	398	0.450

Table 2: Subset of results for Experiment 1 (Euclidean Distance).

Data set	Train	Test	CF	β-mean	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
BeetleFly	20	20	6	0.880	7.939	3.713	0	0.468
BirdChicken	20	20	5	0.898	7.949	1.228	0	0.154
Chinatown	20	343	9	0.752	477.050	158.242	0	0.332
DistalPhalanxOutlineCorrect	600	276	78	0.734	0.692	0.297	3	0.429
Earthquakes	322	139	39	0.855	6.841	2.225	0	0.325
ECG200	100	100	23	0.877	1.953	0.593	1	0.304
ECGFiveDays	23	861	200	0.719	2.620	0.823	14	0.314
FreezerRegularTrain	150	2850	288	0.602	1.121	0.465	21	0.415
FreezerSmallTrain	28	2850	687	0.578	2.591	0.951	177	0.367

Table 3: Subset of results for Experiment 1 (Dynamic Time Warping).

5 Multiple Data sets Experiment (for Euclidean and DTW)

In order to determine if *Native-Guide* can generate good counterfactual explanations for a variety of different data sets, we apply the proposed method on 38 diverse benchmark data sets from the UCR archive (Dau et al. 2019). We report results for the key evaluative measures for the performance of the system (i.e., RCF and OOD).

5.1 Data

We evaluate our method on 38 diverse binary time series classification tasks from the UCR archive (Dau et al. 2019). Examples of these tasks include heartbeat anomaly detection, traffic-flow management, and earthquake classification. The archive has been incrementally extended and contains a substantial collection of real-world and synthetic data from multiple domains and problem types. It is perhaps the most common used benchmark for recent studies on time series classification (Le Nguyen et al. 2019). In order to promote reproducibility, we performed all experiments on the default single training-test split set specified by the benchmark.

5.2 Experimental Setup

Native-Guides are retrieved from the corresponding time series data set using 1-NN instance retrieval with a Euclidean distance metric and then a DTW distance measure (Algorithm 1). The distance between the in-sample counterfactual and the query $d(Q, C_{Native})$ is recorded. Next we generate counterfactual using guided perturbation instance adaption. In the DTW implementation we deploy weighted-DBA to generate counterfactual time series for explanatory purposes. The idea is analogous to the Euclidean implementation as we iteratively approach the query from the perspective of the Native-Guide. We apply a weight β to each time series and gradually increase the weight towards the query until just before there is a class change (see Figure 2). We record $d(Q, C*)$, the distance between the query and the generated counterfactual. Counterfactuals are evaluated based on Proximity (RCF) and Plausibility (OOD) using Local Outlier Factor with parameter k set to $\sqrt{len(X_{train} + X_{test})}$. In the cognitive sciences evidence suggests that people tend to create counterfactuals to imagine how an outcome could have been better instead of worse (Byrne 2019). Therefore, we focus on generating counterfactuals for misclassifications. For each data set we note the number of misclassifications for the base classifier, which is also equal to the number of counterfactuals (CF) that we generate.

5.3 Results and Discussion

The proposed method generates counterfactuals that are significantly closer to the query compared to the in-sample counterfactuals when using both Euclidean and DTW distance measures (see Figure 4 and RCF values in Figure 5). As some of the data sets are not normalized we perform a Man-Whitney U test to statistically confirm that there is a significant difference between $d(Q, C*)$ and $d(Q, C_{Native})$; Euclidean ($p < 0.01$), DTW ($p < 0.01$). Native-Guide also produces plausible counterfactual explanations that are representative of real time series data. Few of the counterfactuals are out-of-distribution (526 out of the 3570 generated counterfactuals for the Euclidean experiment, and 371 out

of 3041 for the DTW implementation). It should be noted that the FreezerSmallTrain data set accounts for the majority (398 Euclidean, 177 DTW) of these OOD values. This is possibly due to the fact that this data set has a small amount of training data compared to the test data indicating that the proposed method works best when there is a diverse availability of training data. Indeed in the companion data set FreezerRegularTrain (more training data) the OOD counts are much lower (8 Euclidean, 21 DTW).

The mean β value across the data sets (0.79 Euclidean, 0.82 DTW) indicates that the generated counterfactual explanations typically have a very strong resemblance to the query. This also suggests that the global properties of the time series such as trend and seasonality have been preserved. In summary, Native-Guide consistently generates good counterfactual explanations for time series classification tasks that are proximal and plausible and the method can easily be coupled with both Euclidean and DTW distance measures. For space reasons, a subset of results can be found in Tables 2 and 3, with full results provided in Supplementary Material B.

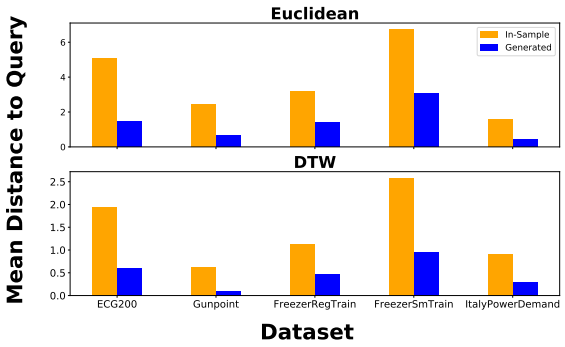

Figure 4: Counterfactual proximities for five representative UCR data sets, indicating that the Native-Guide method can generate counterfactuals which are close to the query.

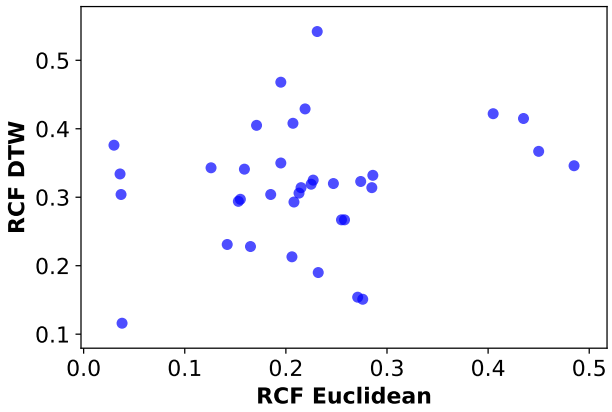

Figure 5: RCF value over all data sets highlighting the generation of proximal counterfactuals (Euclidean and DTW).

6 Conclusion

In this paper a novel method, Native-Guide, was proposed to provide good counterfactual explanations for time series classification tasks. Our method generates counterfactuals based on reference instances that exist in the data set to generate better counterfactual explanations. The method was tested on a case study and a collection of diverse binary data sets from the UCR archive using 1-NN Euclidean and 1-NN DTW classifiers which are benchmark algorithms in this domain due to their simplicity and accuracy. Native-Guide consistently generated counterfactual explanations that were in close proximity to the query and plausible as they were within the distribution of the data. Moreover, these counterfactuals were significantly better than explanations that already existed in the training data. Our method generates new time series data which also holds promise for data augmentation and improving classification performance for sparse data sets (Forestier et al. 2017).

One limitation of our approach is that we are perturbing the whole time series, rather than the most influential sub sequences. Future work will focus on adding sparsity constraints to the generation of counterfactual explanation that are flexible with Euclidean and Dynamic Time Warping distance measures. This indicates the promise of deep learning approaches and the development of a suitable loss function for counterfactual explanation in time series. Throughout the research we link counterfactual explanations in time series to the relevant evidence from psychology and the cognitive sciences (Miller 2019; Byrne 2019). Evaluating our explanations and comparing them to other methods by means of a user-study is another avenue for future research

Implementation and Reproducibility

The *tslearn* library was extensively used in this research (Tavenard et al. 2020). Experiments were implemented on a Dell XPS15 Laptop; Processor: Intel Core i7-10875H; Memory: 16GB DDR4-2133MHz. For the DTW experiment, our experiments were batched and executed in parallel. A single algorithm run was needed to produce results. Full details of this can be found in the supplementary material. We include all data and code for the case study in Supplementary Material A. The UCR data is publicly available to download (Dau et al. 2019), and we include all code needed to reproduce our results for the corresponding Euclidean and DTW experiments in Supplementary Material B.

Acknowledgements

This paper emanated from research funded by (i) Science Foundation Ireland (SFI) to the Insight Centre for Data Analytics (12/RC/2289 P2), (ii) SFI and DAFM on behalf of the Government of Ireland to the VistaMilk SFI Research Centre (16/RC/3835)

References

Bagnall, A.; Lines, J.; Bostrom, A.; Large, J.; and Keogh, E. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31(3): 606–660.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93–104.

Byrne, R. M. 2019. Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning. *IJCAI International Joint Conference on Artificial Intelligence* 2019-August: 6276–6282. ISSN 10450823. doi: 10.24963/ijcai.2019/876.

Cunningham, P.; and Delany, S. J. 2020. k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples) (1): 1–22. URL http://arxiv.org/abs/2004.04523.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica* 6(6): 1293–1305.

Dodge, J.; Liao, Q. V.; Zhang, Y.; Bellamy, R. K.; and Dugan, C. 2019. Explaining models: an empirical study of how explanations impact fairness judgment. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 275–285.

Fauvel, K.; Masson, V.; and Fromont, É. 2020. A Performance-Explainability Framework to Benchmark Machine Learning Methods: Application to Multivariate Time Series Classifiers. *arXiv preprint arXiv:2005.14501* .

Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33(4): 917–963.

Forestier, G.; Petitjean, F.; Dau, H. A.; Webb, G. I.; and Keogh, E. 2017. Generating synthetic time series to augment sparse datasets. In *2017 IEEE international conference on data mining (ICDM)*, 865–870. IEEE.

Gee, A. H.; Garcia-Olano, D.; Ghosh, J.; and Paydarfar, D. 2019. Explaining deep classification of time-series data with learned prototypes. *CEUR Workshop Proceedings* 2429: 15–22. ISSN 16130073.

Guidotti, R.; Monreale, A.; Giannotti, F.; Pedreschi, D.; Ruggieri, S.; and Turini, F. 2019. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems* 34(6): 14–23.

Gunning, D. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* 2: 2.

Karim, F.; Majumdar, S.; Darabi, H.; and Chen, S. 2017. LSTM fully convolutional networks for time series classification. *IEEE access* 6: 1662–1669.

Karlsson, I.; Rebane, J.; Papapetrou, P.; and Gionis, A. 2018. Explainable time series tweaking via irreversible and reversible temporal transformations. In *2018 IEEE International Conference on Data Mining (ICDM)*, 207–216. IEEE.

Keane, M. T.; and Kenny, E. M. 2019. How case based reasoning explained neural networks: an XAI survey of post-hoc explanation-by-example in ANN-CBR twins. *arXiv preprint arXiv:1905.07186* .

Keane, M. T.; and Smyth, B. 2020. Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI). *Proceedings of the 28th International Conference on Case-Based Reasoning, ICCBR 2020* .

Kottek, M.; Grieser, J.; Beck, C.; Rudolf, B.; and Rubel, F. 2006. World map of the Köppen-Geiger climate classification updated. *Meteorologische Zeitschrift* 15(3): 259–263.

Laugel, T.; Lesot, M.-J.; Marsala, C.; Renard, X.; and Detyniecki, M. 2018. Comparison-based inverse classification for interpretability in machine learning. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 100–111. Springer.

Le Nguyen, T.; Gsponer, S.; Ilie, I.; O'Reilly, M.; and Ifrim, G. 2019. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Mining and Knowledge Discovery* 33(4): 1183–1222. ISSN 1573756X. doi:10.1007/s10618-019-00633-3. URL https://doi.org/10.1007/s10618-019-00633-3.

Lipton, Z. C. 2018. The mythos of model interpretability. *Communications of the ACM* 61(10): 35–43. ISSN 15577317. doi:10.1145/3233231.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, 413–422. IEEE.

Ma, J.; and Perkins, S. 2003. Time-series Novelty Detection Using One-class Support Vector Machines. *Proceedings of the International Joint Conference on Neural Networks* 3: 1741–1745.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267: 1–38.

Mittelstadt, B.; Russell, C.; and Wachter, S. 2019. Explaining explanations in AI. In *Proceedings of the conference on fairness, accountability, and transparency*, 279–288.

Molnar, C. 2020. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617.

Mueen, A.; and Keogh, E. 2016. Extracting optimal performance from dynamic time warping. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2129–2130.

Nguyen, T. T.; Le Nguyen, T.; and Ifrim, G. 2020. A Model-Agnostic Approach to Quantifying the Informativeness of Explanation Methods for Time Series Classification. In *Proceedings of the 5th Workshop on Advanced Analytics and Learning on Temporal Data at ECML 2020.* Springer.

Nugent, C.; Doyle, D.; and Cunningham, P. 2009. Gaining insight through case-based explanation. *Journal of Intelligent Information Systems* 32(3): 267–295.

Pearl, J.; and Mackenzie, D. 2018. *The Book of Why*. New York: Basic Books. ISBN 978-0-465-09760-9.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Petitjean, F.; Ketterlin, A.; and Gançarski, P. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44(3): 678–693. ISSN 00313203. doi:10.1016/j.patcog.2010.09.013.

Poyiadzi, R.; Sokol, K.; Santos-Rodriguez, R.; De Bie, T.; and Flach, P. 2020. FACE: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 344–350.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 13-17-August-2016: 1135–1144. doi:10.1145/2939672.2939778.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1(5): 206–215. ISSN 25225839. doi:10.1038/s42256-019-0048-x.

Sakoe, H.; and Chiba, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26(1): 43–49.

Schlegel, U.; Arnout, H.; El-Assady, M.; Oelke, D.; and Keim, D. A. 2019. Towards a rigorous evaluation of XAI Methods on Time Series. *arXiv preprint arXiv:1909.07082* .

Schölkopf, B.; Williamson, R. C.; Smola, A. J.; Shawe-Taylor, J.; and Platt, J. C. 2000. Support vector method for novelty detection. In *Advances in neural information processing systems*, 582–588.

Sokol, K.; and Flach, P. 2020. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 56–67.

Tan, C. W.; Webb, G. I.; and Petitjean, F. 2017. Indexing and classifying gigabytes of time series under time warping. In *Proceedings of the 2017 SIAM international conference on data mining*, 282–290. SIAM.

Tavenard, R.; Faouzi, J.; Vandewiele, G.; Divo, F.; Androz, G.; Holtz, C.; Payne, M.; Yurchak, R.; Rußwurm, M.; Kolar, K.; and Woods, E. 2020. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research* 21(118): 1–6. URL http://jmlr.org/papers/v21/20-091.html.

Van Looveren, A.; and Klaise, J. 2019. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584* .

Vinuesa, R.; Azizpour, H.; Leite, I.; Balaam, M.; Dignum, V.; Domisch, S.; Felländer, A.; Langhans, S. D.; Tegmark, M.; and Nerini, F. F. 2020. The role of artificial intelligence in achieving the Sustainable Development Goals. *Nature communications* 11(1): 1–10.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31: 841.

Wang, Y.; Emonet, R.; Fromont, E.; Malinowski, S.; Menager, E.; Mosser, L.; and Tavenard, R. 2019. Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization. *arXiv preprint arXiv:1906.00917* .

Yan, Y.; Cao, L.; and Rundensteiner, E. A. 2017. Distributed Top-N local outlier detection in big data. In *2017 IEEE International Conference on Big Data (Big Data)*, 827–836. IEEE.

Technical Appendix

The full tables of results for the experiments can be found in the pages below. All code needed to generate these results as well as the csv result files are available in the code + data supplementary file section. The code can be requested by emailing the corresponding author.

- Table 4: Case-Study Euclidean Results
- Table 5: Case-Study (DTW) Results
- Table 6: UCR Euclidean Results
- Table 7: UCR DTW Results

Some Details on UCR Archive Data sets

All data sets used in this experiment are publicly available on the UCR archive. The original DodgerLoopGame and DodgerLoopWeekend data sets have missing values so we use the adjusted versions of these data sets which are also available on the archive under the *Missing Value And Variable Length Data sets Adjusted* section. The Coffee and GunpointAgeSpan data sets have no misclassifications. Therefore we do not generate any counterfactuals for these data sets and details of this can also be found in the code section.

Query	NN-Medeterranian	NN-Oceanic	Actual	Predicted	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
Nice	Salamanca (34.292)	Paris (32.120)	1	0	34.292	3.429	0	0.10
Los Angeles	Beirut (32.815)	Milan (74.194)	1	1	74.194	44.516	0	0.60
Rome	Athens (26.379)	Milan (33.126)	1	1	33.126	5.300	0	0.16
Zurich	Salamanca (49.670)	Ljubljana (21.391)	0	0	49.670	28.808	0	0.58
Tirana	Istanbul (30.361)	Milan (32.931)	1	1	32.931	3.293	0	0.10
Belfast	San-Francisco (74.891)	Dublin (10.447)	0	0	74.891	41.190	0	0.55
Zonguldak	Istanbul (20.924)	Paris (43.892)	0	1	43.892	17.996	0	0.41
Santander	San-Francisco (33.568)	London (36.206)	0	1	36.206	2.534	0	0.07
Ohrid	Istanbul (49.299)	Vienna (32.443)	1	0	49.299	16.269	0	0.33
Seattle	Istanbul (47.793)	London (39.452)	1	0	47.793	10.036	0	0.21
Brussels	Salamanca (49.285)	London (16.570)	0	0	49.285	30.064	0	0.61
Barcelona	Lisbon (30.255)	Milan (40.338)	1	1	40.338	10.488	0	0.26
Split	Istanbul (23.688)	Milan (29.362)	1	1	29.362	4.992	0	0.17
Copenhagen	Salamanca (79.255)	Vancouver (34.457)	0	0	79.255	52.308	0	0.66
Munich	Salamanca (61.880)	Berlin (23.965)	0	0	61.880	40.222	0	0.65
Perth	Cape-Town (30.633)	Melbourne (48.642)	1	1	48.642	20.430	0	0.42
Amsterdam	Salamanca (57.587)	London (21.393)	0	0	57.587	36.280	0	0.63
Auckland	Cape-Town (37.491)	Melbourne (34.964)	0	0	37.491	3.374	0	0.09
Santiago	Cape-Town (34.654)	Melbourne (37.563)	1	1	37.563	3.756	0	0.10
Lille	Salamanca (48.290)	London (14.264)	0	0	48.290	28.974	0	0.60

Table 4: Results Case Study (Euclidean).

Query	NN-Medeterranian	NN-Oceanic	Actual	Predicted	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
Nice	Istanbul (16.419)	London (18.984)	1	1	18.984	1.713	0	0.090
Los Angeles	Beirut (19.981)	Melbourne (31.810)	1	1	31.810	25.550	0	0.803
Rome	Athens (14.013)	Milan (21.664)	1	1	21.664	6.681	0	0.308
Zurich	Istanbul (25.977)	Ljubljana (16.775)	0	0	25.977	10.320	0	0.397
Tirana	Athens (16.213)	Milan (18.025)	1	1	18.025	2.285	0	0.127
Belfast	San-Francisco (39.630)	Dublin (8.972)	0	0	39.630	20.180	0	0.509
Zonguldak	Istanbul (15.667)	London (17.304)	0	1	17.304	17.441	0	1.008
Santander	San-Francisco (16.128)	London (24.369)	0	1	24.369	5.951	0	0.244
Ohrid	Istanbul (27.230)	London (21.655)	1	0	27.230	5.279	0	0.194
Seattle	Istanbul (21.366)	London (18.414)	1	0	21.366	4.257	0	0.199
Brussels	Istanbul (20.673)	London (14.632)	0	0	20.673	13.088	0	0.633
Barcelona	Beirut (15.004)	Milan (29.069)	1	1	29.069	11.464	0	0.394
Split	Istanbul (13.632)	Paris (18.361)	1	1	18.361	8.993	0	0.490
Copenhagen	Istanbul (37.021)	Vancouver (16.347)	0	0	37.021	11.627	0	0.314
Munich	Istanbul (29.651)	Berlin (16.230)	0	0	29.651	12.191	0	0.411
Perth	Cape-Town (18.593)	Melbourne (23.365)	1	1	23.365	9.053	0	0.387
Amsterdam	Istanbul (21.826)	London (16.459)	0	0	21.826	11.900	0	0.545
Auckland	Cape-Town (15.359)	Melbourne (21.829)	0	1	21.829	5.990	0	0.274
Santiago	Cape-Town (19.889)	Melbourne (20.109)	1	1	20.109	20.178	0	1.003
Lille	Istanbul (19.466)	London (12.895)	0	0	19.466	15.362	0	0.789

Table 5: Results Case Study (DTW).

Data set	Train	Test	CF	β-mean	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
BeetleFly	20	20	5	0.800	25.734	5.009	0	0.195
BirdChicken	20	20	9	0.739	18.528	5.014	0	0.271
Chinatown	20	343	19	0.766	712.065	203.432	0	0.286
Computers	250	250	106	0.781	22.642	4.948	0	0.219
DistalPhalanxOutlineCorrect	600	276	78	0.783	1.350	0.307	7	0.227
DodgerLoopGame	20	138	16	0.834	156.624	30.500	0	0.195
DodgerLoopWeekend	20	138	2	0.515	223.807	108.510	0	0.485
Earthquakes	322	139	40	0.963	30.217	1.115	0	0.037
ECG200	100	100	12	0.757	5.071	1.444	0	0.285
ECGFiveDays	23	861	175	0.797	6.321	1.361	0	0.215
FreezerRegularTrain	150	2850	556	0.615	3.180	1.384	8	0.435
FreezerSmallTrain	28	2850	924	0.559	6.763	3.044	398	0.450
GunPoint	50	150	13	0.742	2.413	0.665	0	0.276
GunPointAgeSpan	135	316	10	0.794	294.335	68.297	0	0.232
GunPointMaleVersusFemale	135	316	2	0.955	440.251	16.776	0	0.038
Ham	109	105	42	0.804	8.339	1.717	0	0.206
Herring	64	64	31	0.840	3.480	0.552	0	0.159
HouseTwenty	40	119	38	0.819	39828.102	7370.020	0	0.185
ItalyPowerDemand	67	1029	46	0.700	1.590	0.436	1	0.274
Lightning2	60	61	15	0.763	21.817	5.035	0	0.231
MiddlePhalanxOutlineCorrect	600	291	68	0.835	0.738	0.126	2	0.171
MoteStrain	20	1252	152	0.835	8.675	1.432	69	0.165
PhalangesOutlinesCorrect	1800	858	205	0.807	0.840	0.174	17	0.207
PowerCons	180	180	4	0.872	12.618	1.595	0	0.126
ProximalPhalanxOutlineCorrect	600	291	56	0.791	0.343	0.073	0	0.213
SemgHandGenderCh2	300	600	61	0.964	477.636	17.169	0	0.036
ShapeletSim	20	180	83	0.970	30.736	0.927	0	0.030
SonyAIBORobotSurface1	20	601	183	0.796	6.041	1.259	5	0.208
SonyAIBORobotSurface2	27	953	134	0.787	7.684	1.727	3	0.225
Strawberry	613	370	20	0.772	0.651	0.168	1	0.258
ToeSegmentation1	40	228	73	0.849	17.162	2.619	0	0.153
ToeSegmentation2	36	130	25	0.858	18.890	2.680	0	0.142
TwoLeadECG	23	1139	288	0.756	2.089	0.532	0	0.255
Wafer	1000	6164	28	0.758	9.344	2.308	15	0.247
Wine	57	54	21	0.564	0.173	0.070	0	0.405
WormsTwoClass	181	77	30	0.839	29.917	4.625	0	0.155

Table 6: Results for UCR Experiment (Euclidean Distance).

Data set	Train	Test	CF	β-mean	$d(Q, C_{Native})$	$d(Q, C*)$	OOD	RCF
BeetleFly	20	20	6	0.880	7.939	3.713	0	0.468
BirdChicken	20	20	5	0.898	7.949	1.228	0	0.154
Chinatown	20	343	9	0.752	477.050	158.242	0	0.332
Computers	250	250	75	0.664	7.639	4.127	7	0.542
DistalPhalanxOutlineCorrect	600	276	78	0.734	0.692	0.297	3	0.429
DodgerLoopGame	20	138	12	0.928	67.731	23.693	0	0.350
DodgerLoopWeekend	20	138	6	0.775	93.587	32.342	0	0.346
Earthquakes	322	139	39	0.855	6.841	2.225	0	0.325
ECG200	100	100	23	0.877	1.953	0.593	1	0.304
ECGFiveDays	23	861	200	0.719	2.620	0.823	14	0.314
FreezerRegularTrain	150	2850	288	0.602	1.121	0.465	21	0.415
FreezerSmallTrain	28	2850	687	0.578	2.591	0.951	177	0.367
Gunpoint	50	150	14	0.964	0.636	0.096	0	0.151
GunPointAgeSpan	135	316	5	0.946	62.549	11.877	0	0.190
GunPointMaleVersusFemale	135	316	5	0.958	161.445	18.677	2	0.116
Ham	109	105	56	0.917	4.866	1.035	0	0.213
Herring	64	64	30	0.943	0.886	0.302	0	0.341
HouseTwenty	40	119	19	0.777	15333.661	4661.996	0	0.304
ItalyPowerDemand	67	1029	51	0.772	0.909	0.294	1	0.323
Lightning2	60	61	8	0.806	6.161	3.337	0	0.542
MiddlePhalanxOutlineCorrect	600	291	88	0.774	0.479	0.194	0	0.405
MoteStrain	20	1252	207	0.812	5.292	1.209	46	0.228
PhalangesOutlinesCorrect	1800	858	233	0.764	0.498	0.203	6	0.408
PowerCons	180	180	14	0.875	3.846	1.319	0	0.343
ProximalPhalanxOutlineCorrect	600	291	63	0.799	0.294	0.090	1	0.306
SemgHandGenderCh2	300	600	51	0.971	212.511	70.979	0	0.334
ShapeletSim	20	180	63	0.979	13.461	5.056	0	0.376
SonyAIBORobotSurface1	20	601	165	0.815	3.409	0.999	0	0.293
SonyAIBORobotSurface2	27	953	161	0.847	4.091	1.306	1	0.319
Strawberry	613	370	22	0.852	0.416	0.111	0	0.267
ToeSegmentation1	40	228	52	0.878	6.488	1.906	1	0.294
ToeSegmentation2	36	130	21	0.867	7.477	1.724	0	0.231
TwoLeadECG	23	1139	109	0.839	0.970	0.259	0	0.267
Wafer	1000	6164	124	0.726	3.312	1.060	90	0.320
Wine	57	54	23	0.580	0.154	0.065	0	0.422
WormsTwoClass	181	77	29	0.928	10.765	3.197	0	0.297

Table 7: Results for UCR Experiment (Dynamic Time Warping).