6-18-2021

A Method for Compact Representation of Heterogenous and Multivariate Time Series for Robust Classification and Visualization

Alla Abdella
University of South Florida

A Method for Compact Representation of Heterogenous and Multivariate Time Series for Robust Classification and Visualization

by

Alla Abdella

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ismail Uysal, Ph.D.
Nasir Ghani, Ph.D.
Yasin Yilmaz, Ph.D.
Alessio Gaspar, Ph.D.
Mohammed Elmusrati, Ph.D.

Date of Approval:
June 17, 2021

Keywords: Sequence Modeling, Sense2Vec, Class2Vec, Data Compression

Dedication

To my Parents and Brothers, for everything they have done for me.

To my Friends, for supporting me and believing in me always.

To my Teachers, for all the knowledge that helped me reach this level.

Acknowledgments

The past few years at University of South Florida (USF) have been unforgettable, invaluable and among the best exciting moments in my life. While at USF I not only learned how to do fundamental research, but also had the opportunity to meet with many talented and innovative people in many different fields. It would not have been possible to complete this journey without the help and support of so many people I feel deeply indebted to.

First and foremost, my greatest thanks to my Ph.D. advisor, Dr. Ismail Uysal for his unlimited guidance and support of my personal and professional growth during my graduate studies. He is one of the best professors I have ever met and I realize every day how privileged I am to work with him. When I first joined Dr. Uysal's group, it was a transitional period from wireless communications into machine and deep learning, and Dr. Uysal accepted me without hesitation. He was super patient with me and taught me how to be systematic, rigorous and professional. He always has an insightful, high-level view of the field, with an uncommon detail oriented approach who understands the nature of the problems very well. More importantly, Dr. Uysal is an extremely kind, caring and supportive advisor and I could not have asked for more. He is like an older brother of mine (if he doesn't mind me saying so) and I can talk with him about everything. He has done everything he could, as an educator, and a mentor, to help me become a better researcher. I believe Dr. Uysal provided me with the support and guidance that will benefit my future academic career.

I would also like to extend my thanks and appreciation to Prof. Nasir Ghani, Prof. Yasin Yilmaz, Prof. Mohammed Elmusrati and Prof. Alessio Gaspar for serving on my dissertation committee and their attention, guidance, and invaluable feedback during my candidacy exam and through this final step of completing my Ph.D. I have also been very fortunate to be taught by Dr. Yilmaz the Advanced Data Analytic class where I learned so much from him. I also would like

to thank Dr. Elmusrati who is an extremely caring, enthusiastic and knowledgeable person and I always feel my passion ignited after talking to him or reading his online posts. I look forward to working with Dr. Yilmaz and Dr. Elmusrati in the future. I am likewise grateful to Dr. Ghani as he has always been supportive of me and believed in me since early in my graduate studies. I am also grateful for Dr. Gaspar for his time serving in my committee and his feedback on my work which I had to incorporate to the best extent possible. Last but not least, I cherished the opportunity to work with Prof. Jeffrey Brecht from the University of Florida as part of my graduate studies.

I thank my mentor Dr. Adnan Massood who I continue to learn from and be inspired by every day. His trust and help has sharpen my skills in the data science field in handling large scale projects. It has been an honor and a wonderful experience working with him during my work at UST company.

I would also like to thank all other collaborators from both academia and industry whom I didn't all list here: without their generous sharing of resources, our work would not have been possible. I am very grateful for the support I received during my work at USF. It must be noted that parts of my research was funded by the United States Department of Agriculture (USDA) and the Florida Department of Agriculture and Consumer Services (FDACS). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of USDA, and FDACS. I would like to thank our collaborators for their support of the project including WishFarms for allowing us to conduct the shipping tests and coordinating all the logistics and DeltaTrak for donating the real-time loggers used to collect the data in the study.

Finally, and most importantly, I appreciate my family. They have always been a source of strength and my inspiration. I would like to thank my mother, Reem Zain for her unconditional love, care, and support during my entire life. I love her so much and I would not have made it this far without her. I also would like to dedicate this work to honor my older brother ISMAIL who passed away with a dream of seeing me a doctor. Today our dream comes true my beloved brother! I would like to thank my brothers Sensoussi, Farge, Mohammed, Abdella, Edriss and little sister

Hamedia for their unconditional support, and love. They are the reason for me to be here. For me, they are always the source of strength and inspiration. My mother and family made me who I am today and I never know how to pay them back. I hope that they are at least a little proud of me for what I have become today.

Lastly, I would like to thank my wife Doaa for her love and support. We got married on 2017, I was doing my Master's when I first met Doaa and we have been experiencing almost everything together since then: from my graduate studies to applying to jobs, to raising our wonderful children Omar and Zaid with her unconditional support and feedback on my work. She is not only my partner, my wife, my best friend, but also the person I admire, for her modesty, intelligence, concentration and hard work. Without her, I would not have reach to this stage. I thank Doaa for everything she has done for me.

Table of Contents

List of Tables

List of Figures

Abstract

Processing multivariate sensory time-series with variable lengths is a challenging problem across different application domains due to the naturally complex, high-dimensional, and often non-stationary nature of the data. There are many practical examples of this in the industry particularly for the applications of sensor networks in monitoring production or distribution of goods around the globe. This thesis tackles the specific problem of time-series data representation in how we can better summarize and visualize multi-variate time series data coming from numerous sources in a sensor network distributed across a wide range of application scenarios. On one hand, we think the analysis and processing of multivariate and heterogeneous time-series data is important for predictive tasks like regression and classification. On the other hand, if we build novel systems and methods for data summarization and visualization, they would be crucial components in gathering actionable and robust insight while ensuring accurate analytics down the information chain. This thesis consists of two main parts in its contributions:

- In the first part, we aim to cover the statistical and temporal analysis of a novel multivariate time-series data for food engineering, and present our effort in proposing a new approach (Sense2Vec) for processing variable-length sensory time-series data that leverages various similarity metrics while being robust to noise and outliers. We believe that the proposed representation holds a great promise for future time-series visualization technologies.

- In the second part of this thesis, we introduce a supervised Class2Vec algorithm as an application of Sense2Vec where each class is represented by a single blueprint profile learned from the training dataset. Class2Vec uses dynamic time warping distances between different observations and the class blueprints to create a novel classification framework with an

unprecedented compression of the time series training data. We evaluate this framework thoroughly on 70 different datasets hosted on the UCR Archive.

Our results clearly demonstrate that while Sense2Vec provides a novel and compressed form of data representation for time series data on multi sensor applications, Class2Vec incorporates meaningful data knowledge in generating blueprint class labels when compared to a baseline algorithm for a domain-agnostic approach.

Chapter 1: Introduction

1.1 History of Artificial Intelligence

The pursuit of artificial intelligence led to a surge of interest in imitating the human brain and how it learns, leading to the dawn of “machine learning”. Machine Learning (ML) is defined by Arthur Samuel (IBM) in 1959 as “A subset of AI that often uses statistical techniques to give machines the ability to "learn" from data without being explicitly programmed to do so.” ML can be summarized as follows:

$$MachineLearning = Representation + Objective + Optimization \tag{1.1}$$

While initially infeasible due to hardware and data constraints, recent advancements in Graphics Processing Units (GPUs) and the proliferation of data have brought machine learning algorithms like neural networks to the forefront of attention. Traditional machine learning requires domain knowledge within the field to which it is being applied, in order to inform which features should be selected for use in the learning algorithm. However, further exploration into automating this feature-learning process led to the advancement of "deep learning".Deep learning (DL) is a type of machine learning that performs this automated feature engineering - deep neural networks (DNNs) are used to learn abstract representations and discover hidden structures from the data. The learning is governed by tuning parameters using the backpropagation approach [2]. The relationship between DL, machine learning, neural networks and artificial intelligence is shown in Figure 1.1. DL is flourishing in the new era of automated-learning machines that deal with massive amounts of data. DL has a wide application area, from identifying landscape features like mountains, lakes and buildings from satellite imagery to image processing, segmentation, object

tracking and detection in computer vision to enhancing the user experience in online marketing. Machine learning applications in the computer vision domain were the main motivation for constructing DL algorithms. Applications such as transcribing speech into text, finding outliers in massive data and clustering similar news articles, products, documents and music all make use of DL. The main advantage of DL as opposed to its ancestors is the feature-learning process. The latter used tedious, hard-coded instructions to solve a specific problem and, as a result, suffered the curse of dimensionality [3]. As solving problems that require a large amount of data is inevitable in today's world, the birth of the former was a necessity.

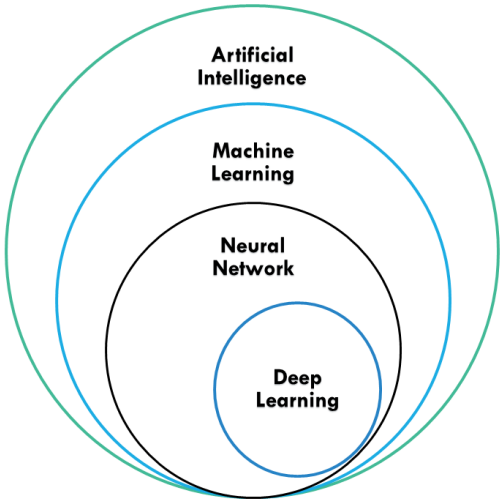

Figure 1.1 The relationship between Artificial Intelligence (AI), Machine learning (ML), Neural Networks (NN), and Deep Learning (DL) as shown by [4].

In 1989, [5] incorporated automatic feature learning into a convolutional operation-based neural network trained by a constrained backpropagation algorithm. The machine learning community believes this is the first convolutional neural network to recognize handwritten digits from US mail. His work was motivated by the weight-sharing technique presented by Hinton in 1986 that reduces the number of learned parameters significantly. Also, in contrast to previous attempts in the literature, the author used a hyperbolic tangent function in the hidden nodes to achieve high convergence. The network was task-specific, however, and highly constrained. Geoffry Hinton introduced an unsupervised, undirected stochastic model called the Boltzmann Machine, which

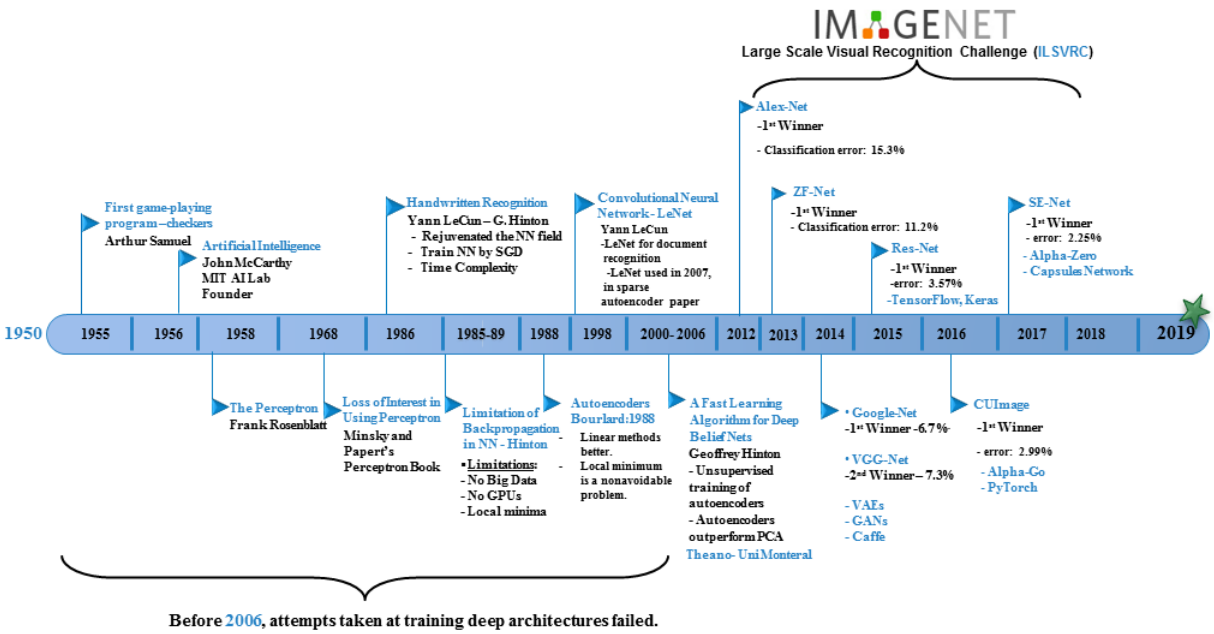

Figure 1.2 Breakthroughs of machine learning in history.

was the first DL model to search for an energy function that maps the input space into another hidden space. Hinton's work modeled the intractable joint probability distribution between the visible and hidden units and then inferred the underlying hidden structure of the presented binary data. He implemented a gradient-based algorithm to solve the problem of different encoder structures. However, Hinton did not apply the algorithm to a variety of complex problems, and did not show how well it generalizes to unseen data points. The proposed Boltzmann network was also slow to train, which discouraged the machine learning community from exploring it further. As a result, the Boltzmann Machine, or the restricted version of it, are not commonly used as deep neural networks, as they are slow to train using backpropagation.

DL involves architectures that are constructed by stacking multiple layers sequentially within an artificial neural network. It is based on computing a non-linear function that maps the input space into an output space, in order to improve the network to be unresponsive to translational

variance and improve selectivity. In the case of image detection, the motivation behind having deep layers is to allow the network to be meticulous in distinguishing similar images of different objects and robust towards illumination variability, pose variability, occlusion and surrounding objects. While hand-crafted features have been used for many years, it requires a high level of expertise and knowledge in the problem domain. However, researchers during the 1970s and 1980s independently discovered the idea of simply training an artificial neural network using backpropagation with stochastic gradient descent to learn the network's weights and biases. The use of the chain rule of derivatives allows for computing the gradients of the objective function with respect to the network parameters [3]. Historically, the computer vision community was not confident that neural networks were a feasible approach to learn and extract the useful features in classification and recognition applications, as it was commonly thought that gradient descent would become stuck in local minima and the weights would not be updated towards the direction of the global minimum. However, in the 1990s, [6] published the first convolutional neural network named ConvNets that classified the MNIST dataset and was trained by backpropagation. LeCun's paper showed that ConvNets was more efficient in training and generalized to never-before-seen data with a small number of parameters. With the unprecedented practical success of ConvNets over the artificial dense neural network, it has since gotten more and more attention from the computer vision community. The most successful marriage between the computer vision and machine learning communities happened in 2012, when [7] applied a deep convolutional network to ImageNet, a set of millions of images from 1000 different categories, and won the ImageNet competition with better accuracy than all other existing approaches which more readily relies on conventional feature engineering [3]. While the revolution in computer-vision-based neural networks began with parallel computing, innovations like data augmentation, dropout for regularization [7], combining of different network architectures for image captions, the ReLU non-linear function [8], ADAM optimization [9] and open-source libraries like Tensorflow, Keras and Pytorch all continue to contribute to the success of DL in computer vision applications. The main breakthroughs throughout the history of machine learning and DL are summarized by Figure 1.2, and Figure 1.8.

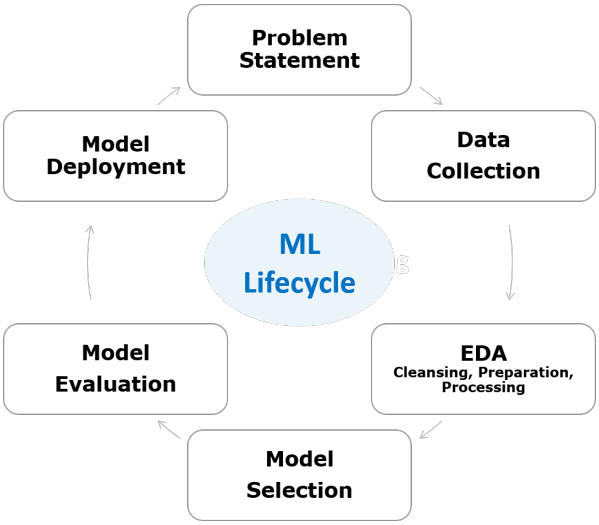

Figure 1.3 Machine learning life-cycle according to CRISPDM methodology.

The ML life-cycle is shown by 1.3. It is inspired by cross-industry standard process for data mining (CRISPDM) methodology [10] which considers six phases: 1. Understanding the project objectives and requirements; 2. Data collection, description, quality checks for outliers and missing values analysis, data exploration and final preparation; 3. Feature engineering scaling numerical variables, sampling, correlation and averaging; 4. Machine learning/DL modeling for learning the mapping of non-parametric functions for regression and classification (find optimum parameters and hyperparameters for learning). 5. Choosing the right evaluation metric based on the problem definition and application (ensures that the model properly achieves the project objectives); and finally 6. Deploying the learned model into production for practical usage.

1.2 Perceptron

The perceptron is the simplest and oldest model of a neuron inside the human brain [11] which constitutes the bedrock of generic learning models from linear regression to multi-layered neural networks. It can be defined as a node that receives some input x weighted by parameter w to output some value y. Given a set of observations $X = (x_1, x_2, ..., x_m)$, and a set of weights $W = (w_1, w_2, ..., w_m)$, the perceptron first calculated a weighted intermediary sum z:

$$z = \sum_{i=1} w_i x_i \tag{1.2}$$

From which the output is computed by an activation function $f(.)$ which may or may not be linear:

$$\hat{y} = f(z) \tag{1.3}$$

A basic illustrative model of the perceptron is shown by Figure 1.4.

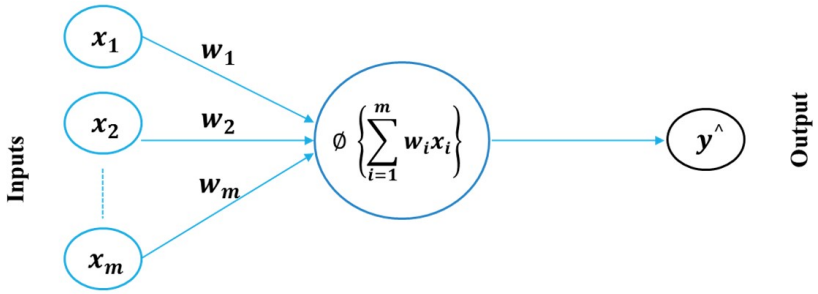

Figure 1.4 Basic model of the perceptron

1.3 Neural Networks

Neural Networks (NNs) build on the idea of the perceptron unit and consist of a stack of layers with multiple neurons or nodes in them. A simple NN consists of one input layer that receives information (i.e. features) and passes its output to the next layer (called the hidden layer) that applies a non-linear function to its weighted input. The output layer receives its input from the previous hidden layer and provides the final prediction. Figure(a) 1.6 shows a simple neural network architecture. Deep neural networks are constructed by increasing the depth and complexity

of the network by adding more layers with more neurons in them to learn more complex functions to map the inputs to the desired outputs as shown in (Figure(b) 1.6). The true advantage of deep neural networks is their ability to extract low level and abstract (latent) features from the raw input data in an automated fashion. As the input data flows from lower hidden layers to upper layers, the network learns different representations of the data, going from simple to more complex. Formally, $f : x \rightarrow y$ is a function (ReLU [12], Softmax [13], Sigmoid, and Tansh [14], etc.) that maps a set of input features x, i.e. a set of image pixel values, into some predicted output y, which represents the network belief on which class this input image belongs to.

1.4 Feedforward Pass

Suppose we have m observations $X = (x_1, x_2, ..., x_m)$ and each observation $x_i \subset \mathbb{R}^d$. Assume we have L layers in the deep network and each layer l_i consists of n number of neurons. For simplicity, we assume all the hidden layers have the same number of nodes. j^{th} node in layer l_i receives features x^n_{l-1} from all the n nodes in layer $l_i - 1$ weighted by w_{l_i-1,l_i}, where w_{l_i-1,l_i} is the weighting matrix between layer l_i and $l_i - 1$ to determine the output of node n^l_j as follows:

$$y_{i,j} = f\left\{\sum_{i=1}^{l-i} w^n_{l_i-1,l_i} x^n_{l-1}\right\} \tag{1.4}$$

$$y_{i,j} = f\left\{\sum_{i=1}^{l-i} w^n_{l_i-1,l_i} x^n_{l-1}\right\} \tag{1.5}$$

And each node's output is passed through a nonlinear activation function $f(.)$ where the choice of $f(.)$ can vary even within the same network. The most popular nonlinear activation functions according to [[3], [15]] include:

- Rectified Linear Unit (ReLU) Function:

$$f(y) = max(0, y) \tag{1.6}$$

- Sigmoid - Logistic Function:

$$f(y) = \frac{1}{1 + e^{-y}} \tag{1.7}$$

- Hyperbolic Tangent Function:

$$f(y) = \frac{e^{+y} - e^{-y}}{e^{+y} + e^{-y}} \tag{1.8}$$

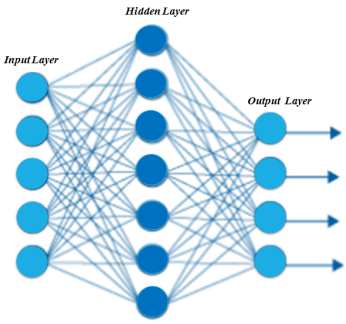

Figure 1.5 Simple Neural Network

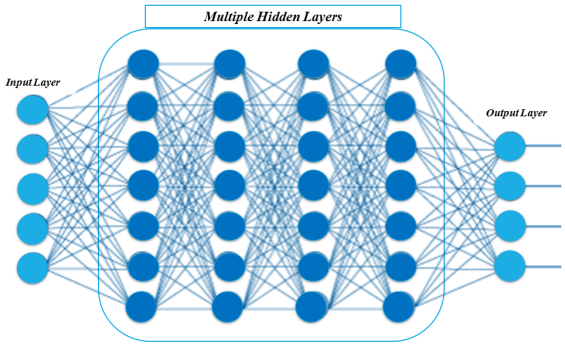

Figure 1.6 Deep Neural Network

1.5 Backward Pass (Backpropagation)

In the forward pass, the output of each node is computed and passed on to the nodes in the next layer. The number of nodes, layers and how they are connected defines the network architecture. The parameters (also known as weights and biases) define the learning process. Backpropagation is an algorithm to learn and adjust the parameters of the neural network to optimize an objective function to minimize error or maximize entropy. Before training, all of the network parameters are randomly initialized. After the first forward pass through the network, the backpropagation algorithm computes the error gradients from the output layer back to the input layer and updates the parameter values of each connection between the nodes of the connected layers. A brief pseudo description of the algorithm is provided below.

Algorithm I: Backpropagation

Input: loss function ł, learning rate η, activation function $\sigma_l = \mathrm{f}(\mathrm{W}_l^T h_{l-i} + b_l)$,

predicted output $\hat{y}$, target output y

Compute the gradient: $\delta \leftarrow \frac{\partial l(y,\hat{y})}{\partial y}$

fori $\leftarrow l$ to 1 **do**

Calculate the gradients for present layer and backpropagate

to the previous layer until you reach the input layer using the chain rule as follows:

$$\frac{\partial l(y,\hat{y})}{\partial W_l} = \frac{\partial l(y,\hat{y})}{\partial h_l}\frac{\partial h_l}{\partial W_l} = \delta\frac{\partial h_l}{\partial W_l} \tag{1.9}$$

$$\delta \leftarrow \frac{\partial l(y,\hat{y})}{\partial h_l}\frac{\partial h_l}{\partial h_{l-1}} = \delta\frac{\partial h_l}{\partial h_{l-1}} \tag{1.10}$$

end

1.6 Hyperparameters

Hyperparameters are determined prior to the optimization of neural network's internal parameters (i.e. weights and biases). Hyperparameters generally define the model structure and topology, improve the overall network performance, optimize the network training time by speeding up the process, etc. Even though hyperparameters are critical to the model performance, there is no predefined method in how we select them. Hyperparameters can be categorized into two sets: structural and functional. The former determines the network's topological specifications such as the number of hidden layers and the number of neurons in each layer, whereas the latter controls the training process such as the learning rate, momentum, weight decay, etc.

The table 1.1 lists neural network hyperparameters related to network topology and the best practices in choosing them based on highly cited literature. And table 1.2 lists neural network hyperparameters related to the training algorithm and the best practices in choosing them based on highly cited literature.

1.7 Machine Learning Approaches

Machine learning can be mainly classified into four areas: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Table 1.1 Network structure hyperparameters and their descriptions

Hyperparameter	**Choice Description**
Topology (number of layers, and number of units per layer)	Number of layers determines the depth of the network. The larger the number of layers, the deeper the neural network. Overfitting is a common problem in deep multilayered networks. The effect of overfitting on the learning process can be minimized by adding regularization techniques. On the other hand, reducing the number of layers may cause underfitting.
Dropout probability	A popular regularization technique implemented per layer in a neural network [16]. It is used to prevent overfitting that results from learning the statistical noise in the training set. The basic idea is to prevent the co-adaptive behavior between neurons by randomly dropping a set of them during the training to improve novel test performance. However, dropout introduces another hyperparameter: the probability of dropping out neurons and their connections. It has been reported that common values for $p \in (0.5, 0.8)$.
Parameter initialization	Carefully initializing the network parameters (i.e. weights and biases) prevents slow convergence which is also affected by the activation function used. The variance of the randomly initialized parameters can be adjusted by different schemes (Xavier [17], layer sequential uniform variance [18], random [7], [19] initialization to train deep rectifier networks, etc). Two main problems can be encountered by poorly initializing the deep neural network. Exploding gradient results in model instability and divergence as the error gradients accumulate during backpropagation. On the contrary, vanishing gradient causes the neurons in the early layers to converge slowly as the error gradient quickly fades as it approaches the input layer. For instance, in image detection, the earlier neurons are responsible for learning some low-level features (i.e edges, lines, etc) and if they are not properly trained because of gradient vanishing, it might lead to poor network performance.
Activation functions	The most popular nonlinear functions are: Sigmoid, Hyperbolic tangent, rectified linear unit (ReLU) [8], and exponential linear unit (ELU) [20]. In contrast to ReLUs, ELUs have negative values that helps center the mean around zero, which enables faster learning. Developing activation functions for deep neural networks is still an active area of research.

Table 1.2 Network training hyperparameters and their descriptions

Hyperparameter	**Choice Description**
Learning rate	The learning rate is mostly determined empirically. One of the recent common approaches proposed in the literature is to decrease or increase the learning rate during training based on the oscillations or plateau of the weight vector, respectively. A well-established second derivative method was introduced by [21] to improve the convergence by not relying on one global learning rate for all the weights but to choose different rates for different weights where "the lower layers should have larger learning rates than the higher ones".
Momentum	Momentum [22] helps speed up the learning during training for faster convergence. Momentum can be seen as a way of smoothing the high curvature surfaces using the momentum term μ, which is a value $\in (0,1)$ and an additional hyperparameter. A typical value for μ is between 0.5 and 0.9. In stochastic gradient descent with momentum, it computes the weighted averages of the gradients which is then used as a multiplier for the learning rate to update the weights and biases of the neural network.
Mini-batch size	A thorough analysis on the subset of benchmark datasets was recently conducted by [23] which showed that the best test performance is always achieved with a batch size of 32 where the entire dataset is divided into mini-batches of 32 training examples.
Number of epochs	Epoch is the number of times the entire dataset is fed forward and backward to the network during the training. As optimizing the cost function using gradient descent is an iterative process, it is a common practice in the literature to pass the entire dataset multiple times to the network while training. As is the case with many hyperparameters, there is no right answer on how to choose the number of epochs for a specific dataset. However, observing the evaluation loss on a validation set and how it changes with respect to the number of epochs is a common practice (called early stopping). The training is stopped when the evaluation accuracy decreases for a certain number of consecutive iterations.

1.7.1 Supervised Learning

Supervised learning has been implemented extensively in many machine learning domains, such as engineering, biology, health sciences, marketing and military applications. In supervised learning, the task is to learn distinguished features from the input observations aided by the true labels of those observations. Predicting temperature values as in regression models or classifying an image into one of a number of possible categories are the most common applications of supervised learning. More formally, the learning of simple and complex patterns is refined by searching for the best input-to-output mapping function from the hypothesis space in which an objective function can be optimized. As the learning is defined by the parameters that are tuned during training, those unknown model parameters are simultaneously adjusted via backpropagation to minimize the loss function, i.e. their values are optimal. Thus, labeled data is needed for any supervised algorithm to perform a narrow or general task. To illustrate, consider a classification task using the convolutional neural network proposed by [6]. Our goal is to build a model that can classify the Fashion-MNIST [24] data set with high accuracy. Fashion-MNIST is a labeled fashion data set consisting of ten different classes that has been recently used to test the accuracy of algorithms instead of MNIST since its intrinsic structure is more challenging. First, the data is split into two non-overlapping sets: the training set and the testing set. During the training stage, a series of consecutive and distinguished batches of observations (constructed as a tuple of instance and target pairs) are fed into the CNN, which uses a softmax output layer to produce a vector of ten posterior probabilities reflecting the network belief about each observation of the data. Our goal is to increase the output probability for the desired class, and this can be achieved by minimizing the error between the ground truth and the predicted output value. As we feed more batches into the CNN during each epoch, the learned parameters, namely the kernel or filter weights, will approach their optimal values since they will be constantly adjusted in the opposite direction of the gradient. These real-valued and shared weights are convolved with a small sliding window from the previous input layers to form the next feature maps. The amount that each weight is adjusted is controlled by stochastic gradient descent (SGD). SGD works as a guide for the weights to move

in the direction where the average error is decreased and the objective function is optimized. Finally, the performance of the supervised model is measured by feeding it the unlabeled testing set. This testing procedure validates how well the CNN generalizes to classifying new data. We also test whether our network overfits by memorizing the training data or generalizes by learning those patterns and providing a sensible answer for never-before-seen data.

1.7.2 Unsupervised Learning

Supervised learning can be seen as a teacher-based approach, an approach that has played and continues to play an indispensable role in the long history of human development [25]. However, much of our intelligence is attained without a teacher [26]. Unsupervised learning models, thus, are trained with unlabeled data and governed by hard and soft clustering. Clustering is an exclusively intrinsic exploratory algorithm [27] that has been used extensively in a variety of applications, ranging from analyzing large datasets to discovering pairwise similarity between features within the same group to uncover some hidden shared characteristics between high-dimensional variables. The goal of clustering is to divide the data into different clusters based on some similarity measure (e.g. Euclidean, Cosine, Jaccard distances, etc.) where the distance of elements belonging to different clusters (i.e. inter-cluster distance) is maximized, and the distance of elements belonging to the same cluster (i.e. intra-cluster distance) is minimized [28]. However, the quality of clustering results depends on the algorithm, the distance function and the application. Recently, with the advance of DL, autoencoders and specifically deep autoencoders have gained significant traction in analysis and clustering of large scale datasets.

1.7.3 History of Autoencoders

The autoencoder network was first proposed by [29] as a linear method to compress the input feature space into a lower-dimensional space. Bourlard concluded that linear approaches like singular value decomposition could be used to learn the optimal parameters of a neural network instead of training it via the backpropagation algorithm, which suffers from local minima. However,

Bourlard's claim was demolished by [30], who fine-tuned the autoencoder network's parameters to reduce the reconstruction loss using gradient descent. The adaptive model of the multi-layer neural network was constructed to reduce the high-dimensional data into lower-dimensional compact codes. The original data points were then reconstructed using these codes as an input to a decoder network. The authors showed that the autoencoder's representation of input features is better than that of a linear-based approach such as principal component analysis. They tested their model by inputting two datasets into the encoder network: MNIST (a dataset of images of handwritten digits) and Reuter (documents of newswire stories). They measured the distance of similarity between the compressed codes using the cosine of the angle and analyzed how varying the similarity measure could affect the reconstruction error.

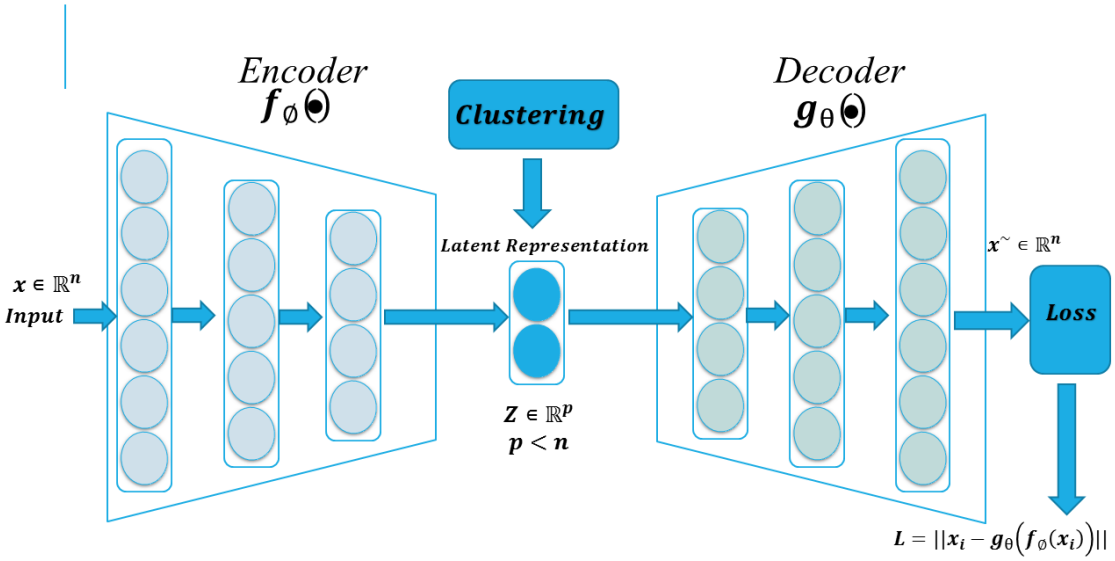

Figure 1.7 Clustering Based Autoencoder Architecture [1].

1.7.4 Deep Autoencoder Architecture

The goal is to generate a code (i.e. latent variable) using a non-linear complex function (i.e. encoder) that can be fed into another neural network (i.e. decoder) to re-generate the input data. In terms of space, the encoder network projects input feature vectors that span the high-dimensional feature space into a low-dimensional latent space spanned by the learned latent vectors. Then,

those latent samples are fed into the decoder network to reconstruct the original data. (Figure 1.7) shows the basic architecture of a deep autoencoder. The encoder is constructed by a set of feed-forward layers parameterized by ϕ that compute a code vector from an input data, for example, an image. The bottleneck in the center of the network represents the compressed representation of the input features that lie on a non-linear manifold. The decoder is defined by a set of reverse layers parameterized by θ that compute a reconstruction of the input data, i.e. the image, from the code vector Z. In the training stage, the network finds the optimal weight matrices and bias vectors that minimize the mean square error between the input image and the reconstructed one by using:

$$J = ||x_i - g_\theta(f_\phi(x_i))||^2 \tag{1.11}$$

Many useful analyses can be conducted over the latent space. For instance, we could apply clustering on the latent code Z since it lies in low-dimensional space and thus increase the performance. Also, we can reduce the dimensionality of the data to help with visualization in order to gain some insight about its intrinsic structure. A popular example is using a latent space with size 2 or 3 to visualize the cluster differences in a 2D or 3D space respectively.

1.8 Sequence Models: Review

Over a half-century, statistical parametric methods provided the state-of-the-art performance for time-series modeling and prediction across different applications and fields [31]. [32] classify parametric methods into two main groups based on their mathematical complexity: exponential smoothing models [33], and Auto Regressive Integrated Moving Average(ARIMA) [34], [35]. In parametric "linear" methods, the future value can be predicted by a linear function of the past observations. Moving Average (MA) is one of the simplest parametric models and it is defined as an arithmetic mean of the previous time-series values with equally assigned weights over a specific time interval [36], [35]. Another variation of MA named Simple Exponential Smoothing

was developed by [37] where time-series values that are closer to the predicted value in time are multiplied by larger weights. However, this approach fails to capture the upward and downward trends and generates either understated or exaggerated predictions. Holt's Exponential Smoothing [33] overcomes this problem by introducing another hyperparameter to the Simple Exponential Smoothing model. However, the added hyperparameter for modeling the trend increases the search space to statistically learn future values. ARIMA models [38], [39] have been proposed to overcome the drawbacks of Autoregressive Moving Average (ARMA) models. ARMA models [40] are suitable for univariate stationary time-series modeling [35]. However, in many practical applications, time-series exhibit non-stationary behavior. Thus, ARIMA added the integration procedure to generalize ARMA to adapt to non-stationarity. In addition, power transformations are used on time-series upward and downward trends to make the series stationary. Another well-known parametric method is Seasonal ARIMA, also know as SARIMA. It was proposed by [36] as a successful variation of ARIMA to count for the seasonal periods and effects in the time-series. The aforementioned methods model only univariate time- series applications, thus, Vector Autoregression (VAR) was introduced as a linear framework to provide a systematic way to capture "rich dynamics" in multivariate time-series [41]. Parametric methods are simple and generally assume a known prior over the distribution of the time-series data. The advancement of machine learning brought sophisticated non-parametric methods for time-series modeling and prediction [42], [43].

Non-parametric methods predict the future observations as nonlinear functions of past ones without prior assumptions on the data distribution. Most recently, a successful combination of parametric and non-parametric methods enabled hybrid models to improve the prediction performance on a variety of time-series applications [44], [32]. Using machine learning methods to model time-series and predict future behavior can be formulated by dividing the dataset into two sets: training set, for model estimation, and testing set, for model performance evaluation. In supervised machine learning, the time-series is transferred into a dataframe with a sequence of observation and target values, where the target is the future value for each sequence of consecutive observations. The main assumption is that the time-series samples are independent and

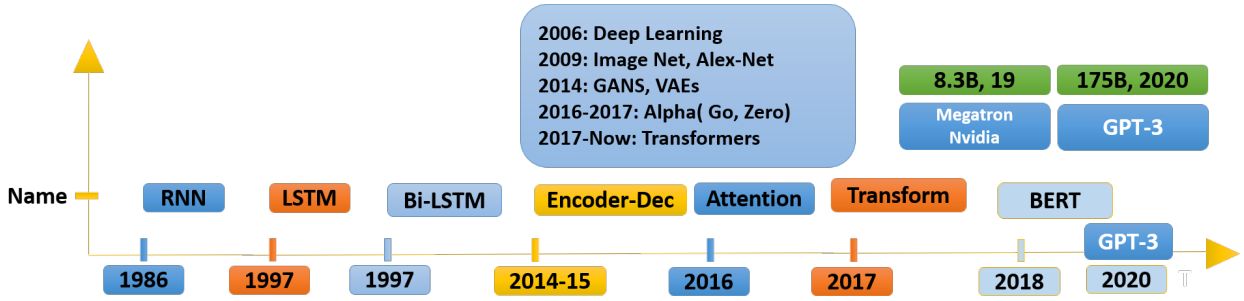

Figure 1.8 Historic depiction of deep learning breakthrough models.

identically distributed. However, time-series observations carry temporal information where each point in the time-series may impact the next which weakens this assumption. Among these algorithms that use this approach are Artificial Neural Network (ANN) [11], [29], [21] and SVM [45]. Neural Networks (NNs) consist of a stack of layers with multiple neurons or nodes in them. A simple NN consists of one input layer that receives information (i.e. features) and passes its output into the next hidden layer that applies a non-linear function on its weighted input. Rectified Linear Unit (ReLU), Sigmoid - Logistic, and Hyperbolic Tangent Function are well-known non-linear activation functions [46], [3], [15]. ANNs have attracted many researchers in the domain of time-series modeling and forecasting and were successfully applied into various time-series applications [47], [48]. Convolutional Neural Networks [6] have successfully being implemented across many domains including face recognition [49], image classification [50], and more recent time-series forecasting and classifications [51], [52], [53], [54], [55], [56]. Recurrent Neural Networks (RNNs) [2], [57] [58] and Long Short-Term Memory (LSTM) [59] and gated recurrent [60], encoder-decoder structures [61] [62], encoder-decoder based attention [63], transformers [64], and most recently variations of Bidirectional Transformers for Language Understanding (BERT) has been concretely confirmed to be the state of the art on eleven natural language processing applications [65].

LSTM is a novel variation of RNN to address the long-term dependency and the vanishing gradient problems [66], [67], [68] and exploding gradient [69]. LSTM has been applied in many univariate [70], [71], [72] and multivariate time-series applications [73], [74], [75] as a non-

parametric method. However, LSTM constructs its hidden states by integrating the past time steps. Bidirectional Long Short-Term Memory networks (Bi-LSTMs) [76] were proposed to integrate information from both past and future time steps by means of two hidden states. In all the above mentioned algorithms, one of the biggest challenges on how to properly represent time-series for learning tasks especially when time-series profiles are of variable length. RNN has proven to be a successful deep neural network architecture in mapping fixed source sequences into target sequences when the alignment is defined ahead of time [62]. A simple encoder-decoder approach to learning representations of sequences was developed by [61] for statistical machine translation. The RNN encoder encodes the input sequence into a fixed representation vector known as the encodings and use the decoder to map back the learned vector representation into the target sequence. [77] introduces a novel differentiable attention method to allow the recurrent neural to focus on different parts of the input space acts as a selective criteria of important information from different hidden states of the network. A variation of this idea was applied to neural machine translation by allowing the network to implement soft-search attention on different words of the input sentence that are relevant to to human intuition in performing language translation [63]. One-step forecasting models predict only the next time step of the time series. Recurrent neural networks has been used in predicting one-step-ahead by [77].And multi-step predictions can be implemented by sequence-to-sequence (seq2seq) models directly [62] [61]. Given an input variable length sequence, seq2seq can predict another sequence of future values for a fixed and variable range of horizons [78]. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks [79] probabilistic forecasts, based on training an auto-regressive recurrent network model on a large number of related time series.

While DeepAR only learns a univariate distribution, [80] combined RNN based-models with copulas to model multivariate probability distributions.

1.9 Deep Learning Frameworks

Below you can find some of the popular frameworks used in contemporary machine learning and the benchmark datasets on which researchers can compare their algorithms.

Table 1.3 Datasets specifications.

Name	Year	Organization	Source
Theano	2007	Uni Monteral	http://deeplearning.net/software/theano/
Caffe	2014	UC-Berkeley	http://caffe.berkeleyvision.org/
TensorFlow	2015	Google	https://www.tensorflow.org/
Keras	2015	FancoisChollet	https://keras.io/
MXNET	2015	Amazon	https://mxnet.apache.org/
PyTorch	2016	FaceBook	http://pytorch.org/
Caffe2	2017	FaceBook	https://caffe2.ai/

1.10 Computer Vision Benchmark Datasets

Table 1.4 Deep learning frameworks.

Name	Size(Pixels)	Number of classes	Source
CIFAR-10	(32,32,3)	10	https://www.cs.toronto.edu/ kriz/cifar.html
CIFAR-100	(32,32,3)	100	https://www.cs.toronto.edu/ kriz/cifar.html
MNIST	(28,28,1)	10	http://yann.lecun.com/exdb/mnist
Fashion-MNIST	(28,28,1)	10	http://arxiv.org/abs/1708.07747
ImageNet	(224,224,3)	1000	http://www.image-net.org/
SVHN	(32,32,3)	10	http://ufldl.stanford.edu/housenumbers/
NORB	3D images	5	http://www.cs.nyu.edu/ylclab/data/norb/
MIT-67	scenes	67	http://web.mit.edu/torralba/www/indoor.html
Pascal-VOC-07	20 objects	20	http://host.robots.ox.ac.uk/pascal/VOC/
Face recognition	20 objects	20	http://vis-www.cs.umass.edu/lfw/
Google-Images	9 million	600	https://github.com/openimages/dataset
Face-recognition	20 objects	20	http://vis-www.cs.umass.edu/lfw/

Chapter 2: An Overview of Time Series Approaches: Background

2.1 Note to Reader

Portions of this chapter have been previously published in our paper at the Journal of Food Engineering and have been reproduced with the permission from Journal of Food Engineering.

2.2 Motivation

Handling multivariate time-series data with variable lengths is one of the most elusive and challenges in machine learning applications for cold-chain. Pre-processing multivariate time-series with variable lengths has been studied for a wide range of applications including representation learning [[81], [82]], time-series prediction [[83], [71], [84]], clustering [[85], [86], [87]], classification [[88], [74], [89]], and anomaly detection [[90], [91], [92]], sequence modeling and machine translation [[93], [61]]. The challenges of time-series pre-processing with variable lengths were addressed by [94], and [95] for clustering analysis and by [96] for wireless sensor reduction using a theoretical framework. Our own research group has studied how artificial neural networks (ANN) can be applied to learn the mapping between sensors' locations in the cold-chain using time-series data [[97], [98], [99]]. However, these experiments were conducted on simulated temperature data, and do not adequately represent the real-world's complex non-linear environment being presented in this chapter. We believe this dataset can ultimately provide the food transportation and applied machine learning research communities with valuable data to inspire modeling innovations and IoT sensor applications for time series analysis.

2.3 Value of the Data

- With a combined decades worth of research in perishable post-harvest logistics, the authors believe that there is still a lot of unknowns when it comes to what actually happens in the cargo hold throughout the shipment. Another reason why this data is novel and important is that unlike many other studies which involved a singular entity, it is the result of a significant collaboration between all the stakeholders in the cold chain from growers to distributors to retailers to academics.

- Better understand the temperature profile of a standard cold-chain from harvest to store and how the temperature is distributed inside a perishable produce container during shipments of different durations.

- We hope that this data will motivate the food transportation research community to delve into developing more sophisticated regression and classification algorithms for univariate and multivariate time series, better clustering methods, learning representations with dimensionality reduction, and a better mathematical and statistical understanding of what is happening in cargo hold.

- The analysis and processing of temperature time-series data for predictive tasks represent a significant challenge especially when profiles may have variable lengths, high variability, and abnormalities as is common in many cold chain applications. The dataset will enable researchers from a wide array of fields and backgrounds to apply analytical tools such as machine learning and physical models in testing and comparing the performance of their predictive or diagnostic algorithms on the cold-chain.

- Develop temporal algorithms for solving regression, classification and clustering tasks on time series and deploy advanced data analytics to predict the future behaviour of the data profiles during transportation from harvest to the DC, or to identify if a sensor profile satisfied a specific quality control criteria for a retailer;

- Educational purposes which include analysis of univariate or multivariate time series data using statistical methods for regression, deep learning for time series classification, learning representation and location-based prediction; [1]

2.4 Specifications

- Subject: Agricultural Sciences
- Specific subject area: Food engineering and time series
- Type of data: Tabular data - CSV files
- Data format: Mixed (raw and preprocessed)
- How data were acquired: Figure 2.1 shows a US Map that highlights the shipping routes that were monitored to acquire the data. Two of the shipments originated from Plant City, Florida with final destinations in Florida and Georgia. Four shipments originated from Salinas, California with final destinations in Maryland, Pennsylvania, Virginia, South Carolina, North Carolina, Georgia and Texas.
- Instruments: DeltaTrak's Reusable Real−Time−Logger (RTL) Mini devices are used to log both temperature and location data in real time. The RTLs have a wide operational temperature range of $-30°$C to 95.55°C with a temperature accuracy of +/$-1°$C. More information about the hardware used in this study can be found in the appendix.

  Data was extracted via the cloud application which can establish secure communications with the GSM loggers. Python [101] was employed to perform subsequent data analysis.
- Parameters for data collection: The strawberry cold chain beginning at the field, the strawberries are harvested and placed into plastic clam shells packed into cardboard flats with 8 clamshells (each weighing one pound) per flat, which are subsequently stacked together to build pallets containing between 18−20 layers with 6 flats per layer. Once the pallets are

[1] Possible sensor and data applications, and several future research trajectories are summarized in [100].

built in the field on the back of a flatbed trailer, they are driven to the nearest processing facility to be precooled down to transportation and storage temperatures (0°C).

- Data source location

  Institution: University of South Florida and University of Florida

  Country:US

  States for collected samples/data: Florida, Georgia, Maryland, Pennsylvania, Virginia, South Carolina, North Carolina, Texas, California.

- Primary data sources: WishFarms.

- Data accessibility:

  Direct URL to data: https://data.mendeley.com/datasets/nxttkftnzk/draft?a=7d8b1fed-c1c3-4aa3-8cf3-5b385d221237Dataset link

2.5 Data Collection Process

2.5.1 General Description of the Dataset

The objective of the data collection process was to obtain the wide range of temperature profiles to which the strawberry shipments around the United States are subjected from the time of the harvest to the arrival at the distribution center. In total there were six shipments which cover both short and long−distance transportation scenarios. Two of the shipments originated from the strawberry capital of the East Coast, Plant City, Florida with final destinations both in Florida and in Georgia. Four shipments originated from Salinas, California, the most productive agricultural region in California sometimes referred to as the world's agriculture technology capital with final destinations in a wide range of states including Maryland, Pennsylvania, Virginia, South Carolina, North Carolina, Georgia and Texas. The process for data collection has been uniform throughout each experimental trial. Figure 2.1 highlights the aforementioned states.

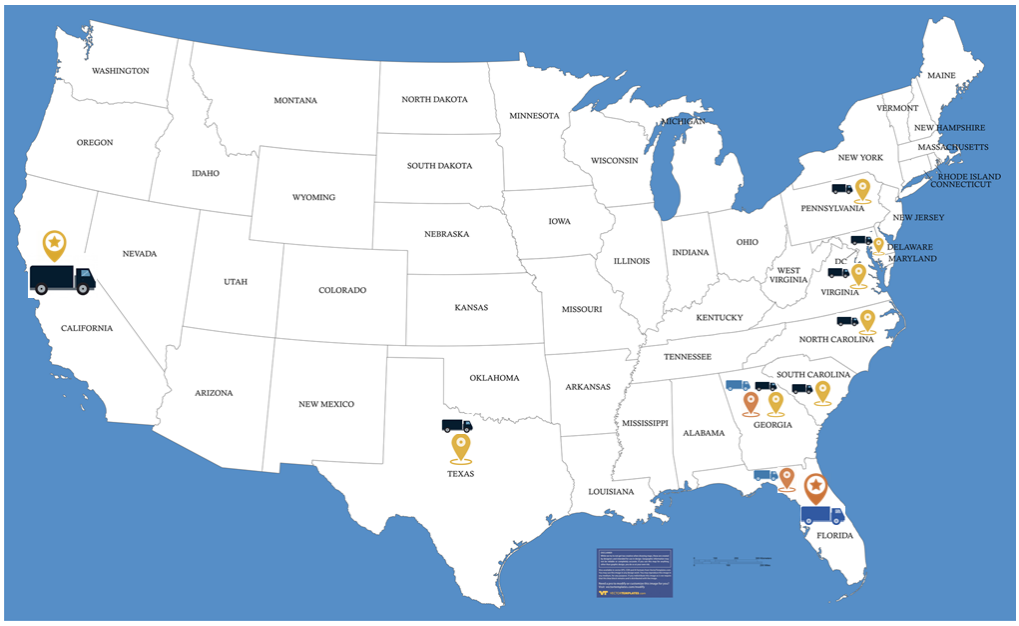

Figure 2.1 The shipping routes that were monitored in this study: Two of the shipments originated from Plant City, Florida (orange star) with final destinations both in Florida and in Georgia. Three shipments originated from Salinas, California (yellow star) with final destinations in a wide range of states including Maryland, Pennsylvania, Virginia, South Carolina, North Carolina, Georgia and Texas.

2.5.2 Temperature Sensors

We used DeltaTrak's Reusable Real−Time−Logger (RTL) Mini devices as shown in Figure 2.2 to log both temperature and location data in real time. The data is transmitted in real time via GSM cellular networks which eliminate the need to collect the loggers at the end of the shipment to be able to have access to the recorded data. The loggers have a wide temperature range of $-30°C$ to $95.55°C$ with a temperature accuracy of $+/-1°C$ in the range of interest for this experiment. Unlike previous studies in this area, this device also eliminates the need for any prior infrastructure setup to be able to automatically collect the data (such as readers for radio frequency identification (RFID) transponders). More information about the hardware used in this study can be found in the appendix.

Figure 2.2 DeltaTrak's Reusable Real−Time−Logger (RTL) Mini devices to log both temperature and location data in real time.

2.5.3 Data Collection Setup

The loggers were instrumented inside the pallets of strawberries right at the point and time of harvest during the pallet buildup stage. A total of three loggers were placed in a single pallet distributed equally along the vertical axis. Specifically one logger was placed closer to the bottom of the pallet (3rd layer from the bottom), another was placed closer to the middle layer of the pallet and a third was placed closer to the top layer of the pallet (3rd layer from the top) between the fruits. A total of three instrumented pallets were sent out with each of the six shipments. Similar to the placement of loggers within the pallet, the instrumented pallets inside the container were distributed equally along the horizontal axis. Specifically, one instrumented pallet was placed at the front of the container (i.e., close to the front of the truck), another was placed near the middle of the container and a third was placed at the back of the container (i.e., close to the loading end), each named accordingly (front, middle, rear) in the dataset. Hence, there were 9 loggers in total for each instrumented shipment labeled with respect to the loggers' location in the pallet and the pallets' location in the container (front−top, front−middle, front−bottom, ..., rear−bottom).

2.5.4 Cold-chain Background

Traditionally, in the strawberry cold chain beginning at the field, the strawberries are harvested and placed into plastic clam shells (the packaging the consumers are most familiar with)

packed into cardboard flats with 8 clamshells (each weighing one pound) per flat, which are subsequently stacked together to build pallets containing between 18−20 layers with 6 flats per layer. Once the pallets are built in the field on the back of a flatbed trailer, they are driven to the nearest processing facility to be precooled down to transportation and storage temperatures (0°C). This represents the most critical period in the proper temperature management regimen of strawberries due to the fact that even a single hour delay in precooling could result in days of shelf life loss which are not readily observable until the product is placed on the grocery display for sale. After the precooling is completed, the pallets are either i) shipped directly to the distribution centers (DC) with pending purchase orders from the retailers connected to the DCs or ii) stored at the processing facility if the instantaneous supply from the fields is greater than the accumulated demand from the DCs [[102]]. While the ideal transportation and storage temperatures are identical (0°C), more variations in the profiles are observed for transported pallets due to the fact that the loading doors open and close during the loading/unloading stages. It is difficult to pinpoint the exact time the pallet is received at the DC due to the fact that the storage temperatures at DC track the transportation temperatures closely – however the temperature variations are significant enough during the DC−to−store stage to be observable on the recorded temperatures. However, it is important to note that since the GPS locations come from cellular towers and that no specific guidance have been provided to the store employees in handling and disposal of the loggers received with the incoming strawberry pallets, the point of divergence in the temperature profiles of the loggers (which likely indicate when each pallet is sent to a different store) in each individual shipment also represents the last reliable data point in the profiling of field−to−precooling−to−DC chain of events. In other words, if this data is to be used for statistical studies of field−precooling−transport temperatures, one can disregard the time−temperature points following the easily identifiable point of divergence of logger profiles within the same shipment as shown in Figure 2.4. *It is important to note that this uncertainty led us to exclude these divergent profile subsections in both the visualization and the calculation of the preliminary statistics and box plots of the data below. However, the entirety of the data is still provided in the original dataset for those researchers who would like*

to delve deeper into the unique temperature characteristics of these "product potentially at (or on its way to) the store" time intervals.

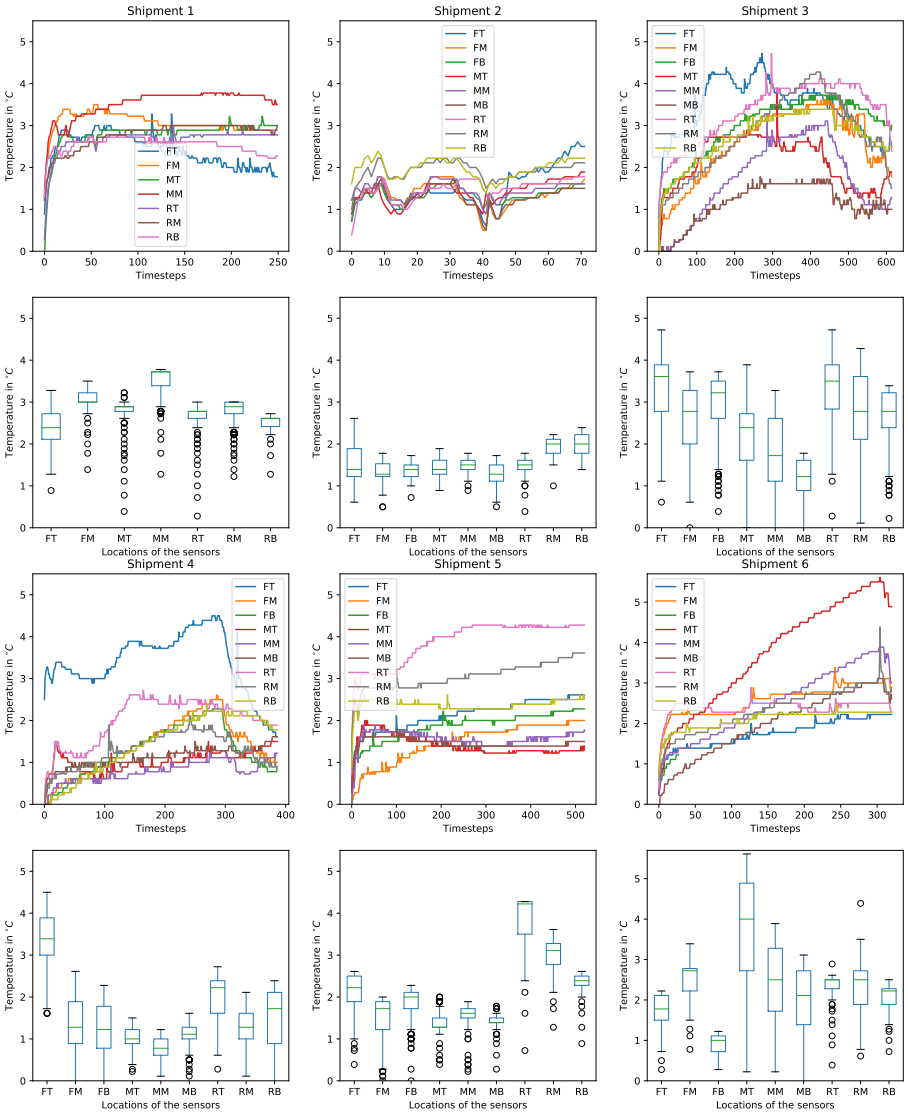

Figure 2.3 Temperature profiles of multivariate time series data and its distribution from precooling to the end of transportation.

2.6 Cold Chain Temperature Profiles as Multivariate Time Series

A time series $\mathscr{X}_t$ of size n can be defined as a collection of data points measured sequentially over equally spaced time intervals. i.e., $\mathscr{X}_t = (x_1, x_2, ..., x_n)$, where $x_t \in \mathbb{R}$ is an observation at time t. Time series signals can be classified with respect to their generative process into two main groups: deterministic, if there is a mathematical function f mapping the time series values to y; and stochastic if a random error occurs within the time series signal in addition to the mathematical function f. Generally a time series $\mathscr{X}_t$ is generated from an underlying stochastic process via a probability rule that governs the joint distribution of the random variables that generate that process [[34], [35]]. A time series can be univeriate if its observation is recorded over a single variable, and multivariate otherwise. time series can also be classified as continuous or discrete based on the interval of measurement. Specifically, a time series can be considered continuous if its observations are measured continuously over a specific time interval, whereas discrete time series would contain observations at equally and discretely spaced time intervals such as minutes, hours and days [[31], [32]]. time series are represented by graphs where the observations are plotted against the times of such observations. In total, this dataset includes 54 time series (with 9 sensor profiles for each of the 6 shipments) with a varying number of observations due to the length of the shipment and the sensor start/stop times. Figure 2.3 displays each sensor profile separately for each of the 6 shipments where top−left subfigure is the first and bottom−right subfigure is the last shipment. Please observe that these figures display real−world, noisy and complex multivariate time series as signature representatives of each shipment. Table 3.1 provides the time sampling rate, number of samples per day and the time duration for all shipments used in this study. The calculations including harvest, precooling, and transportation periods. We also demonstrate simple statistical distributions for each time series in Figure 2.3 ,where each subfigure represents the box−plots for the nine sensors in each shipment with second−row first subfigure and last−row last subfigure represent the first and last shipments respectively. The black dots in each figure represent the outliers for that particular distribution.

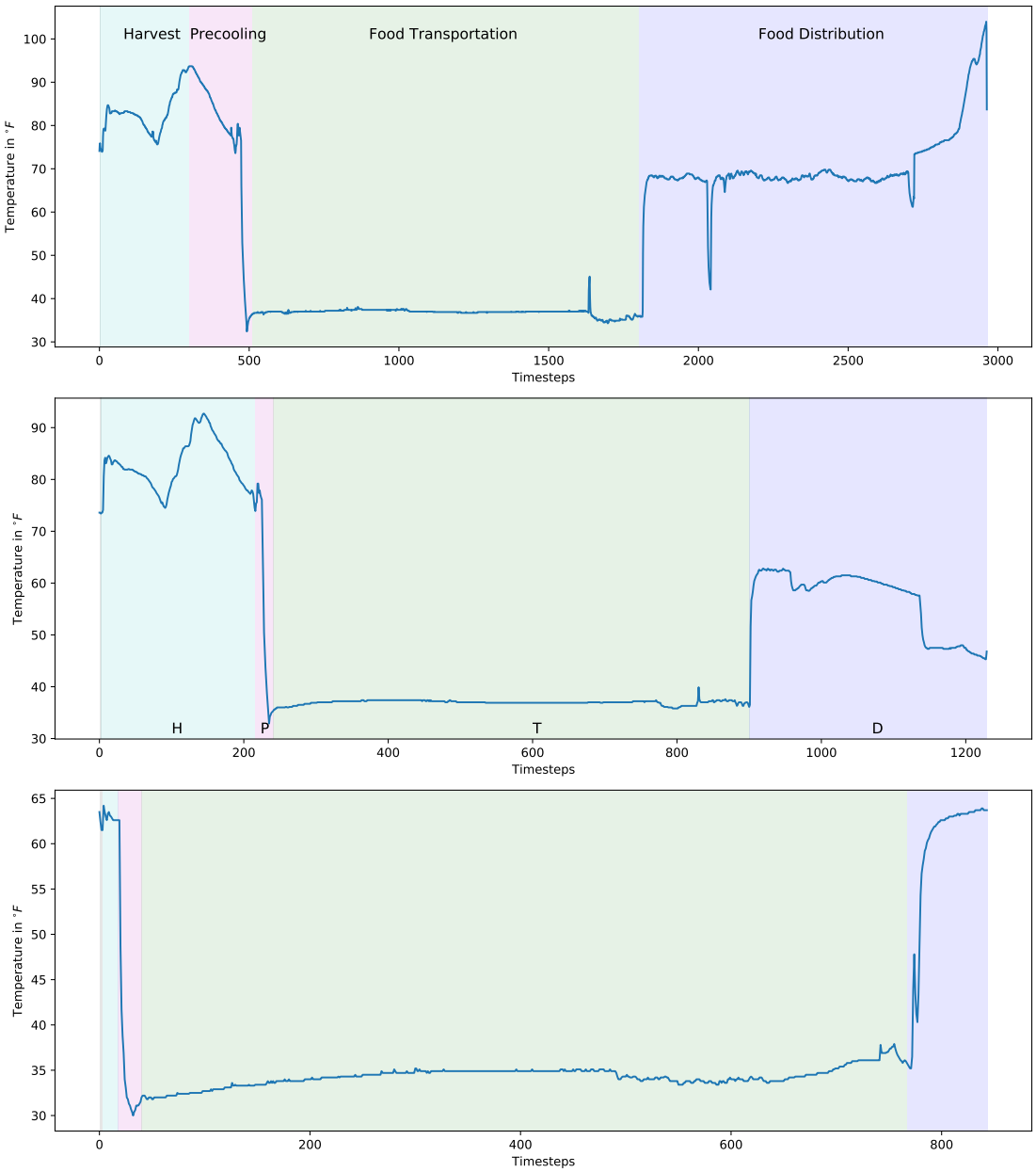

Figure 2.4 Full temperature profiles for three different sensors from three different shipments from harvest (field temperatures) to precooling to transportation to retail store display. The colors for harvest, precooling, transportation, and distribution are cyan, magenta, green, and blue, respectively.

Table 2.1 Time intervals for the temperature measurements for all sensor recordings used in the data collection.

Shipment Name	Variable/Sensor Name	Starting Time of Recording	Ending Time of Recording
	Front-Top	3/12/19 12:30 PM	4/2/19 12:24 PM
	Front-Middle	3/12/19 12:28 PM	3/22/19 10:19 AM
	Front-Bottom	NA	NA
	Middle-Top	3/12/19 12:30 PM	3/25/19 11:50 PM
Shipment 1	Middle-Middle	3/12/19 12:26 PM	3/22/19 9:27 AM
	Middle-Bottom	NA	NA
	Rear-Top	3/12/19 12:26 PM	4/3/19 8:42 AM
	Rear-Middle	3/12/19 12:29 PM	3/22/19 2:01 AM
	Rear-Bottom	3/12/19 12:29 PM	3/21/19 8:04 AM
	Front-Top	4/4/19 3:04 AM	4/4/19 10:10 PM
	Front-Middle	4/4/19 3:03 AM	4/22/19 7:06 AM
	Front-Bottom	4/4/19 3:07 AM	4/20/19 6:08 PM
	Middle-Top	4/4/19 3:07 AM	4/14/19 3:12 PM
Shipment 2	Middle-Middle	4/4/19 3:04 AM	4/4/19 10:07 PM
	Middle-Bottom	4/4/19 3:04 AM	4/22/19 9:42 AM
	Rear-Top	4/4/19 3:03 AM	4/11/19 7:03 PM
	Rear-Middle	4/4/19 3:07 AM	4/10/19 10:49 AM
	Rear-Bottom	4/4/19 3:07 AM	4/12/19 6:59 AM
	Front-Top	7/9/19 9:09 AM	7/17/19 6:00 PM
	Front-Middle	7/9/19 8:58 AM	7/16/19 3:45 PM
	Front-Bottom	7/9/19 8:59 AM	7/20/19 4:08 PM
	Middle-Top	7/9/19 8:56 AM	7/24/19 6:03 AM
Shipment 3	Middle-Middle	7/9/19 8:57 AM	7/24/19 9:04 AM
	Middle-Bottom	7/9/19 8:58 AM	7/15/19 6:56 PM
	Rear-Top	7/9/19 8:58 AM	7/22/19 3:09 AM
	Rear-Middle	7/9/19 8:57 AM	7/24/19 9:19 AM
	Rear-Bottom	7/9/19 8:57 AM	7/18/19 3:09 AM
	Front-Top	7/9/19 9:08 AM	7/24/19 9:15 AM
	Front-Middle	7/9/19 8:56 AM	7/24/19 8:59 AM
	Front-Bottom	7/9/19 8:57 AM	7/16/19 2:10 PM
	Middle-Top	7/9/19 9:10 AM	7/15/19 3:53 PM
Shipment 4	Middle-Middle	7/9/19 9:11 AM	7/19/19 5:10 AM
	Middle-Bottom	7/9/19 9:12 AM	7/16/19 9:23 AM
	Rear-Top	7/9/19 9:12 AM	7/24/19 9:19 AM
	Rear-Middle	7/9/19 9:13 AM	7/19/19 5:00 AM
	Rear-Bottom	7/9/19 9:13 AM	7/16/19 2:34 PM
	Front-Top	7/10/19 10:24 AM	7/17/19 5:19 PM
	Front-Middle	7/10/19 10:24 AM	7/24/19 3:10 PM
	Front-Bottom	7/10/19 10:24 AM	7/20/19 1:47 PM
	Middle-Top	7/10/19 10:19 AM	7/18/19 11:01 AM
Shipment 5	Middle-Middle	7/10/19 10:32 AM	7/19/19 7:07 AM
	Middle-Bottom	7/10/19 10:41 AM	7/18/19 11:35 AM
	Rear-Top	7/10/19 10:22 AM	7/18/19 4:31 PM
	Rear-Middle	7/10/19 10:33 AM	7/18/19 5:21 PM
	Rear-Bottom	7/10/19 10:22 AM	7/24/19 2:06 PM
	Front-Top	7/10/19 10:22 AM	7/26/19 5:03 AM
	Front-Middle	7/10/19 10:18 AM	7/25/19 10:19 AM
	Front-Bottom	7/10/19 10:39 AM	7/10/19 8:32 PM
	Middle-Top	7/10/19 10:25 AM	7/18/19 9:51 AM
Shipment 6	Middle-Middle	7/10/19 10:25 AM	7/15/19 6:29 AM
	Middle-Bottom	7/10/19 10:25 AM	7/16/19 11:06 AM
	Rear-Top	7/10/19 10:26 AM	7/17/19 7:58 PM
	Rear-Middle	7/10/19 10:27 AM	7/17/19 5:55 PM
	Rear-Bottom	7/10/19 10:27 AM	7/23/19 4:43 AM

Table 2.2 All sensors across all shipments summary statistics – integer and numeric variables.

Sensor Name	Ship No	Timestamps	Mean	STD	Min	25%	50%	75%	Max
	Ship 1	250	2.40	0.39	0.89	2.11	2.39	2.72	3.28
	Ship 2	72	1.51	0.48	0.61	1.22	1.39	1.89	2.61
Front Top	Ship 3	615	3.43	0.67	0.61	2.78	3.61	3.89	4.72
	Ship 4	521	2.13	0.36	0.39	1.89	2.22	2.50	2.61
	Ship 5	521	2.13	0.36	0.39	1.89	2.22	2.50	2.61
	Ship 6	321	1.75	0.35	0.28	1.50	1.78	2.11	2.22
	Ship 1	250	3.07	0.25	1.39	3.00	3.00	3.22	3.50
	Ship 2	72	1.35	0.30	0.50	1.22	1.28	1.53	1.78
Front Middle	Ship 3	615	2.56	0.83	-0.39	2.00	2.78	3.28	3.72
	Ship 4	521	1.47	0.47	-0.11	1.22	1.72	1.89	2.00
	Ship 5	521	1.47	0.47	-0.11	1.22	1.72	1.89	2.00
	Ship 6	321	2.58	0.37	0.78	2.22	2.72	2.78	3.39
	Ship 1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Ship 2	72	1.35	0.19	0.72	1.22	1.39	1.50	1.72
Front Bottom	Ship 3	615	2.99	0.70	0.39	2.61	3.22	3.50	3.72
	Ship 4	521	1.87	0.33	0.00	1.72	2.00	2.11	2.28
	Ship 5	521	1.87	0.33	0.00	1.72	2.00	2.11	2.28
	Ship 6	28	0.93	0.26	0.28	0.72	1.00	1.11	1.22
	Ship 1	250	2.79	0.37	-0.22	2.78	2.89	2.89	3.22
	Ship 2	72	1.43	0.26	0.89	1.28	1.39	1.61	1.89
Middle Top	Ship 3	615	2.13	0.60	-0.39	1.61	2.39	2.72	3.89
	Ship 4	521	1.40	0.23	0.39	1.28	1.28	1.50	2.00
	Ship 5	521	1.40	0.23	0.39	1.28	1.28	1.50	2.00
	Ship 6	321	3.74	1.31	0.22	2.72	4.00	4.89	5.61
	Ship 1	250	3.51	0.36	1.28	3.39	3.72	3.72	3.78
	Ship 2	72	1.46	0.20	0.89	1.39	1.50	1.61	1.78
Middle Middle	Ship 3	615	1.72	0.93	-0.89	1.11	1.72	2.61	3.28
	Ship 4	521	1.57	0.20	-0.11	1.50	1.61	1.72	1.89
	Ship 5	521	1.57	0.20	-0.11	1.50	1.61	1.72	1.89
	Ship 6	321	2.50	0.86	0.22	1.72	2.50	3.28	3.89
	Ship 1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Ship 2	72	1.29	0.25	0.50	1.11	1.28	1.50	1.72
Middle Bottom	Ship 3	615	1.13	0.50	-0.89	0.89	1.22	1.61	1.78
	Ship 4	521	1.45	0.12	0.28	1.39	1.39	1.50	1.78
	Ship 5	521	1.45	0.12	0.28	1.39	1.39	1.50	1.78
	Ship 6	321	1.97	0.78	-0.22	1.39	2.11	2.72	3.11
	Ship 1	250	2.65	0.30	0.28	2.61	2.78	2.78	3.00
	Ship 2	72	1.48	0.25	0.39	1.39	1.50	1.61	1.78
Rear Top	Ship 3	615	3.34	0.69	0.28	2.83	3.50	3.89	4.72
	Ship 4	521	3.88	0.51	0.72	3.50	4.22	4.28	4.28
	Ship 5	521	3.88	0.51	0.72	3.50	4.22	4.28	4.28
	Ship 6	321	2.39	0.22	0.39	2.28	2.50	2.50	2.89
	Ship 1	250	2.77	0.31	1.22	2.72	2.89	3.00	3.00
	Ship 2	72	1.93	0.21	1.00	1.78	2.00	2.11	2.22
Rear Middle	Ship 3	615	2.79	0.91	0.11	2.11	2.78	3.61	4.28
	Ship 4	521	3.05	0.31	1.28	2.78	3.11	3.28	3.61
	Ship 5	521	3.05	0.31	1.28	2.78	3.11	3.28	3.61
	Ship 6	321	2.34	0.54	0.61	1.89	2.50	2.72	4.39
	Ship 1	250	2.54	0.18	1.28	2.42	2.61	2.61	2.72
	Ship 2	72	1.98	0.23	1.39	1.78	2.00	2.22	2.39
Rear Bottom	Ship 3	615	2.71	0.60	-0.22	2.39	2.78	3.22	3.39
	Ship 4	521	2.37	0.13	0.89	2.28	2.39	2.50	2.61
	Ship 5	521	2.37	0.13	0.89	2.28	2.39	2.50	2.61
	Ship 6	321	2.09	0.23	0.72	1.89	2.22	2.28	2.50

Table 2.3 Variables descriptions.

Variable	Type	Description	Source of Data
Front Top	Numeric	Sensor placed at the front pallet in the shipping container instrumented closer to the top layer of the pallet (3rd layer from the top) between the fruits.	Shipments 1 through 6
Front Middle	Numeric	Sensor placed at the front pallet in the shipping container instrumented closer to the middle layer of the pallet between the fruits.	Shipments 1 through 6
Front Bottom	Numeric	Sensor placed at the front pallet in the shipping container instrumented closer to the bottom of the pallet (3rd layer from the bottom) between the fruits.	Shipments 2 through 6
Middle Top	Numeric	Sensor placed at the middle pallet in the shipping container instrumented closer to the top layer of the pallet (3rd layer from the top) between the fruits.	Shipments 1 through 6
Middle Middle	Numeric	Sensor placed at the middle pallet in the shipping container instrumented closer to the middle layer of the pallet between the fruits	Shipments 1 through 6
Middle Bottom	Numeric	Sensor placed at the middle pallet in the shipping container instrumented closer to the bottom of the pallet (3rd layer from the bottom) between the fruits.	Shipments 2 through 6
Rear Top	Numeric	Sensor placed at the rear pallet in the shipping container instrumented closer to the top layer of the pallet (3rd layer from the top) between the fruits.	Shipments 1 through 6
Rear Middle	Numeric	Sensor placed at the rear pallet in the shipping container instrumented closer to the middle layer of the pallet between the fruits.	Shipments 1 through 6
Rear Bottom	Numeric	Sensor placed at the rear pallet in the shipping container instrumented closer to the bottom of the pallet (3rd layer from the bottom) between the fruits.	Shipments 1 through 6

Table 2.4 Time sampling rate and the time duration for all shipments used in this study.

Shipments	Approximate Sampling Rate (in Minutes)	Number of Samples (per Day)	Total Duration
Shipment 1	5	264	6 days, 9 hours and 28 minutes
Shipment 2	10	127	2 days, 1 hour and 9 minutes.
Shipment 3	10	132	6 days, 9 hours and 25 minutes
Shipment 4	10	132	5 days, 12 hours and 5 minutes
Shipment 5	10	130	6 days, 14 hours and 53 minutes
Shipment 6	10	132	4 days, 4 hours and 35 minutes

Table 2.5 Timestamp counts of temperature sensor profiles (raw data count).

Sensors/Count	Ship1	Ship2	Ship3	Ship4	Ship5	Ship6
Front-Top	3208	100	1094	2019	925	1995
Front-Middle	2663	2054	952	2027	1882	2054
Front-Bottom	N/A	2054	1490	922	1276	54
Middle-Top	3599	1320	1952	811	1036	1003
Middle-Middle	1300	100	1960	1316	1145	605
Middle-Bottom	N/A	2054	844	916	1012	781
Rear-Top	2964	1008	1666	2049	1041	995
Rear-Middle	1230	827	1999	1287	1047	974
Rear-Bottom	1118	1105	1173	930	1885	1721

2.7 Statistical Description and Visualization of the Dataset

Table 2.2 show the number of observations for each sensor in each shipment for the period we are interested in (from precooling to the end of transportation), and Tables 2.2 through 2.5 represent simple numerical characteristics and statistics of the dataset. Table 2.5 shows the total number of observations for each sensor in each shipment in its raw shape. Please note that two sensors in the first shipment malfunctioned and did not report any data to the cell towers. Please also note that one sensor did not fully report its data on the last shipment. As a result, a researcher looking for a full representation of harvest−to−DC profiles for different locations and shipment are encouraged to use the remaining 51 profiles in their analysis.Table Table 2.2 provides a complete statistical description fro each sensor profile.

2.8 Time Series Performance Metrics

To evaluate the similarity between two time series x_t^i and y_t^i, different metrics have been proposed in the literature. To name few:

- The Mean Forecast Error (MFE): On average, it measure how predicted values deviate from the real value.

$$MFE(x_t, y_t) = \frac{1}{N_s} \sum_{i=1}^{N_s} |x_t^i - y_t^i| \tag{2.1}$$

- The Mean Absolute Percentage Error (MAPE): MAPE is a percentage representation of mean absolute error.

$$MAPE(x_t, y_t) = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{|x_t^i - y_t^i|}{x_t^i} \tag{2.2}$$

- The Mean Squared Error (MSE):

  MSE, also known as the L2-norm, is a well-known performance metric due to its clear physical interpretation and the fact that it can be easily computed by taking the average

of the squared differences between time series x_t and y_t is defined by:

$$MSE(x_t, y_t) = \frac{1}{N_s} \sum_{i=1}^{N_s} (x_t^i - y_t^i)^2 \tag{2.3}$$

- The Root Mean Squared Error (RMSE):RMSE is nothing but the square root of MSE.

$$RMSE(x_t, y_t) = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} (x_t^i - y_t^i)^2} \tag{2.4}$$

- The Sum of Squared Error (SSE): On total, it measures the squared deviation of predicted values from real values.

$$SSE(x_t, y_t) = \sum_{i=1}^{N_s} (x_t^i - y_t^i)^2 \tag{2.5}$$

- The Normalized Mean Squared Error (NMSE): It uses the variance of the test set to normalize the MSE.

$$NMSE(x_t, y_t) = \frac{1}{N_s * \sigma^2} \sum_{i=1}^{N_s} (x_t^i - y_t^i)^2 \tag{2.6}$$

- The Mean Absolute Error (MAE): On average, how the absolute predicted values deviate from the absolute real values.

$$MAPE(x_t, y_t) = \frac{1}{N_s} \sum_{i=1}^{N_s} (x_t^i - y_t^i) \tag{2.7}$$

For the aforementioned equations, we refer to the observation i at time step t in time series x with x_t^i.

- Bayesian Criterion Information (BIC) [103]: BIC is best for explanation as it is allows consistent estimation of the underlying data generating process.

$$BIC(M) = 2logL(M) + p(M) * log(n) \tag{2.8}$$

n: number of observations.

- Akaike information criterion (AIC) [104]: Maximizing the likelihood function for each individual model, and choose the model for which AIC the smallest.

$$AIC(M) = 2\log L(M) + 2 * p(M) \tag{2.9}$$

p: dimension of the model.

Chapter 3: Time Series Analysis

3.1 Note to Reader

Portions of this chapter have been previously published in our paper at the Journal of Food Engineering and have been reproduced with the permission from Journal of Food Engineering.

3.2 Motivation

In this chapter, we present our effort in conducting a comprehensive statistical and temporal analysis to analyze the dynamic factors for the temporal heterogeneity, complexity, similarity, and discrepancy on a novel location aware multivariate time series dataset that was introduced in the previous chapter. The dataset represents the temperature variations across 6 different shipments with 9 sensors in each shipment monitoring and recording sensory data at 15−minute intervals across multiple days. Each container had three instrumented pallets where each pallet had a temperature sensor placed near the top, middle and the bottom of the pallet for a total of nine sensors in each shipment. The goal of this paper is to provide the descriptive statistics of the novel dataset with some time series visualization, and time-series statistical and temporal analysis.

3.3 Data Analysis

Our analysis is conducted to answer four main research questions:

1. How can one measure the similarities and discrepancies among the different multivariate time-series temperature profiles?

2. How can one measure the reading consistency between the temperature profiles both within the same container and across different containers?

3. How is the original temperature profile represented in the frequency domain and how is its reconstruction affected by the level of compression?

4. How can one measure the modeling complexity for forecasting using the autocorrelation of each sensor and its lagged profiles?

5. Based on the collected multivariate time-series, what are the possible applications for data analysis, food engineering and time-series?

These questions are answered below in their respective subsubsections following the same numbering scheme.

3.3.1 Differences and Similarities Based on Pearson's Correlation

To answer the first question, we use heatmaps of Pearson product−moment correlation coefficients to measure the similarities and discrepancies among shipments. Pearson's correlation coefficient can be written according to [105] as:

$$\sigma_m = \frac{\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)}{\sqrt{\sum_{i=1}^{n_x}(x_i-\mu_x)^2}\sqrt{\sum_{i=1}^{n_y}(y_i-\mu_y)^2}} \tag{3.1}$$

where μ_x and μ_y are the average means of the time-series signals at length n_x and n_y ,respectively. . Both can be written as:

$$\mu_x = \frac{1}{n_x}\sum_{i=1}^{n_x}(x_i); \qquad \mu_y = \frac{1}{n_y}\sum_{i=1}^{n_y}(y_i) \tag{3.2}$$

When analyzing shipments instrumented with multiple environmental sensors placed in different locations it is important to look at the similarities and differences between different sensor locations inside the same container. Pearson's correlation coefficient provides a robust way to summarily visualize the temporal similarities which may exist. For example, a high correlation coefficient between two different locations may indicate that a single sensor could be sufficient to

represent both temperature recordings whereas low correlation coefficients across the board may indicate significant differences in temperature distributions inside the container.

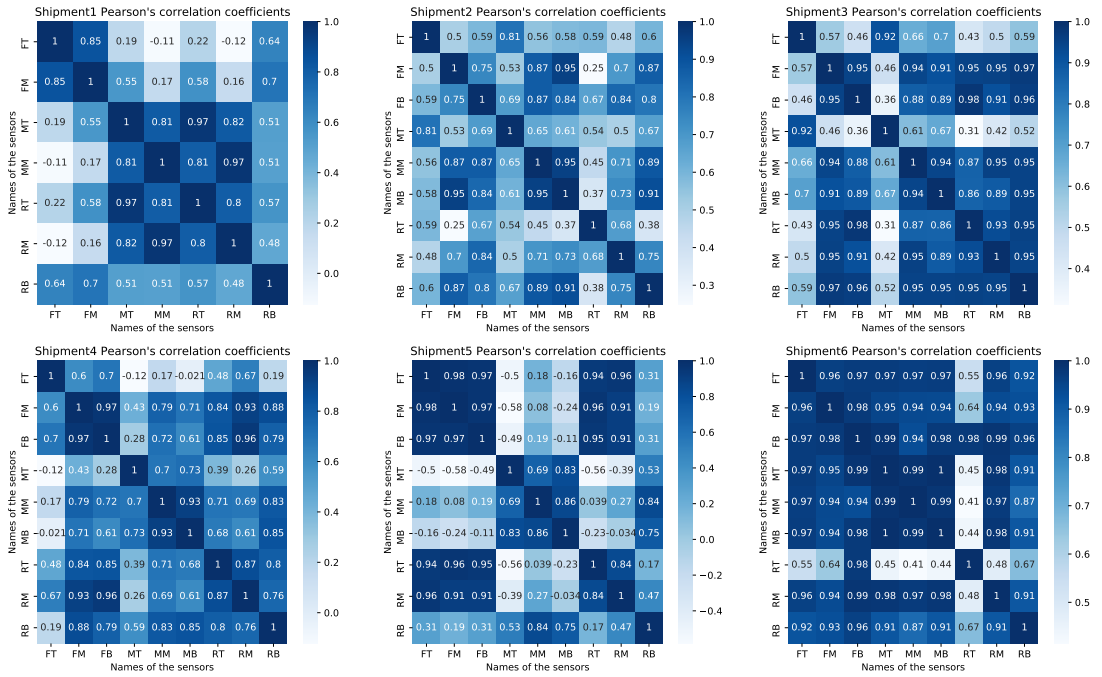

Figure 3.1 Multivariate time series “full profile” Pearson’s correlations.

Figure 3.1 shows six heatmaps of Pearson’s correlation coefficients between the multivariate time-series profiles for each of the six shipments equipped with nine sensors each labeled by two letters where the first letter represents the instrumented pallet location inside the container and the second represents the sensor position inside that pallet. For example, FT (Front−Top) represents the location of the top sensor inside the front pallet. The heatmaps reveal the heterogeneity and complexity of the collected multivariate time-series by demonstrating clear difference in the gradient variations across different sensors both within the same shipment and among different shipments as well. The gradient levels are driven by the strength of the correlations between the temperature sensors profiles. The profiles include a full representation of the cold chain from measuring harvest temperatures to the precooling process, and the subsequent transportation and

distribution ending at the retail stores. Based on the inherent randomness of each of these processes, the temperature levels can change significantly to create an overall underlying process that is both stochastic and nonstationary with moments changing over time . One can observe that the shipments 2,3, and 6 have more consistent readings of strawberry temperatures among different sensors that are reflected by stronger correlations in comparison to shipments 1,4, and 5. Shipment 6 has the highest positive correlations values while shipment 5 has the lowest. This information can be helpful in location−based predictions for wireless sensor networks and data analytics applications, where physical sensors can be removed, and their readings predicted using other sensors. However, a careful examination is needed in identifying the best candidates for such forecasting applications. For example, the heatmap for shipment 6 shows that the FT and RB sensors have a strong correlation coefficient of 0.92 which suggests a high probability of successful prediction of one location's temperature profile from the other's, however the same locations have a correlation coefficient of only 0.31 for shipment 5 which indicates the complexity and inconsistency of time-series forecasting and predictive analytics in such a scenario.

3.3.2 Dynamic Time Warping (DTW) Distance Analysis

To answer the second question, we have conducted an extensive analysis on the time-series temperature profiles using dynamic time warping (DTW) distances according to [106] as a measure of comparing one time-series profile to another. DTW warps the time axis of one (or both) temperature profile sequences to achieve a better alignment.

To align two sequences, DTW constructs an $n_x \times n_y$ matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $d(x_i, y_j)$ between the two temperature points x_i and y_j (Typically the Euclidean distance is used, so $d(x_i, y_j) = \sqrt{\sum(x_i - y_j)^2}$). Each matrix element (i, j) corresponds to the alignment between the temperature points x_i and y_j. DTW warping path is subject to three constraints according to [105]: boundary conditions, continuity and monotonicity; to minimize the overall warping cost, it can be written as following:

$$DTW(x,y) = min\left\{\frac{\sqrt{\sum_{l=1}^{L}(z_i)}}{L}\right\} \tag{3.3}$$

where L, the number of elements in the warping path, is used to compensate for the fact that warping paths may have different lengths. In order to find the minimum path, the warping path Z is contiguous: $Z = z_1, z_2, \ldots, z_L$ and $max(n_x, n_y) \leqslant L < (n_x + n_y\text{-}1)$. DTW use dynamic programming to compute the cumulative distance $\zeta(i,j)$ from $d(i,j)$ "current position" in the matrix and the minimum of the cumulative distances of the adjacent elements:

$$\zeta(i,j) = d(x_i, y_j) + min\{\zeta(i-1, j-1) + \zeta(i-1, j) + \zeta(i, j-1)\} \tag{3.4}$$

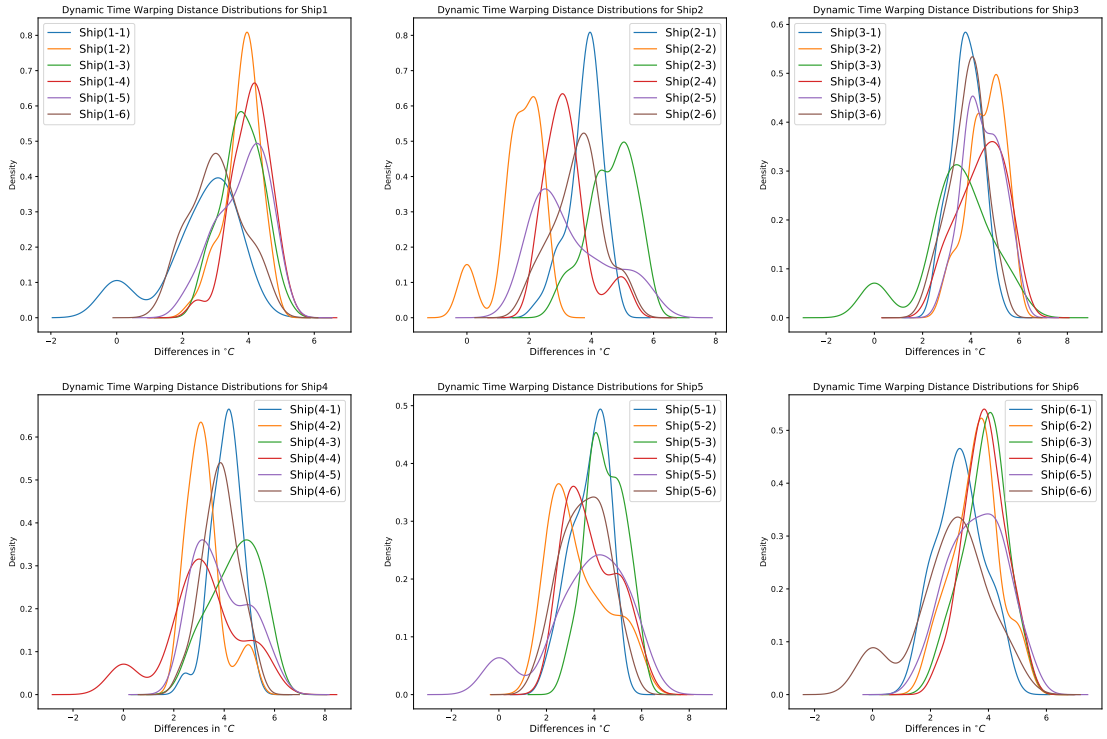

Figure 3.2 Dynamic Time Warping (DTW) analysis of the "full temperature profiles" from the multivariate time series across all shipment pairs.

To demonstrate the true complexity and heterogeneity of the temporal dynamics for the sensor recordings, Figure 3.2 displays the dynamic time warping (DTW) distances between different sensor measurements among all the shipments for a total of 36 distributions. In total we have computed 2916 DTW distances (36 x 81) which correspond to 36 pair combinations among the six shipments (6 x 6) and 81 pair combinations among the nine sensors in each shipment (9 x 9). Every distribution in Figure 3.2 represents a shipment level DTW distance values of 81 distance−pairs between each sensor location and the remaining eight sensors' locations inside the container/shipment. The individual distributions are divided into two parts as the 6 homogenous distributions (i.e. within the same shipment for different sensor locations), and the 30 which represent the DTW distance−pairs across different shipments. *Thus, it would be impractical to show all the 2916 values including the zeros, and instead we show the distributions to indicate how close or far some sensors are in terms of their temperature readings*. Narrow distributions indicate similar behaviours and reading consistency whereas wider distributions are indicative of different behaviors and reading inconsistency. Narrower distributions indicate stronger temporal alignments or more similarities between sensors and represent a good illustration of the reading consistency of strawberry temperatures among different sensors inside the shipping containers. Conversely, wider distributions indicate higher levels of misalignment in terms of the profile behavior. We believe that the differences in DTW distance distributions, such as the means, standard deviations, and the skewness directions can jointly be interpreted as challenging indicators for the temporal heterogeneity, complexity, similarity, and discrepancy of the collected multivariate time-series. These distance−based distributions can be helpful in location−based predictions for wireless sensor networks and data analytics applications as the selection, identification, and grouping of the best candidates are necessary preprocessing steps for increasing the accuracy of time-series forecasting, clustering, and classification applications. Besides, the distributions of DTW distances can be used as strong and weak evidences of similarity and discrepancy for time-series analysis. For example, the DTW distance distribution between the shipments 5 and 2 are wider compared to the one computed for shipments 4 and 2, which indicates a strong evidence of temporal similarity

between shipments 4 and 2 and vice versa for shipments 5 and 2. Finally, this analysis can further help in aggregating similar shipments to help reduce the redundancy of selecting similar time-series profiles for representation which would ultimately reduce the time and memory complexity of the time-series analysis as well.

3.3.3 Frequency Domain Analysis of the Temperature Profiles

To answer the third question, we use Fast Fourier Transform (FFT) and its inverse IFFT to determine FFT coefficients and energy concentration in the frequency domain. A Fast Fourier transform (FFT) was introduced by [107] as a numerical computing algorithm to transform time or space domain signal or sequence to frequency domain representation. This representation is a spectrum of frequencies' coefficients that can be used to reconstruct the original signal by applying the Inverse Fast Fourier Transform (IFFT). Discrete Fourier Transform (DFT) is discrete and can be computed according to [[107], [108]]by:

$$\mathscr{F}\{x[m]\} = \sum_{k=0}^{N-1} x[k] e^{\frac{-2\pi i k m}{N}} \tag{3.5}$$

where $m = 0, 1, ..., N-1$, N is the sequence length or number of time-series observations, k/N is the frequency. $w = e^{\frac{-2\pi i}{N}}$, and $w_N^{N_n+K} = w_N^k$.

The computed values F{x[m]} are complex, as they computed by the sum of complex exponential.

Analyzing the frequency domain spectrum of the multivariate time-series data is important for database query especially if the input sequence needs to be matched to one of many time-series sequences in the database [109]. Fast Fourier Transform (FFT) is a fast computing version of Discrete Fourier Transform (DFT) to decompose the time-series into multiple frequency components. Each frequency response is represented by one coefficient of DFT. The time-series profiles in the Figure 3.3 represents the original Middle−Middle sensor profile from shipment 5 and its reconstruction using different frequency coefficients. We observe for the time-series profiles, one can reconstruct the original profile from the frequency domain with a minimum number of coefficients

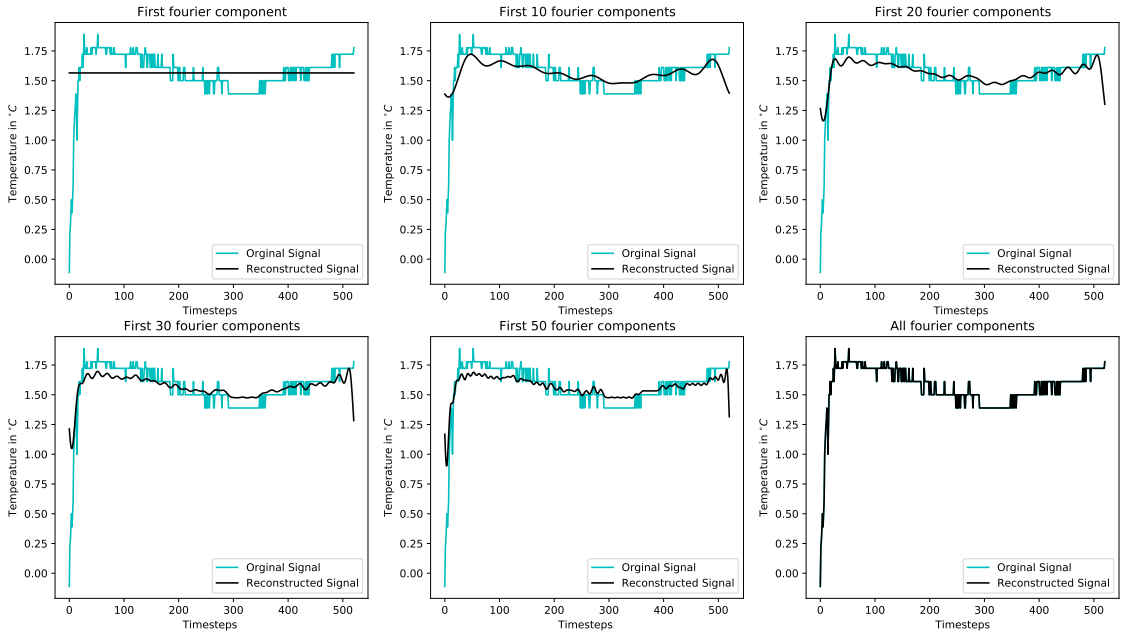

Figure 3.3 Fast Fourier Transform (FFT) and inverse (IFFT) temperature profile reconstructions for Middle−Middle sensor from shipment 5.

as shown by Figure 3.3 as the energy is concentrated around zero for the magnitude of the complex coefficients. Besides, this is an indication that storing and retrieving a large number of shipment profiles for time-series analysis is possible using a few coefficients of the FFT, thus decreasing the time−complexity of searching. However, there is an application trade-off between a fast querying process at the expense of losing some reconstruction resolutions of the original profile.

3.3.4 Autocorrelations of Individual Sensor Profiles

To answer the fourth question, we calculate the autocorrelation coefficients for each sensor across the six shipments and provide the results in figure 3.4 which plots these coefficients for time lags between 0 and 20. The subfigures for each shipment clearly illustrates the difficulty of the multivariate time-series data in forecasting applications specifically for the food supply chain. For instance, shipments 2 and 3 demonstrate different autocorrelation coefficients for each sensor (decaying or increasing in magnitude and changing in direction) as the number of lags

increases. The lags of the autocorrelation function and the partial autocorrelation function help determine the order of p and q in the ARIMA model where p is the autoregressive model order and q is the moving average model order. All of the analysis that we have conducted on this novel and temporally rich data has highlighted the main challenges of working with temperature profiles along the cold chain in terms of modeling, statistical analysis and IoT sensor data analytics specifically for food engineering. As a result, we will dedicate the next subsection to discuss the possible applications this work can inspire.

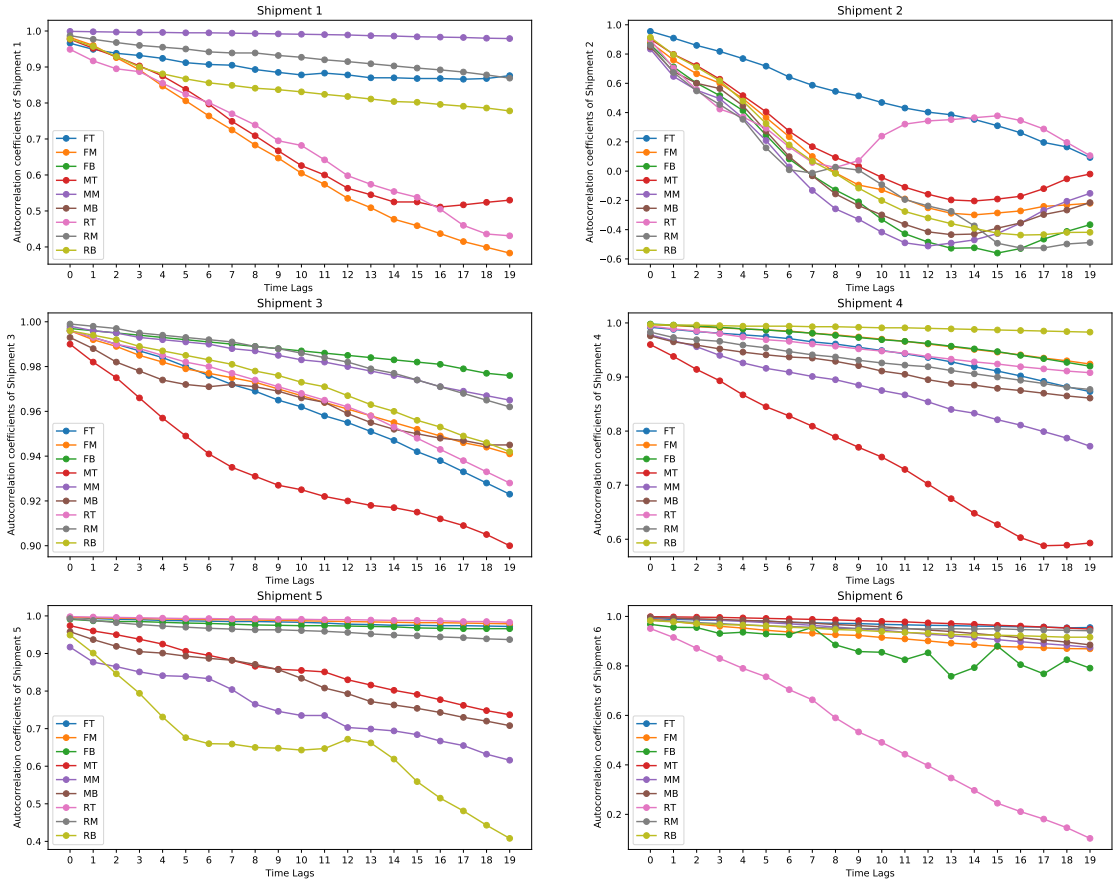

Figure 3.4 Autocorrelation coefficients for all sensors in shipments 1 through 6 for different lags.

Table 3.1 Autocorrelation coefficients for the sensors in shipment 1 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.966	0.982	N/A	0.977	0.999	N/A	0.949	0.987	0.978
t−2	0.949	0.96	N/A	0.952	0.998	N/A	0.917	0.977	0.957
t−3	0.938	0.926	N/A	0.929	0.997	N/A	0.895	0.968	0.929
t−4	0.932	0.892	N/A	0.903	0.996	N/A	0.887	0.96	0.9
t−5	0.924	0.847	N/A	0.875	0.996	N/A	0.855	0.955	0.881
t−6	0.912	0.806	N/A	0.838	0.995	N/A	0.824	0.95	0.867
t−7	0.907	0.764	N/A	0.797	0.995	N/A	0.801	0.942	0.856
t−8	0.905	0.725	N/A	0.749	0.994	N/A	0.77	0.939	0.849
t−9	0.893	0.683	N/A	0.709	0.993	N/A	0.739	0.939	0.841
t−10	0.885	0.647	N/A	0.667	0.992	N/A	0.695	0.932	0.837
t−11	0.878	0.605	N/A	0.626	0.991	N/A	0.682	0.927	0.831
t−12	0.883	0.574	N/A	0.6	0.99	N/A	0.642	0.92	0.824
t−13	0.878	0.535	N/A	0.563	0.989	N/A	0.598	0.915	0.818
t−14	0.87	0.509	N/A	0.545	0.987	N/A	0.574	0.909	0.811
t−15	0.87	0.477	N/A	0.525	0.986	N/A	0.554	0.903	0.804
t−16	0.868	0.459	N/A	0.525	0.984	N/A	0.538	0.897	0.802
t−17	0.868	0.437	N/A	0.511	0.983	N/A	0.505	0.892	0.796
t−18	0.866	0.415	N/A	0.517	0.982	N/A	0.46	0.886	0.791
t−19	0.868	0.399	N/A	0.524	0.98	N/A	0.436	0.878	0.786
t−20	0.876	0.383	N/A	0.53	0.979	N/A	0.431	0.869	0.778

Table 3.2 Autocorrelation coefficients for the sensors in shipment 2 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.955	0.88	0.865	0.903	0.834	0.849	0.898	0.861	0.914
t−2	0.909	0.759	0.712	0.798	0.647	0.68	0.704	0.674	0.796
t−3	0.859	0.665	0.601	0.721	0.553	0.6	0.557	0.548	0.711
t−4	0.818	0.601	0.516	0.628	0.495	0.564	0.424	0.455	0.618
t−5	0.769	0.495	0.414	0.516	0.363	0.447	0.37	0.354	0.479
t−6	0.717	0.366	0.246	0.405	0.209	0.273	0.293	0.159	0.325
t−7	0.643	0.235	0.083	0.273	0.029	0.099	0.165	0.009	0.179
t−8	0.587	0.1	−0.03	0.167	−0.132	−0.032	0.059	−0.013	0.067
t−9	0.545	−0.012	−0.129	0.093	−0.258	−0.156	0.025	0.027	−0.016
t−10	0.514	−0.095	−0.21	0.033	−0.329	−0.236	0.072	0.007	−0.116
t−11	0.469	−0.127	−0.33	−0.043	−0.417	−0.299	0.239	−0.092	−0.201
t−12	0.431	−0.19	−0.428	−0.11	−0.49	−0.364	0.321	−0.194	−0.276
t−13	0.403	−0.251	−0.485	−0.158	−0.512	−0.415	0.343	−0.238	−0.32
t−14	0.385	−0.288	−0.527	−0.197	−0.492	−0.434	0.352	−0.275	−0.359
t−15	0.353	−0.299	−0.524	−0.204	−0.47	−0.43	0.365	−0.374	−0.391
t−16	0.31	−0.286	−0.56	−0.191	−0.427	−0.39	0.378	−0.493	−0.424
t−17	0.262	−0.272	−0.529	−0.172	−0.355	−0.354	0.346	−0.525	−0.437
t−18	0.196	−0.241	−0.465	−0.12	−0.268	−0.297	0.288	−0.525	−0.434
t−19	0.165	−0.231	−0.412	−0.053	−0.206	−0.266	0.196	−0.498	−0.419
t−20	0.093	−0.22	−0.366	−0.02	−0.152	−0.215	0.106	−0.488	−0.418

Table 3.3 Autocorrelation coefficients for the sensors in shipment 3 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.996	0.996	0.997	0.99	0.998	0.993	0.996	0.999	0.996
t−2	0.993	0.992	0.996	0.982	0.996	0.988	0.993	0.998	0.994
t−3	0.99	0.989	0.995	0.975	0.995	0.982	0.99	0.997	0.992
t−4	0.987	0.985	0.994	0.966	0.993	0.978	0.988	0.995	0.989
t−5	0.984	0.982	0.993	0.957	0.992	0.974	0.985	0.994	0.987
t−6	0.98	0.979	0.992	0.949	0.991	0.972	0.982	0.993	0.985
t−7	0.976	0.977	0.991	0.941	0.99	0.971	0.98	0.992	0.983
t−8	0.972	0.975	0.99	0.935	0.988	0.972	0.977	0.991	0.981
t−9	0.969	0.973	0.989	0.931	0.987	0.971	0.974	0.989	0.978
t−10	0.965	0.97	0.988	0.927	0.985	0.969	0.971	0.988	0.976
t−11	0.962	0.967	0.987	0.925	0.983	0.966	0.968	0.986	0.973
t−12	0.958	0.964	0.986	0.922	0.982	0.964	0.965	0.984	0.971
t−13	0.955	0.961	0.985	0.92	0.98	0.959	0.962	0.982	0.967
t−14	0.951	0.958	0.984	0.918	0.978	0.955	0.958	0.979	0.963
t−15	0.947	0.955	0.983	0.917	0.976	0.952	0.953	0.977	0.96
t−16	0.942	0.952	0.982	0.915	0.974	0.95	0.948	0.974	0.956
t−17	0.938	0.949	0.981	0.912	0.971	0.948	0.943	0.971	0.953
t−18	0.933	0.946	0.979	0.909	0.969	0.947	0.938	0.968	0.949
t−19	0.928	0.944	0.977	0.905	0.967	0.945	0.933	0.965	0.946
t−20	0.923	0.941	0.976	0.9	0.965	0.945	0.928	0.962	0.942

Table 3.4 Autocorrelation coefficients for the sensors in shipment 4 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.992	0.997	0.998	0.96	0.978	0.976	0.994	0.983	0.997
t−2	0.988	0.995	0.996	0.938	0.967	0.965	0.989	0.973	0.996
t−3	0.984	0.993	0.994	0.914	0.956	0.959	0.985	0.969	0.996
t−4	0.981	0.991	0.992	0.893	0.94	0.952	0.98	0.966	0.995
t−5	0.978	0.989	0.989	0.867	0.926	0.946	0.974	0.959	0.994
t−6	0.975	0.987	0.987	0.845	0.916	0.941	0.969	0.954	0.994
t−7	0.971	0.985	0.984	0.828	0.909	0.937	0.966	0.947	0.994
t−8	0.965	0.981	0.981	0.809	0.901	0.935	0.961	0.941	0.993
t−9	0.961	0.978	0.977	0.789	0.895	0.929	0.957	0.937	0.993
t−10	0.955	0.974	0.973	0.77	0.885	0.921	0.952	0.931	0.992
t−11	0.949	0.97	0.969	0.752	0.875	0.911	0.948	0.926	0.991
t−12	0.943	0.966	0.966	0.729	0.867	0.905	0.944	0.922	0.991
t−13	0.936	0.961	0.962	0.702	0.854	0.895	0.938	0.919	0.99
t−14	0.928	0.956	0.957	0.675	0.84	0.888	0.933	0.912	0.989
t−15	0.919	0.951	0.952	0.648	0.833	0.885	0.928	0.906	0.988
t−16	0.911	0.946	0.947	0.627	0.821	0.879	0.924	0.9	0.987
t−17	0.902	0.941	0.94	0.603	0.811	0.875	0.919	0.894	0.986
t−18	0.892	0.935	0.934	0.588	0.799	0.87	0.915	0.888	0.985
t−19	0.882	0.93	0.927	0.589	0.787	0.865	0.911	0.881	0.984
t−20	0.873	0.924	0.92	0.593	0.772	0.861	0.908	0.877	0.983

Table 3.5 Autocorrelation coefficients for the sensors in shipment 5 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.996	0.996	0.991	0.974	0.917	0.958	0.998	0.993	0.949
t−2	0.993	0.995	0.987	0.96	0.877	0.937	0.997	0.988	0.901
t−3	0.991	0.994	0.985	0.95	0.865	0.919	0.996	0.982	0.846
t−4	0.99	0.993	0.985	0.938	0.851	0.905	0.995	0.977	0.794
t−5	0.988	0.992	0.983	0.925	0.841	0.901	0.994	0.973	0.731
t−6	0.987	0.991	0.981	0.906	0.839	0.893	0.993	0.97	0.676
t−7	0.986	0.99	0.98	0.895	0.833	0.887	0.993	0.967	0.66
t−8	0.985	0.99	0.978	0.882	0.804	0.882	0.992	0.965	0.659
t−9	0.985	0.989	0.976	0.866	0.765	0.871	0.992	0.963	0.65
t−10	0.984	0.988	0.975	0.858	0.746	0.857	0.991	0.963	0.648
t−11	0.983	0.987	0.974	0.855	0.735	0.834	0.991	0.961	0.643
t−12	0.981	0.986	0.974	0.851	0.735	0.808	0.99	0.959	0.647
t−13	0.978	0.985	0.973	0.83	0.703	0.793	0.99	0.956	0.672
t−14	0.977	0.984	0.972	0.816	0.699	0.772	0.989	0.952	0.662
t−15	0.975	0.983	0.971	0.802	0.694	0.763	0.988	0.949	0.619
t−16	0.975	0.982	0.968	0.791	0.684	0.754	0.988	0.947	0.559
t−17	0.974	0.982	0.967	0.777	0.667	0.743	0.987	0.944	0.515
t−18	0.974	0.981	0.966	0.762	0.655	0.73	0.985	0.942	0.481
t−19	0.973	0.98	0.966	0.748	0.632	0.72	0.985	0.939	0.443
t−20	0.972	0.979	0.966	0.737	0.616	0.708	0.983	0.937	0.408

Table 3.6 Autocorrelation coefficients for the sensors in shipment 6 for different lags.

Lags	FT	FM	FB	MT	MM	MB	RT	RM	RB
t−1	0.991	0.989	0.968	0.999	0.996	0.997	0.951	0.985	0.983
t−2	0.988	0.98	0.956	0.998	0.992	0.993	0.915	0.98	0.978
t−3	0.986	0.969	0.955	0.997	0.989	0.99	0.871	0.975	0.971
t−4	0.984	0.96	0.931	0.996	0.985	0.988	0.83	0.971	0.967
t−5	0.983	0.953	0.936	0.994	0.98	0.984	0.79	0.966	0.965
t−6	0.981	0.944	0.929	0.992	0.975	0.981	0.756	0.962	0.962
t−7	0.976	0.937	0.926	0.99	0.97	0.977	0.704	0.959	0.956
t−8	0.974	0.932	0.956	0.988	0.964	0.973	0.663	0.956	0.953
t−9	0.972	0.926	0.885	0.986	0.957	0.967	0.59	0.952	0.947
t−10	0.971	0.923	0.858	0.983	0.95	0.963	0.533	0.951	0.944
t−11	0.968	0.915	0.855	0.98	0.944	0.958	0.491	0.95	0.939
t−12	0.966	0.909	0.825	0.978	0.936	0.952	0.443	0.95	0.935
t−13	0.964	0.901	0.853	0.974	0.928	0.945	0.397	0.949	0.932
t−14	0.963	0.892	0.758	0.971	0.922	0.939	0.347	0.95	0.927
t−15	0.961	0.886	0.793	0.968	0.915	0.931	0.297	0.949	0.925
t−16	0.96	0.879	0.88	0.964	0.906	0.923	0.245	0.95	0.924
t−17	0.958	0.876	0.805	0.961	0.898	0.914	0.211	0.947	0.922
t−18	0.957	0.873	0.768	0.957	0.89	0.905	0.181	0.945	0.919
t−19	0.954	0.87	0.825	0.953	0.883	0.896	0.146	0.944	0.916
t−20	0.955	0.869	0.791	0.949	0.875	0.885	0.103	0.942	0.916

3.4 Discussion

One potential application which could be of interest to quality control personnel throughout the cold chain is to quickly and effectively detect any anomalies in the form of temperature abuses or breaks along the cold chain. In this particular instance, we had not observed any anomalies across any of the six shipments, however in case one wants to implement a general framework to detect such anomalies in a similar time series dataset the following discussion will introduce some of the methods widely accepted in the literature.

3.4.1 Applied Methods for Detecting Abnormality

Anomaly detection is defined by [110] as "the identification of the data that do not conform to the pattern as expected". FFT, and Pearson product-moment correlation coefficients with the help of running the Dickey-Fuller test [[111]] were used by [112] to first classify the time series into three distinct classes: (1) stationary, (2) periodic and (3) non-stationary and non-periodic to guide the selection of different thresholds for detecting the abnormal behaviours. For instance, if the time series is stationary, a local and a global mean from small and large windows are computed (where the mean difference is given by $mean = |MeanLocal - MeanGlobal|/MeanGlobal$ which is used as a threshold for identifying anomalies. With the help of FFT and the Pearson correlation coefficient, a time series can be classified into periodic or not, based on exceeding a certain correlation threshold. The authors use the standard deviations of two consecutive sliding time windows $RStd = |Stdcurrent - Stdprevious|/Stdprevious$, to detect an anomaly. Finally, the authors used deep learning approaches for sequence modeling using Gated Recurrent Units [[60]] to detect anomalies by fitting the prediction errors into a normal distribution, and by mapping them into a cumulative distribution function (CDF), where the model detect an anomaly if the CDF value exceeds a threshold. On the other hand, a DTW-based anomaly detector was used to detect cyber-attacks known as Distributed Denial of Service (DDoS), worm propagation, and port scanning attacks on the internet traffic network [113]. In this work, the authors use DTW to compute the non-linear alignment of the time series between control traffic packets and data traffic packets

to identify the dissimilar behavior between the two without the past familiarity of the network attacks. Similar to other approaches, a predefined hyperparameter is computed and then used as a threshold to find the outliers that fall outside the range of values defined by median absolute deviation (MAD), $MAD = median|x_i$ - $median(x_i)|$, where, x_i denotes individual observations in the given data. The authors consider any distance value more than three scaled MAD away from the median as an outlier. A more advanced unsupervised architecture was proposed in [114] that uses a variational autoencoder (VAE) with re-encoder and latent constraint network, named as VELC. VELC can find a compressed representation of the time-series data through a long-short-term-memory (LSTM) probabilistic encoder and learn the latent variables that generate the input data. The error reconstruction error is optimized both in the original space and the learned latent space to accurately model the normal time series. Subsequently, from the latent space distribution, the input time series is reconstructed through an LSTM probabilistic decoder. The authors achieved state-of-the-art performance in distinguishing normal and abnormal samples over ten datasets comparing 4 different models. Recently, a survey has been published by [115] on anomaly detection in univariate time-series to compare statistical approaches like autoregressive model (AR), moving average (MA), autoregressive moving integrated average model (ARIMA), and simple exponential smoothing (SES) with other machine learning algorithms like k-means clustering – subsequence time-series clustering (STSC), isolation forest, and extreme gradient boosting (XGBoost, XGB) and deep learning models like multiple layer perceptron (MLP), convolutional neural networks (CNN), LSTM, and autoencoders. Statistical methods assume the data is stationary, and mostly quantifies the anomaly score as the difference between the prediction and ground truth, i.e. the error. On the other hand, machine learning methods do not assume any prior knowledge on the underlying process of the data and can detect outliers either in a supervised or unsupervised manner based on whether the labels are used in the training phase to optimize the objective functions. For example, STSC applies a window approach to convert the time series into a set of sub-sequences and then apply k-means clustering to find the optimum number of clusters. An anomaly will be flagged if the error between the clusters centroids and each subsequence exceeds the predefined

threshold. Similar to machine learning approaches, deep learning neural networks learn more complex functions from the input time series and find the optimum parameter through backpropagation with gradient descent to minimize the cost function or error. The error is compared to a threshold to check for anomalies. Surprisingly, even as deep learning has gained immense popularity over the recent years, the published results conclude that the statistical models actually achieved the best results in detecting both point and collective anomalies while having trouble dealing with contextual anomalies. This finding highlights the main advantages of deep learning in understanding the context over a long period and across different variables and also learn good representations from the input data. On a dataset like the one introduced in this paper, a statistical approach is more likely to yield better results due to both the size of the data and some of the statistical similarities we observed among shipments and sensor locations.

3.4.2 Sensors' Variability Control

In this particular study, we introduce to the food engineering and time-series research communities a novel multivariate sensory time-series dataset collected as a result of significant collaboration between stakeholders from growers to distributors to retailers to academics. We instrument and monitor each shipment individually and on site such that the variability among them is rather controlled (almost the exact same places in the container for instance). Hence, some of the observations regarding "highly correlated" locations could be made in this article which would otherwise be inappropriate for an uncontrolled shipment. However, it is worth noting that in a much larger pilot with less control over the shipments, there will also be a lot more data coming from the sensors to help with the inherent variability issue.

Chapter 4: Sense2Vec: Time Series Representation

4.1 Note to Reader

Portions of this chapter have been previously published in our paper at IEEE Sensors Journal and have been reproduced with the permission from IEEE.

4.2 Introduction

The analysis and processing of multivariate and heterogeneous time-series data for predictive tasks represent a significant challenge especially when profiles may have variable lengths. Time-series analysis is ranked as the tenth most challenging problem in data mining due to its "unique properties" [116]. The manifestation of this practical challenge can be seen in different industries, particularly for the applications of sensor networks in monitoring production and distribution of goods around the globe. A comprehensive study was highlighted by [117] which shows that food production, distribution and consumption are the main factors that contribute to 60% of the overall food waste levels. In [97], the author(s) reported that 50% of harvested strawberries are wasted due to inadequate temperature monitoring and control. Hence, delivering high quality food along the supply chain is critical to both suppliers and consumers. The specific use case we provide in this initial application is the temperature monitoring of the perishable supply chain due to the universally accepted observation that temperature is the most critical environmental factor to consider along the cold chain. High temperatures lead to significant decay in shelf life, and low temperatures may lead to damaging the quality of delicate food. Monitoring and controlling the refrigeration of food along the cold-chain (transportation, storage, and distribution of perishable food items) are critical to reducing the amount of food waste. However, the cost of installation

of the monitoring devices such as wireless sensor networks (WSNs) and radio frequency identification (RFID) systems limits monitoring resolution generally to one per container [117], [97]. Thus, limiting the number of sensors requires careful analysis of the network through multivariate time-series analysis of the temperature profiles. This requires a new approach in how the collected time-series data is represented to the learning algorithm especially when recording times are not uniform. Over a half-century, statistical parametric methods provided the state-of-the-art performance for time-series analysis, modeling and prediction across different applications and fields [31]. To name a few, Auto-Regressive Moving Average (ARMA) [118], Auto-Regressive Integrated Moving Average(ARIMA) [35], Seasonal ARIMA [36], exponential smoothing [33] and Vector Autoregression (VAR) [41] models, are parametric methods and generally assume a known prior over the distribution of the time-series data which make them inadequate for many practical applications. The advancement of machine learning brought sophisticated non-parametric methods for time-series modeling and prediction [42], [43]. For instance, Artificial Neural Networks (ANNs) [11], [29] have attracted many researchers in the domain of time-series modeling and forecasting and were successfully applied to various time-series applications [47], [119]. Convolutional Neural Networks (CNNs) [6] have successfully been implemented across many domains including time-series forecasting [51], [52], and classification [53], [56]. Recurrent Neural Networks (RNNs) [2], [57], construct their hidden states "output" by autoregression of present on past values. Long short-term memory (LSTM) [59] and Gated Recurrent Unit (GRU) [60], are two advanced variations of RNN to address the long-term dependency and the vanishing and exploding gradient problems [66], [120]. LSTM has been applied for univariate [70], and multivariate time-series applications [73], [75], where Bidirectional LSTM (Bi-LSTMs) [76] can integrate information from both past and future time steps by means of two hidden states. Encoder-decoder structures [61], [62], encoder-decoder based attention [93], autoregressive recurrent networks [121], and LSTNet [122] use a combination of CNNs and RNNs. Most recently, transformers [64], and one of its variations, Bidirectional Transformers for Language Understanding (BERT) [123] were confirmed to be the state of the art on eleven natural language processing applications and

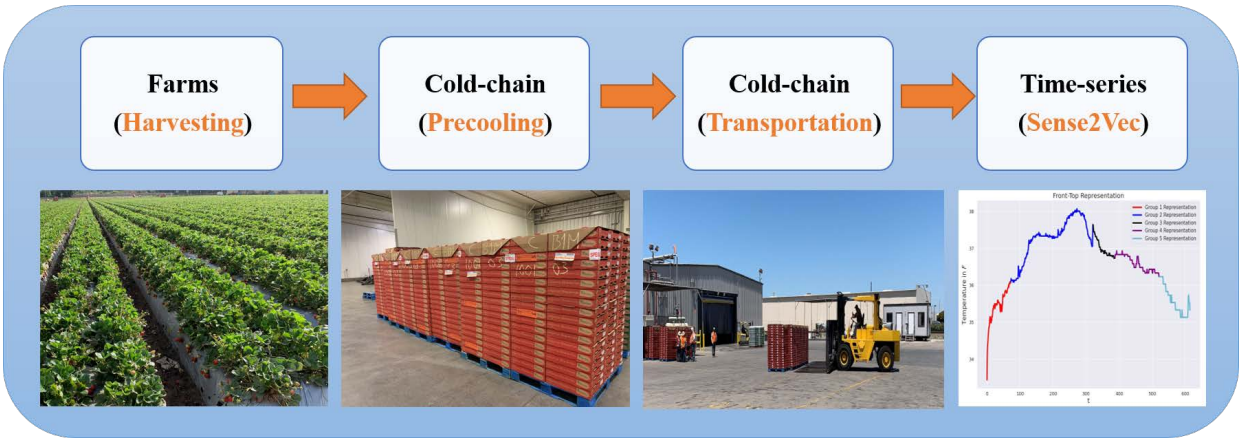

Figure 4.1 Cold-chain time-series based machine learning pipeline. The images show the real environment for data collection in this work.

XLNet [124], by leveraging the best of both autoregressive (AR) language modeling and autoencoding (AE). While the advances in both topology and parametric optimization methods for these advanced learning algorithms led to the improvements in performance, the specific way in which the time-series data is represented is a critical component of algorithm design.

4.3 Contribution

As the main contributions of this work, we propose three learning vector representation algorithms to combine multiple related time-series datasets of variable lengths using Pearson's correlation coefficient (CW-MAC), Dynamic Time Warping (DTW-MAC), and moving averages (UW-MAC) to generate a candidate time-series to be used in subsequent time-series applications such as regression, classification, and clustering.

4.4 Related Work

Handling multivariate time-series with variable lengths is a challenge and requires careful preprocessing. Preprocessing multivariate time-series with variable lengths has been studied for representation learning [125], [82], time-series prediction [83], [71], [84] clustering [85], [86], [87], classification [88], [74], [89], and anomaly detection [92], [91], [90], sequence modeling and machine translation [93], [61]. The challenges of preprocessing time-series with variable lengths

were addressed by [95], and [94] for clustering analysis. Zero padding method was used which is an invalid approach for time-series like temperature where the value 0 may represent an important value such as 0°C for the food transportation work discussed here. Another method is cropping, where all the time-series signals that has sequences longer than the shortest one are clipped. This approach leads to missing a lot of information especially for multivariate time-series with large variable lengths. The author(s) of [87] conclude that cropping may lead to underfitting, and use one large concatenation of the remaining values for each time step. In this work we propose a brand-new approach to better summarize and visualize multi-variate time series data coming from numerous sources in a sensor network to gather actionable and robust insight while maintaining accurate information for further analysis, prediction and classification of data. Time-series for wireless sensor reduction using a theoretical framework was studied by [96]. Also, artificial neural networks (ANN) has been applied to learn the mapping between sensors' locations in the application of cold-chain using time-series data [97], [98], [99]. However, all the experiments were conducted on simulated temperature data, which does not adequately represent the real-world's complex non-linear multivariate time-series data with variable lengths used in this paper. In a prediction scenario, it is important to consider different environments with highly dynamic and complex operations. Precooling is an excellent example where the airflow in the tunnel, loading temperature heterogeneity, forklift movements, air velocity, and relative humidity create a very complex modeling environment. As a result, this requires a new approach to provide the following:

- A reliable and robust representation of the time-series data where noise and other outliers are removed with a weighted averaging operation.
- A scalable solution where one can analyze time-series data with any length in duration including any number of variables.
- A compact yet memory effective representation by compressing potentially very high-volume data into small vectorial representations.

4.5 Sense2Vec Algorithms

4.5.1 Problem Formulation

Assume we have a collection of multivariate time-series sensor data denoted by X_i^j where subscript i indicates the sensor ID (i.e., location) and superscript j indicates the shipment number for that sensor (i.e., first, second, etc.). For instance, in our specific scenario where 9 sensors were placed in five different shipments, X_1^3 means the temperature vector collected for the third shipment from the first sensor location. Please note that none of the temperature vectors need to be of the same size and the proposed algorithm is specifically designed to account for such a condition.

Assuming m sensor locations and k shipments. First, we perform ascending ordering on the time-series signals X_i, for $i = 1,2,3,\ldots,m$, with superscripts 1 and k for the shortest and longest signals in time respectively: $X_i^{[1]}$; $X_i^{[2]}$; $X_i^{[3]}$; $\ldots$; $X_i^{[k]}$. For the application described in this paper, each shipment has nine sensors X_{FT}, X_{FM} ,X_{FB}, X_{MT}, X_{MM}, X_{MB}, X_{RT}, X_{RM}, X_{RB}. The subscripts stand for front-top, front-middle, front-bottom, middle-top, middle-middle, middle-bottom, rear-top, rear-middle and rear-bottom respectively where the first word describes the location of the sensor-instrumented pallet in the container (front, middle or rear) and the second word describes the location of the sensor in the pallet itself (top, middle or bottom).

Next, we compute k temporal Pearson's Correlation matrices starting by clipping the length of all $k-1$ signals to have the same length as the shortest signal. Then, we compute $k \times k$ cross-correlation matrix where σ_{ij} represents the Pearson's correlation coefficient between time-series i and time-series j.

Later, we clip all signals to have the same size as the shortest signal in the group and calculate the $k \times k$ Pearson's correlation matrix. Next we clip all signals to have the same size as the second shortest signal in the group and calculate the $(k-1) \times (k-1)$ correlation matrix. We continue this process until we are left with the longest signal in the group to generate a total of k correlation matrices each generated from different clusters of clipped signals as shown below where σ_{ij} represents the Pearson's correlation coefficient between signals i and j:

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1k} \\ \sigma_{21} & \sigma_{22} & \ddots & \vdots \\ \sigma_{31} & \ddots & \ddots & \sigma_{3k} \\ \vdots & & \sigma_{k-1k-1} & \sigma_{k-1k} \\ \sigma_{k1} & \cdots & \sigma_{kk-1} & \sigma_{kk} \end{bmatrix}; \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1k-1} \\ \sigma_{21} & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \sigma_{k-11} & \cdots & \sigma_{k-1} \end{bmatrix};$$

$$\begin{bmatrix} \sigma_{11}\cdots & \sigma_{1k-2} \\ \vdots & \vdots \\ \sigma_{k-21}\cdots & \sigma_{k-2} \end{bmatrix}; \cdots \begin{bmatrix} 1 \end{bmatrix}$$

Pearson's product-moment correlation coefficients can be written according to [105] as:

$$\sigma_{xy} = \frac{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_i})^2}\sqrt{\sum_{i=1}^{n^{[0]}}(y_i - \bar{y_i})^2}} \tag{4.1}$$

where $n^{[0]}$ is the length of the shortest signal in the group for the first correlation matrix which includes all k signals. It is calculated similarly for the subsequent correlation matrices with $k-1$ signals where $\bar{x_i}$ is the average means of the ordered time-series signals of length $n^{[0]}$.

$$\bar{x_i} = \frac{1}{n^{[0]}}\sum_{i=1}^{n^{[0]}}(x_i) \tag{4.2}$$

The main motivation of selecting Pearson's product-moment correlation coefficients when analyzing the time-series profiles in each group is to look at the similarities and differences between different sensor locations inside different shipments or shipping-containers. Pearson's correlation coefficient provides a robust way to summarily visualize the temporal similarities which may exist. For example, a high correlation coefficient between two different locations may indicate that a single sensor could be sufficient to represent both temperature recordings whereas low correlation coefficients across the board may indicate significant differences in temperature distributions

inside the container. Next, for each Pearson's correlation matrix, we compute normalized weight coefficients for each signal as follows:

$$W_i^{[p]} = \frac{\sum_{j=1}^{k} \sigma_{(p+1)j} - 1}{(\sum_{i=1}^{k}(\sum_{j=1}^{k} \sigma_{ij} - 1))} \tag{4.3}$$

where $p = 1, 2, \ldots k$ represents the normalized weight coefficients for the first, second, $\ldots m$ time-series signals, respectively.

It is important to note that the sum of normalized weight coefficients for each matrix is intuitively equal to one: $\sum_{p=1}^{k}(W_i^{[p]}) = 1$.

Finally, the normalized weight coefficients are used to construct a correlation based representative signal for that sensor location by capturing the underlying distributions for each group while preserving the temporal nature of signals.

The first group of time-series signal can be combined as following:

$$\hat{X}_{group_1}^{[1]} = W_k^{[1]}.X_i^{[1][0:n^{[0]}]} + W_k^{[2]}.X_i^{[2][0:n^{[0]}]} + \cdots W_k^{[k]}.X_i^{[k][0:n^{[0]}]} \tag{4.4}$$

$$\hat{X}_{group_2}^{[2]} = W_{k-1}^{[2]}.X_i^{[2][n^{[0]}:n^{[1]}]} + W_{k-1}^{[3]}.X_i^{[3][n^{[0]}:n^{[1]}]} + \cdots W_{k-1}^{[k]}.X_i^{[k][n^{[0]}:n^{[1]}]} \tag{4.5}$$

$$\hat{X}_{group_3}^{[3]} = W_{k-2}^{[3]}.X_i^{[3][n^{[2]}:n^{[3]}]} + W_{k-2}^{[4]}.X_i^{[4][n^{[2]}:n^{[3]}]} + \cdots W_{k-2}^{[k]}.X_i^{[k][n^{[2]}:n^{[3]}]} \tag{4.6}$$

$$\vdots \quad = \quad \vdots$$

$$\hat{X}_{group_k}^{[k]} = W_1^{[k]}.X_i^{[k]n^{[k-1:k]}} \tag{4.7}$$

Recall that $W_1^{[k]} = 1$. and $X_i^{[k]n^{[k-1:k]}}$ is the remaining records of the longest time-series signal.

The final compound signal for the first group can be obtained by concatenate vertically all the $\hat{X}^{[i]}_{group_j}$ to form one time-series signal that is the best representative to all the individual time-series signals.

$$\hat{X}_i^{[Representative]} = [\hat{X}^{[1]}_{group_1}, \hat{X}^{[2]}_{group_2}, \hat{X}^{[3]}_{group_3}, \ldots, \hat{X}^{[k]}_{group_k}] \tag{4.8}$$

Notice that $\hat{X}_i^{[Representative]}$ will have the same length as the longest time-series signal among the group that we want to represent by one candidate signal.

Please note that this problem formulation is based upon the temporal correlations of each temperature vector to identify which ones have more influence in generating the representative temperature profile. Hence, we call this the correlation-weighted moving average coefficient algorithms or CW-MAC.

The second variation of the algorithm is called dynamic time warping (DTW) moving average coefficient (DTW-MAC) and can be achieved by computing the normalized weights for each group using the DTW distances according to [106] as a measure of comparing one time-series profile to another. DTW has been successfully implemented across different application domains [[126], [127]]. The authors of [128] stated that "after an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount." DTW measures similarity between two time-series sequences of temporal and phase misalignments and with different lengths and allows sequences to be expanded or compressed along the time axis [129]. Mainly, DTW warps the time axis of one (or both) temperature profile sequences to achieve a better alignment. To align two sequences, DTW constructs an $n_x \times n_y$ matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $d(x_i, y_j)$ between the two temperature points x_i and y_j (typically the Euclidean distance is used, so $d(x_i, y_j) = \sqrt{\sum(x_i - y_j)^2}$). Each matrix element (i, j) corresponds to the alignment between the temperature points x_i and y_j. DTW warping path is subject to three constraints according to [105]: boundary conditions, continuity and monotonicity; to minimize the overall warping cost, it can be written as follows:

Algorithm I: Correlation Weighted - Moving Average Coefficient (CW-MAC)

Input: $\Omega_{input} = \{ \mathscr{X}_i^{n^{[0]}}, \mathscr{X}_i^{n^{[1]}}, ..., \mathscr{X}_i^{n^{[k]}} \}$

Output: A compound time-series signal that represents all the variable-length input signals.

foreach *sensor i* ***in*** *allsensors* **do**

foreach *shipment j* ***in*** *allshipments* **do**

Select same location time-series across different shipments j:

Order set Ω_{input} in ascending order

Apply clipping to all vectors to achieve the same size as the shortest one $\mathscr{X}_i^{n^{[0]}}$

end

Compute k Pearson's Correlation coefficients to form a temporal matrices where:

$$\sigma_{xy} = \frac{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_k})(y_i - \bar{y_k})}{\sqrt{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_k})^2}\sqrt{\sum_{i=1}^{n^{[0]}}(y_i - \bar{y_k})^2}}$$

Compute a normalized weight coefficients for each row of different coefficients in the matrix by :

$$\mathscr{W}_i^{[p]} = \frac{\sum_{j=1}^{k}\sigma_{(p+1)j} - 1}{(\sum_{i=1}^{k}(\sum_{j=1}^{k}(\sigma_{ij} - 1) - m}$$

while $\mathscr{W}_i^{[p]} \neq W_1^{[k]} = 1$ **do**

Compute $\{\mathscr{W}_{m-1}^{[k-1]}, \mathscr{W}_{m-2}^{[k-2]}..., 1\}$

Combine the input signals in each group using the weight as:

$$\hat{\mathscr{X}}_{group_1}^{[1]} = \mathscr{W}_k^{[1]}.\mathscr{X}_i^{[1][0:n^{[0]}]} + \mathscr{W}_k^{[2]}.\mathscr{X}_i^{[2][0:n^{[0]}]} + \cdots \mathscr{W}_k^{[k]}.\mathscr{X}_i^{[k][0:n^{[0]}]}$$

$$\hat{\mathscr{X}}_{group_2}^{[2]} = \mathscr{W}_{k-1}^{[2]}.\mathscr{X}_i^{[2][n^{[0]}:n^{[1]}]} + \mathscr{W}_{k-1}^{[3]}.\mathscr{X}_i^{[3][n^{[0]}:n^{[1]}]} + \cdots \mathscr{W}_{k-1}^{[k]}.\mathscr{X}_i^{[k][n^{[0]}:n^{[1]}]}$$

$$\hat{\mathscr{X}}_{group_3}^{[3]} = \mathscr{W}_{k-2}^{[3]}.\mathscr{X}_i^{[3][n^{[2]}:n^{[3]}]} + \mathscr{W}_{k-2}^{[4]}.\mathscr{X}_i^{[4][n^{[2]}:n^{[3]}]} + \cdots \mathscr{W}_{k-2}^{[k]}.\mathscr{X}_i^{[k][n^{[2]}:n^{[3]}]}$$

$$\vdots \quad = \quad \vdots$$

$$\hat{\mathscr{X}}_{group_k}^{[k]} = \mathscr{W}_1^{[k]}.\mathscr{X}_i^{[k]n^{[k-1:k]}}$$

Finally form one time-series signal by concatenate the $\hat{\mathscr{X}}_{group_1}^{[j_s]}$.

$$\hat{\mathscr{X}}_i^{[Representative]} = [\hat{\mathscr{X}}_{group_1}^{[1]}, \hat{\mathscr{X}}_{group_2}^{[2]}, \ldots, \hat{\mathscr{X}}_{group_k}^{[k]}]$$

end

end

$$DTW(x,y) = min\left\{\frac{\sqrt{\sum_{l=1}^{L}(z_i)}}{L}\right\} \tag{4.9}$$

where L is used to compensate for the fact that warping paths may have different lengths. In order to find the minimum path, the warping path Z is contiguous: $Z = z_1, z_2, \ldots, z_L$ and $max(n_x, n_y) \leqslant L < (n_x + n_y\text{-}1)$. DTW uses dynamic programming to compute the cumulative distance $\zeta(i,j)$ from $d(i,j)$ "current position" in the matrix and the minimum of the cumulative distances of the adjacent elements:

$$\zeta(i,j) = d(x_i, y_j) + min\{\zeta(i-1, j-1) + \zeta(i-1, j) + \ldots$$

$\zeta(i, j-1)$(4.10)

Here, we compute $\zeta(i,j)$ for each pair of profiles in each group, then we normalize it and use it as the new normalize weights $W_i^{[p]}$ in equations 4.4 to 4.7.

The main motivation of comparing the CW-MAC representations with the DTW-MAC representations is to quantify the reading consistency among related time-series profiles in general and more specifically for each group. Hence, small DTWs indicate stronger temporal alignments or more similarities between sensors and represent a good illustration of the reading consistency of strawberry temperatures among different sensors inside the shipping containers. Conversely, large DTWs indicate higher levels of misalignment in terms of the profile behavior. We believe that the differences in DTW distance distributions computed due the experimentation phase, such as the means, standard deviations, and the skewness directions can jointly be interpreted as challenging indicators for the temporal heterogeneity, complexity, similarity, and discrepancy of the collected multivariate time-series. These distance−based distributions can be helpful in location−based predictions for wireless sensor networks and data analytics applications as the selection, identification, and grouping of the best candidates are necessary preprocessing steps for increasing the accuracy of time-series forecasting, clustering, and classification applications.

Algorithm II: Dynamic Time Warping Moving Average Coefficient (DTW-MAC)

Input: $\Omega_{input} = \{ X_i^{n^{[0]}}, X_i^{n^{[1]}}, ..., X_i^{n^{[k]}} \}$

Output: A compound time-series signal that represents all the variable-length input signals.

Repeat the first steps similar to the CW-MAC: ordering, and clipping.

Compute k Dynamic time warping coefficients to form a nonlinear alignment matrices of distances:

Initial: $dis = list[\,]; DTW = dict\{\}$

foreach *sensor i* ***in*** *group k* **do**

DTW[(i, -1)] = float('infinite')

DTW[(-1, i)] = float('infinite')

DTW[(-1, -1)] = 0

end

foreach *sensor i* ***in*** $n^{[0]}$ **do**

foreach *sensor j* ***in*** $n^{[0]}$ **do**

{if i is equal to j, then (dist = 0)}

$dist = (sensori[i] - sensori[j])$ ... $dis.append(dist)$

$DTW[(i,j)] = dist + min\{DTW[(i-1,j)], DTW[(i,j-1)], DTW[(i-1,j-1)]\}$

$d = \sqrt{(DTW[n^{[0]}-1], n^{[0]}-1])}$

end

end

Compute a normalized weight coefficients for each row of different coefficients in the matrix by:

$$W_i^{[p]} = \frac{1}{\left(\frac{\Sigma_{j=1}^{k} d_{(p+1)j}-1}{(\Sigma_{i=1}^{k}(\Sigma_{j=1}^{k}(d_{ij}-1)-m}\right)}$$

while $W_i^{[p]} \neq W_1^{[k]}$ = *1* **do**

Compute $\{W_{m-1}^{[k-1]}, W_{m-2}^{[k-2]} ..., 1\}$

Combine input signals in each group using weights as similar to equations $4, 5, 6$ **and** 7:

Compute: $\left[\hat{X}_{group_1}^{[1]}, \hat{X}_{group_2}^{[2]}, \hat{X}_{group_3}^{[3]}, \hat{X}_{group_k}^{[k]}\right]$

Finally form one time-series signal by concatenate the $\hat{X}_{group_1}^{[j_s]}$.

$\hat{X}_i^{[Representative]} = [\hat{X}_{group_1}^{[1]}, \hat{X}_{group_2}^{[2]}, \dots]$

end

A different and more trivial variation of the algorithm can be achieved by setting each correlation based weight to be the same (unit value) where we assume each shipment has the same impact on the representative profile. This variation of the proposed algorithm is called unity-weighted moving average coefficient algorithm or UW-MAC and will be included as a baseline comparison in the analysis to follow.

Finally, these representation methods can further help in aggregating similar shipments to help reduce the redundancy of selecting similar time-series profiles for representation which would ultimately reduce the time and memory complexity of the time-series analysis as well.

4.6 End to End Example

To assist the reader in understanding how exactly the proposed method work, in this example, we show how the computation of the algorithm is performed to obtain the final representation (combined vector) of the input related time-series profiles.

- Input Dataset = $[ship1, ship2, ship3, ship4, ship5]$, with lengths equal to $[72, 615, 387, 521, 321]$, respectively. Each time vector is obtained by taking the portion of the sensor recording from the end of the precooling to the arrival at DC.

- Order the shipments/datasets in ascending order with respect to time length.

- Ordered-Input = $[ship1, ship5, ship3, ship4, ship2]$ and their corresponding lengths are ordered as following: $[72, 321, 387, 521, 615]$.

Create groups of sensors related to one another in a specific way such as the position inside the container or being in the same shipment. In this example, we use position. For example, for the Front-Top sensor set:

$Front_{Topsensorset} = concatenate([Shipment1[Front_{Top}], Shipment2[Front_{Top}],$
$Shipment3[Front_{Top}], Shipment4[Front_{Top}], Shipment5[Front_{Top}])$.

The profiles are shown in Figure 4.2.

Algorithm III: Unity Weighted Moving Average Coefficient (UW-MAC)

Input: Ω_{input} = { $\mathscr{X}_i^{n^{[0]}}$, $\mathscr{X}_i^{n^{[1]}}$, ..., $\mathscr{X}_i^{n^{[k]}}$ }

Output: A compound time-series signal that represents all the variable-length input signals.

foreach *sensor i* ***in*** *allsensors* **do**

  foreach *shipment j* ***in*** *allshipments* **do**

    Select same location time-series across different shipments j:

      Order set Ω_{input} in ascending order

      Apply clipping to all vectors to achieve the same size as the shortest one $\mathscr{X}_i^{n^{[0]}}$

  end

  Compute Moving Average (MA) on the first group of k clipped signals to form a combined averaged time-series with length is equal to: $n^{[0]}$

  Compute:

$$\hat{\mathscr{X}}_{group_1}^{[0]} = \frac{1}{k} * (\mathscr{X}_1^{n^{[0]}} + \mathscr{X}_2^{n^{[0]}} + \cdots \mathscr{X}_k^{n^{[0]}})$$
$$\hat{\mathscr{X}}_{group_2}^{[1]} = \frac{1}{k-1} * (\mathscr{X}_2^{n^{[1]}} + \mathscr{X}_3^{n^{[1]}} + \cdots \mathscr{X}_{k-1}^{n^{[1]}})$$
$$\hat{\mathscr{X}}_{group_3}^{[2]} = \frac{1}{k-2} * (\mathscr{X}_3^{n^{[2]}} + \mathscr{X}_4^{n^{[2]}} + \cdots \mathscr{X}_{k-2}^{n^{[2]}})$$
$$\vdots \quad = \quad \vdots$$
$$\hat{\mathscr{X}}_{group_1}^{[k]} = \mathscr{X}_i^{[k]n^{[k-1:k]}}$$

  Finally form one time-series signal by concatenate the $\hat{\mathscr{X}}_{group_1}^{[js]}$.

$$\hat{\mathscr{X}}_{set_1}^{[ToTAL]} = [\hat{\mathscr{X}}_{group_1}^{[0]}, \hat{\mathscr{X}}_{group_1}^{[1]}, \ldots, \hat{\mathscr{X}}_{group_1}^{[k]}]$$

end

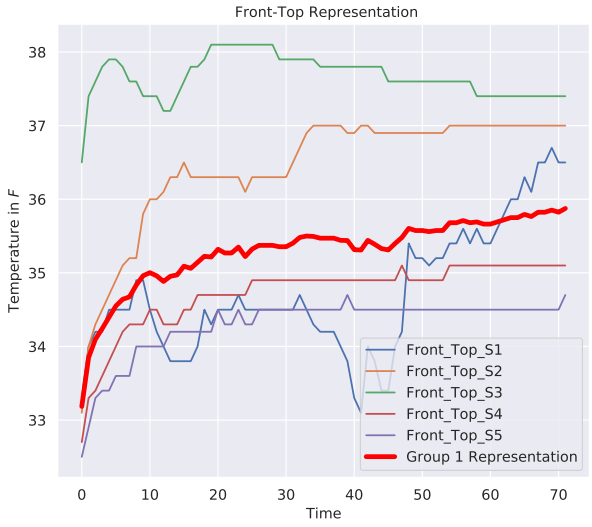

Figure 4.2 CW-MAC representation for the first group of five time-series profiles clipped to a fixed length equal to the shortest signal's length in the group $n^{[0]}$ =72.

Apply clipping to all Front-Top vectors in the set to achieve the same size as the shortest one $X_i^{n^{[0]}}$ where in this example $n^{[0]} = 72$.

Compute k Pearson's Correlation coefficients to form a temporal matrix where k is the number of equal-length time-series profiles in each group.

$$\sigma_{xy} = \frac{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_{i=1}^{n^{[0]}}(x_i - \bar{x_i})^2}\sqrt{\sum_{i=1}^{n^{[0]}}(y_i - \bar{y_i})^2}} \tag{4.11}$$

where $\bar{x_i} = \frac{1}{n^{[0]}}\sum_{i=1}^{n^{[0]}}(x_i)$

We have here a 5×5 symmetric matrix as shown below, where the diagonals are equal to one. σ_{12} represents the Pearson's correlation coefficient between the Front-Top time-series profile from the first shipment in the ordered list of shipments and the second ones; in this case the correlation coefficient between shipment 1 and shipment 5. The results of the first Pearson's correlation is shown in Figure 4.3. The heatmap in Figure 4.3 reveal the heterogeneity and complexity of the collected multivariate time-series by demonstrating clear differences in the gradient variations across different sensors among different shipments. The gradient levels are driven by the strength

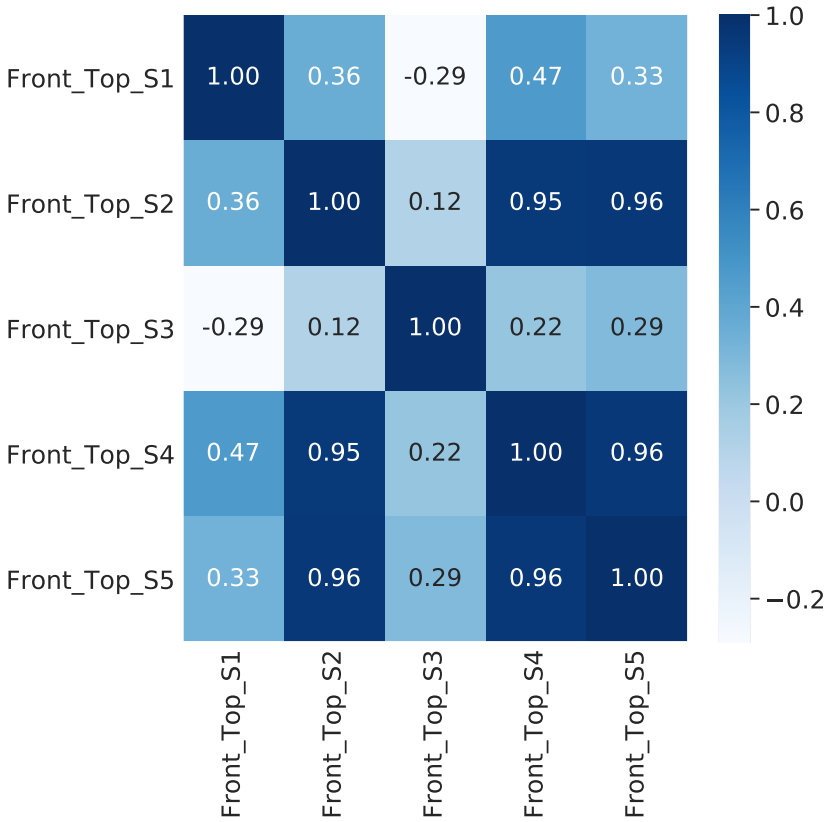

Figure 4.3 Pearson's correlation coefficients heat-map.

of the correlations between the temperature sensors profiles. Also, one can observe that group 1 correlations vary from small negative numbers "weak correlation" to large numbers "strong correlation" to reflect the level of reading consistency of strawberry temperatures among different shipments and location-related sensors.

Compute normalized weight coefficients for each row in the matrix $corr_{Front-Top_{group1}}$ by:

$$W_1 = \sigma_{12} + \sigma_{13} + \sigma_{14} + \sigma_{15}$$
$$W_2 = \sigma_{21} + \sigma_{23} + \sigma_{24} + \sigma_{25}$$
$$W_3 = \sigma_{31} + \sigma_{32} + \sigma_{34} + \sigma_{35}$$
$$W_4 = \sigma_{41} + \sigma_{42} + \sigma_{43} + \sigma_{45}$$

$$W_5 = \sigma_{51} + \sigma_{52} + \sigma_{53} + \sigma_{54}$$

$$W_{total} = W_1 + W_2 + W_3 + W_4 + W_5$$

$$W_{k=5}^{[1]} = \frac{W_1}{W_{total}}; W_{k=5}^{[2]} = \frac{W_2}{W_{total}}; W_{k=5}^{[3]} = \frac{W_3}{W_{total}}; W_{k=5}^{[4]} = \frac{W_4}{W_{total}}; W_{k=5}^{[5]} = \frac{W_5}{W_{total}}$$

where $W_{k=5}^{[1]}$ is the normalized weight for the shortest profile in the first group, and $W_{k=5}^{[5]}$ is the normalized weight for the first 72 timestamps of the longest profile of Front-Top sensor.

Create the first group-specific representation by the weighted sum of the time-series profiles using $W_k^{[i]}$ as following:

$$\hat{FT}_{group_1}^{[1]} = W_k^{[1]}.FT_1^{[ship1][0:n^{[0]}]} + W_k^{[2]}.FT_1^{[ship2][0:n^{[0]}]} + W_k^{[3]}.FT_1^{[ship3][0:n^{[0]}]} + W_k^{[4]}.FT_1^{[ship4][0:n^{[0]}]} + W_k^{[5]}.FT_1^{[ship5][0:n^{[0]}]}$$

where k = 5, $n^{[0]}$ =72, and $\hat{FT}$ is an abbreviation for Front-Top with [ˆ] to indicate group specific profiles. The resulted group-specific representation is shown by Figure 4.2.

The same procedure is repeated by removing shipment 1 from the ordered list and taking the next 73:321 samples for the remaining 4 vectors to generate a new 4×4 correlation matrix and following the same procedure. The result of the remaining group-specific representations is shown by Figure 4.4.

Finally, the overall combined representation for the Front-Top sensor is shown in Figure 4.5, which can now be used in location−based predictions for wireless sensor networks and data analytics applications, where physical sensors can be removed, and their readings predicted using other sensors.

4.6.1 Experimental Results

We present the results of our algorithms across two different applications:

- Network level: in this scenario, we obtain one representation vector per shipment. This representation serves as a compact high-level visualization of the overall sensor profiles measuring the different cold-chain stages/applications. Figures 4.6 to 4.10 shows the one-vector

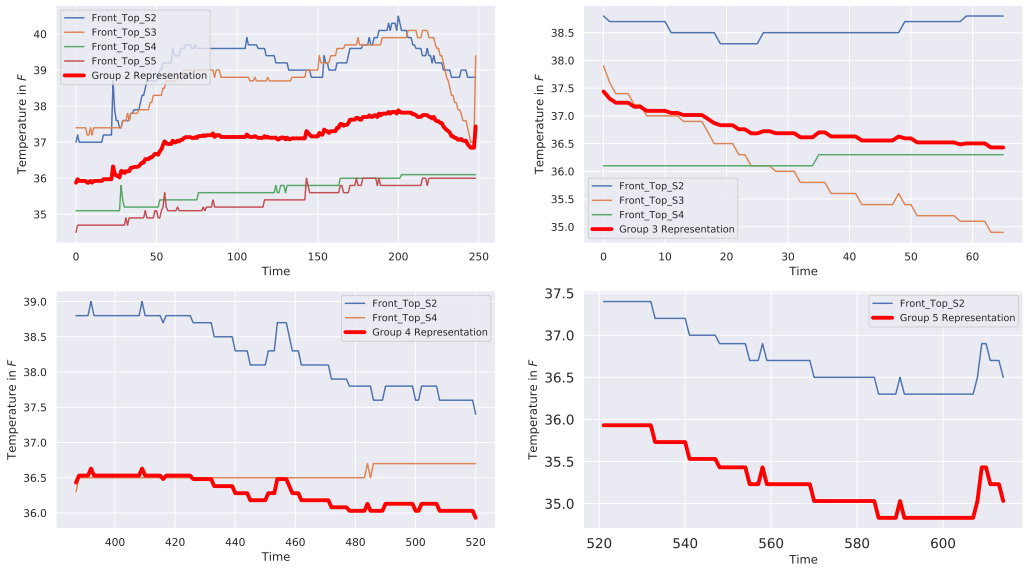

Figure 4.4 CW-MAC representations for the second, third, forth, and fifth group of Front-top time-series profiles that are differently clipped to a fixed length equal to the shortest signal's length in each one of the groups.

representation of the entire network of different sensors inside each shipping container. The red-line in each figure represents how CW-MAC algorithms encode all the profiles into one compact representation that reduces the dimensionality significantly from 9 time-series profiles and provide a better way of visualizing the overall sensor readings.

- Network of networks level: in this scenario, we attempt a harder problem than the network level. Unlike the previous case, in the network of networks scenario, sensors across different shipments are combined with different lengths, which is one of the main motivations in proposing these algorithms. We obtain one representation vector per sensor location. For example, finding a compact representation of the same sensor location across different shipments and environments which provides greater flexibility when different points of view are needed to analyze a situation. Figures 4.11 through 4.18 show how the CW-MAC algorithm finds one-vector representation of all the different profiles for the same-locations and across different shipments. In this case, one vector representation for all five shipments, not only

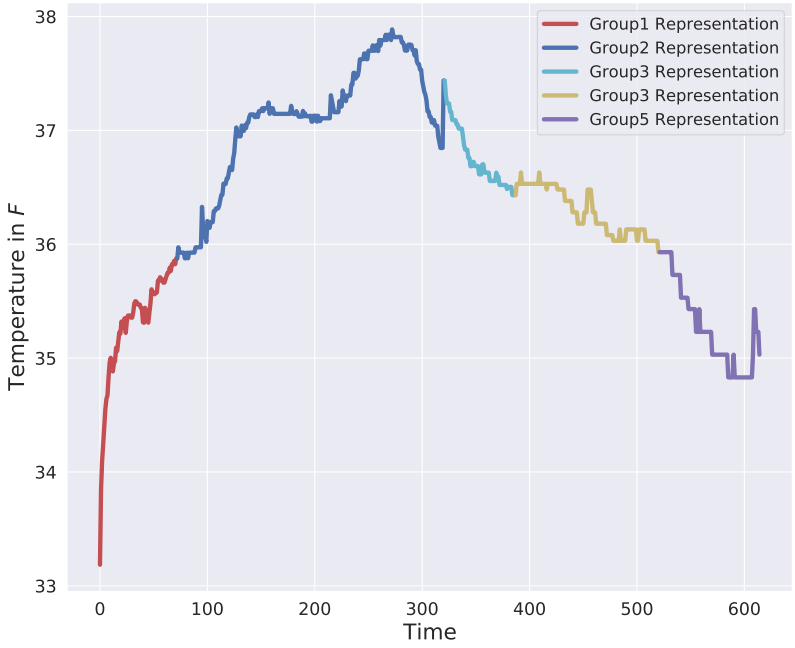

Figure 4.5 CW-MAC representation for Front-Top, the colors show group specific representations.

reduces the dimensionality but provide a better understanding of what happened to the temperature profiles, what was expected, and how. The results for DTW-MAC and UW-MAC are shown in Figures 4.19 through 4.26, Figures 4.27 through 4.34, respectively.

We start with the objective that we want to combine multiple time-series signals that are similar in behavior, but has different lengths due to all the variability in the monitored environment. Can we find a good candidate signal that can summarize the temperature in the cold-chain process for all sensors profiles and at the same time reveal the location of the sensors? One can see that sensors facing the ceiling of the container will be more exposed to sun radiations compared to the middle and the bottom and thus, their representations should somehow have a higher temperature increase due to the transportation phase across different shipments. This vectorial representation of different/related time-series profiles can be interpreted as a latent representation of the overall "sensor behavior" in the network through the cold-chain cycle. The combined vectors serve as an encoded representation of the original time series recordings. We think of CW-MAC, DTW-MAC,

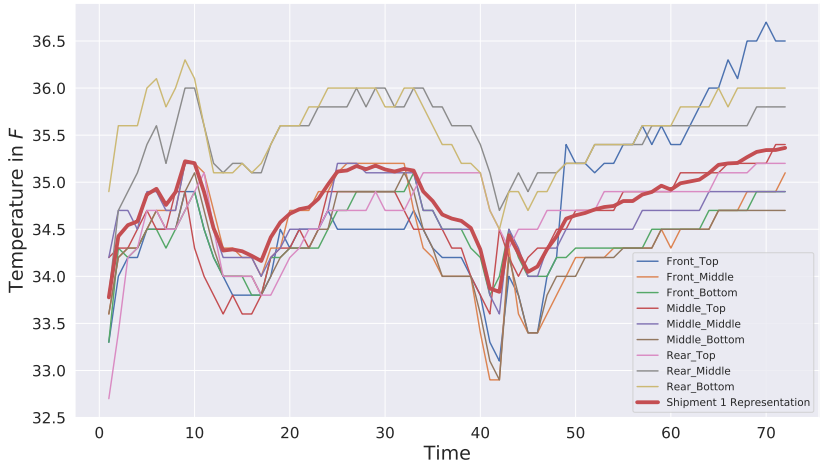

Figure 4.6 Shipment 1 time-series representation using CW-MAC.

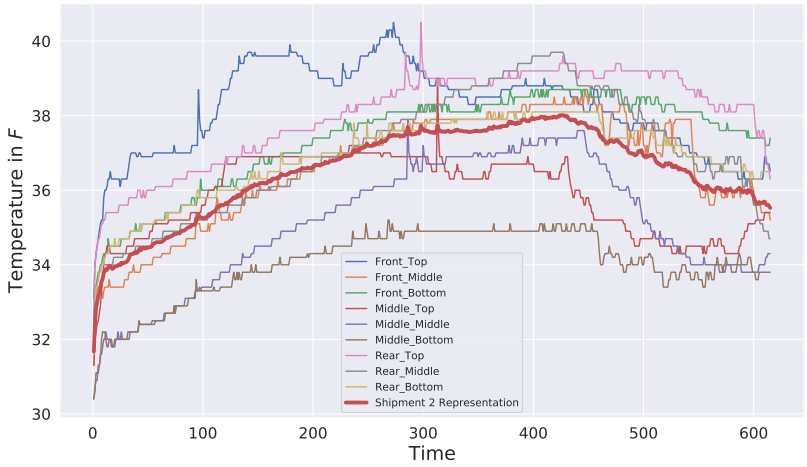

Figure 4.7 Shipment 2 time-series representation using CW-MAC.

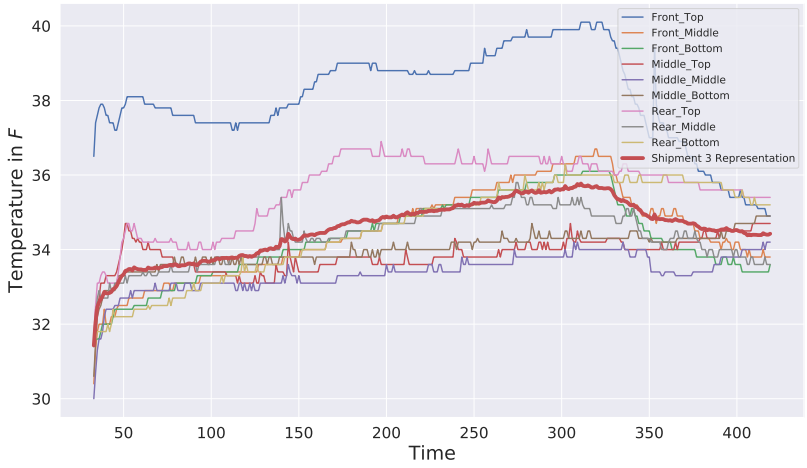

Figure 4.8 Shipment 3 time-series representation using CW-MAC.

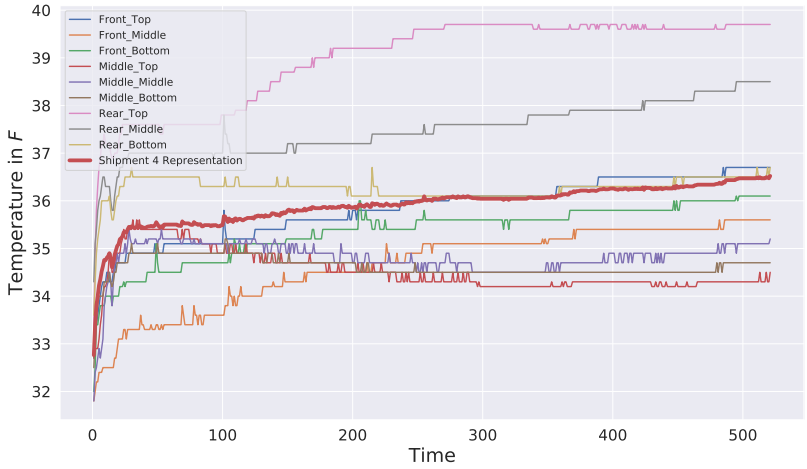

Figure 4.9 Shipment 4 time-series representation using CW-MAC.

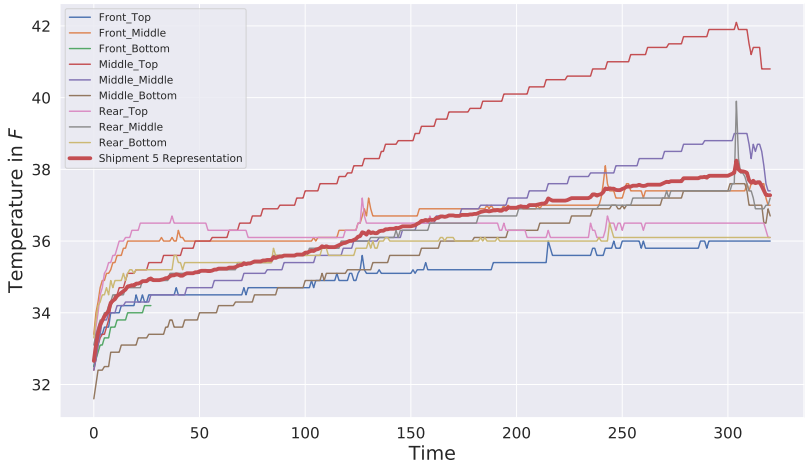

Figure 4.10 Shipment 5 time-series representation using CW-MAC.

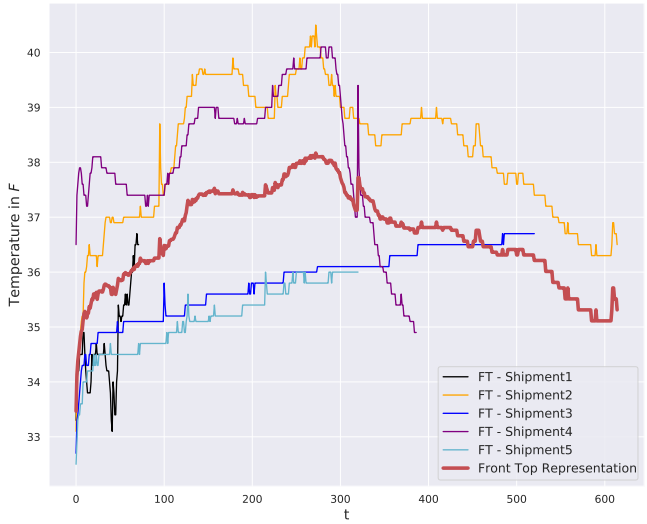

Figure 4.11 Front-Top representation using CW-MAC.

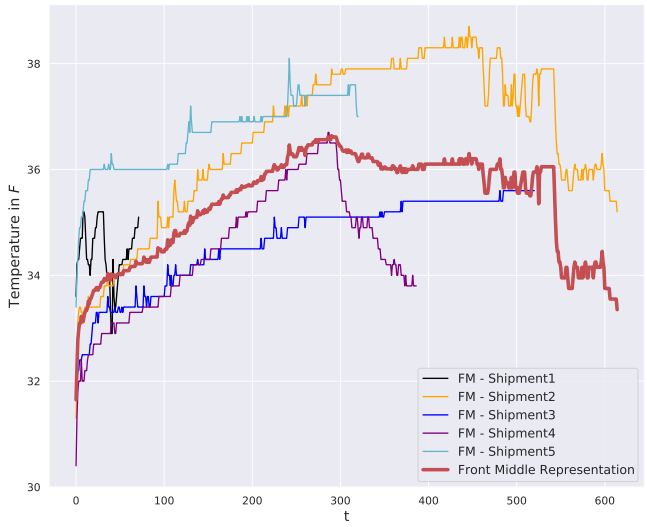

Figure 4.12 Front-Middle representation using CW-MAC.

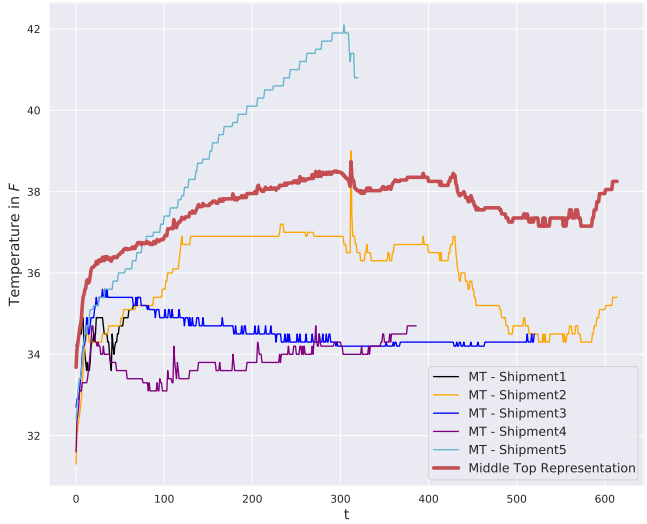

Figure 4.13 Middle-Top representation using CW-MAC.

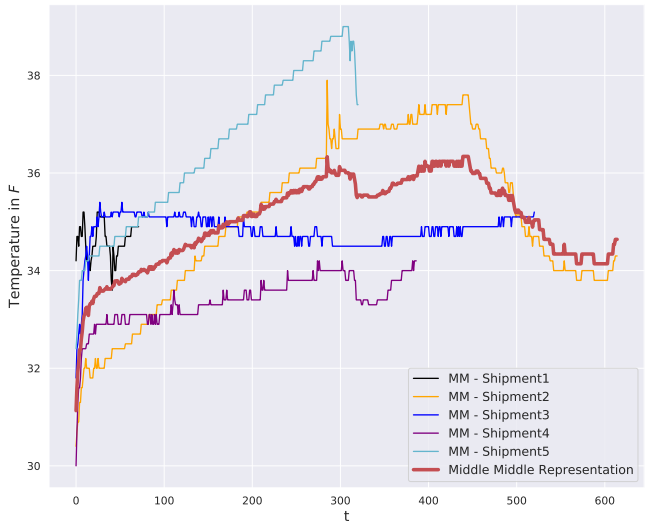

Figure 4.14 Middle-Middle representation using CW-MAC.

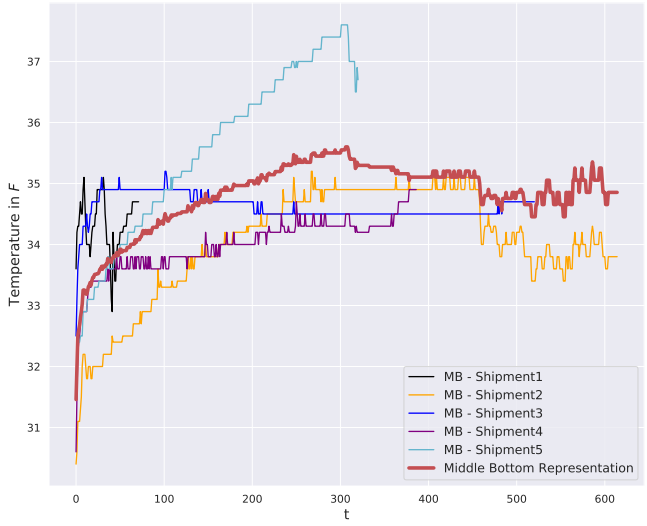

Figure 4.15 Middle-Bottom representation using CW-MAC.

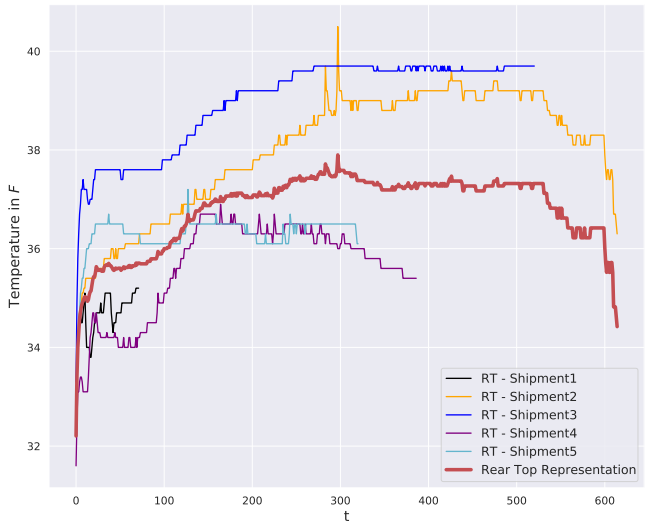

Figure 4.16 Rear-Top representation using CW-MAC.

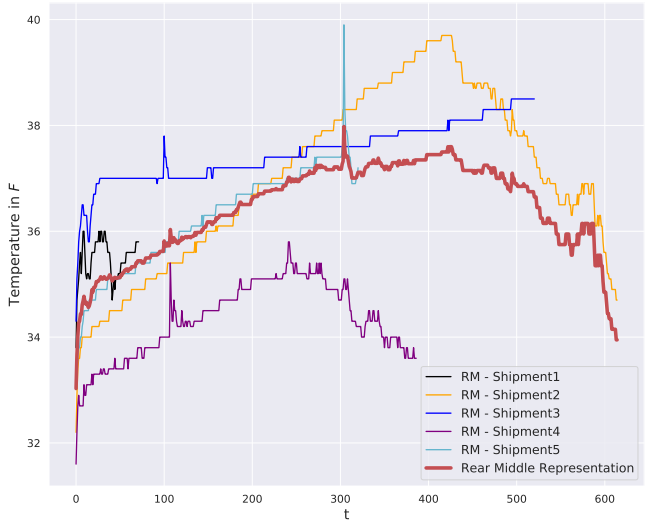

Figure 4.17 Rear-Middle representation using CW-MAC.

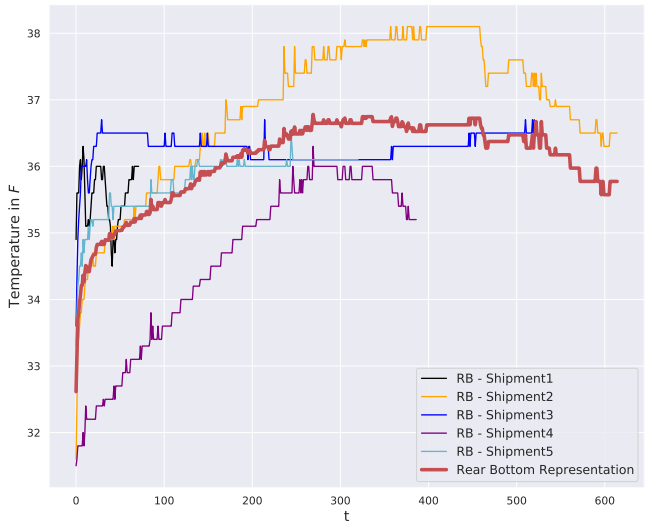

Figure 4.18 Rear-Bottom representation using CW-MAC.

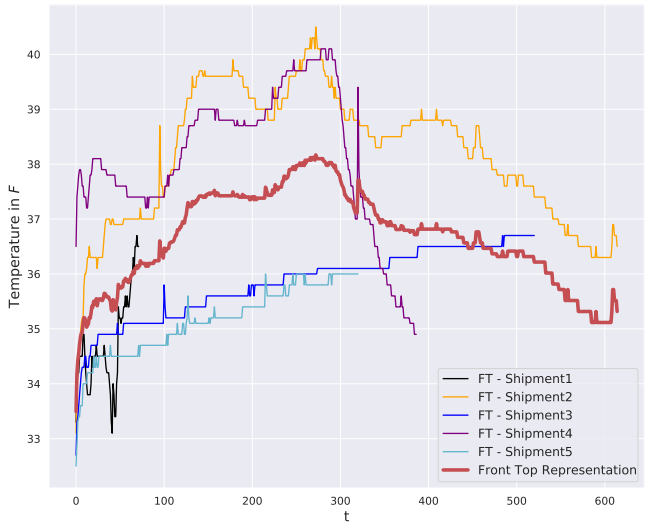

Figure 4.19 Front-Top representation using DTW-MAC.

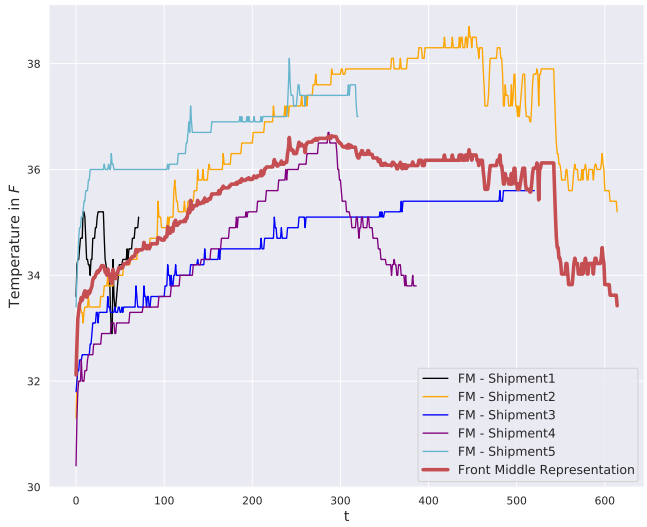

Figure 4.20 Front-Middle representation using DTW-MAC.

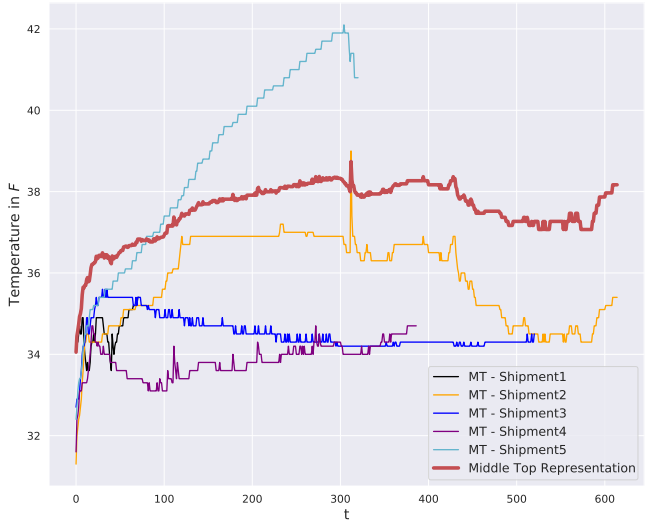

Figure 4.21 Middle-Top representation using DTW-MAC.

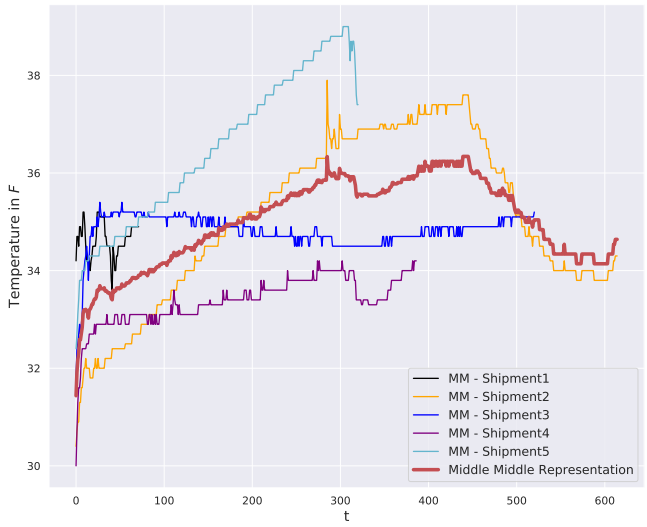

Figure 4.22 Middle-Middle representation using DTW-MAC.

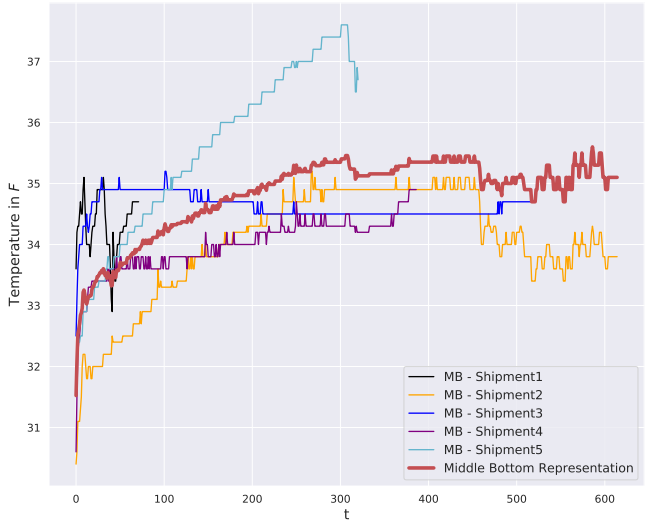

Figure 4.23 Middle-Bottom representation using DTW-MAC.

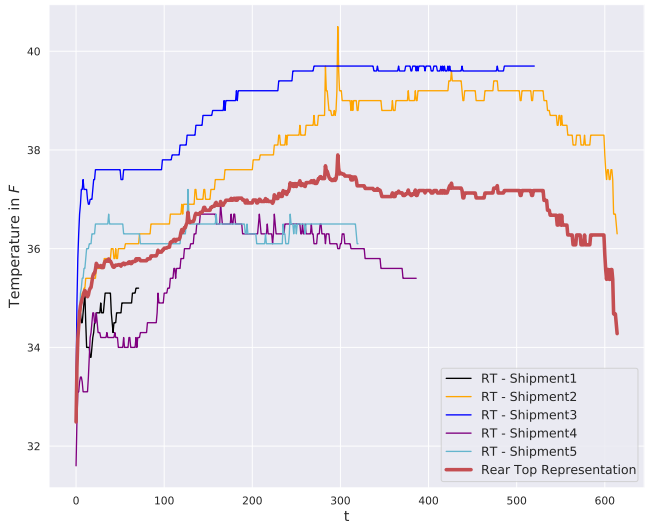

Figure 4.24 Rear-Top representation using DTW-MAC.

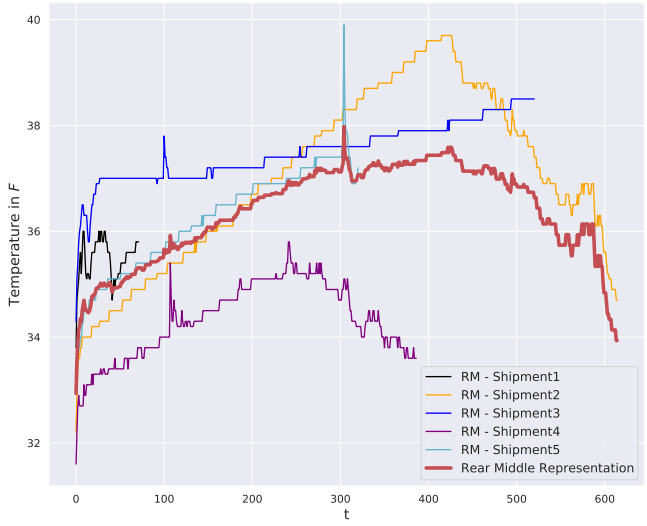

Figure 4.25 Rear-Middle representation using DTW-MAC.

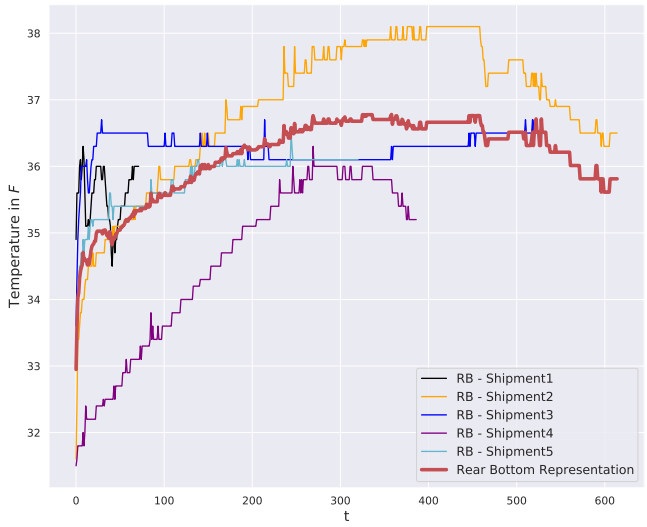

Figure 4.26 Rear-Bottom representation using DTW-MAC.

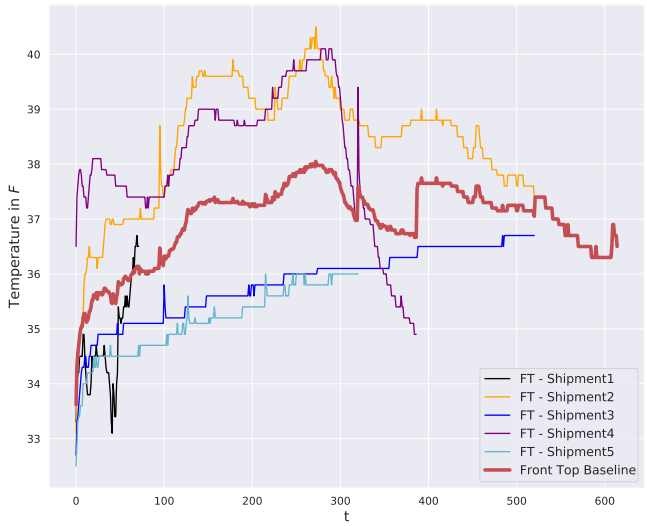

Figure 4.27 Front-Top representation using UW-MAC.

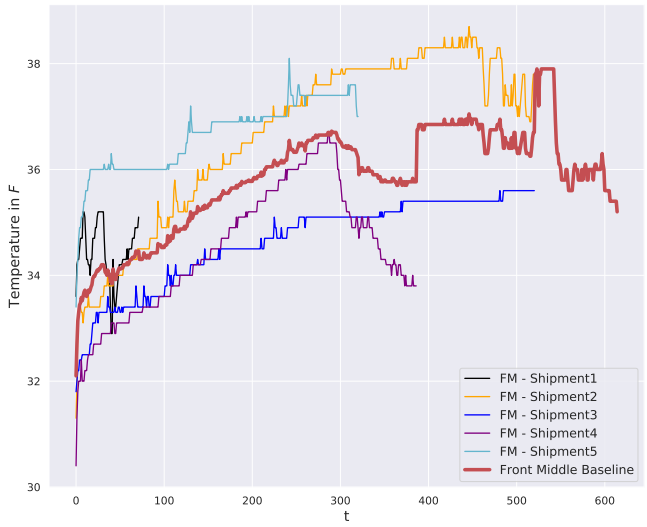

Figure 4.28 Front-Middle representation using UW-MAC.

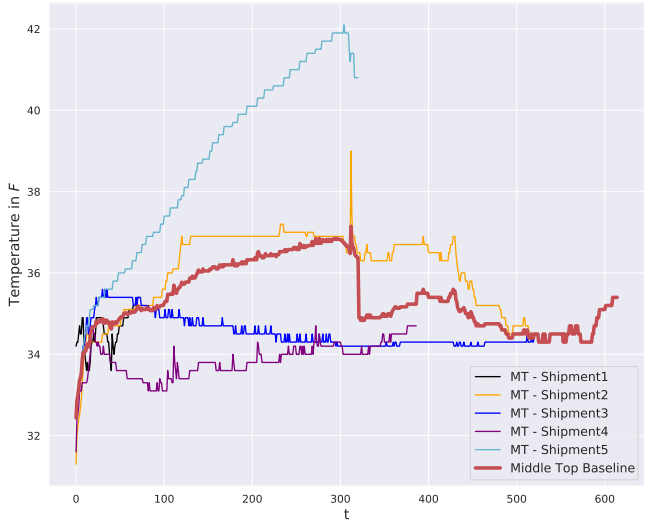

Figure 4.29 Middle-Top representation using UW-MAC.

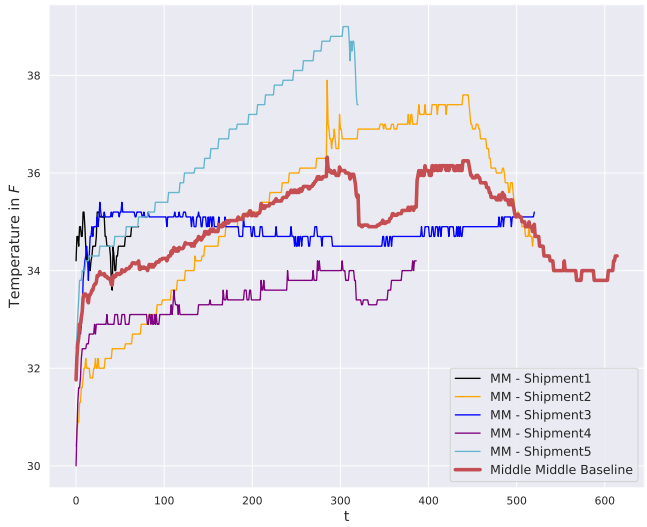

Figure 4.30 Middle-Middle representation using UW-MAC.

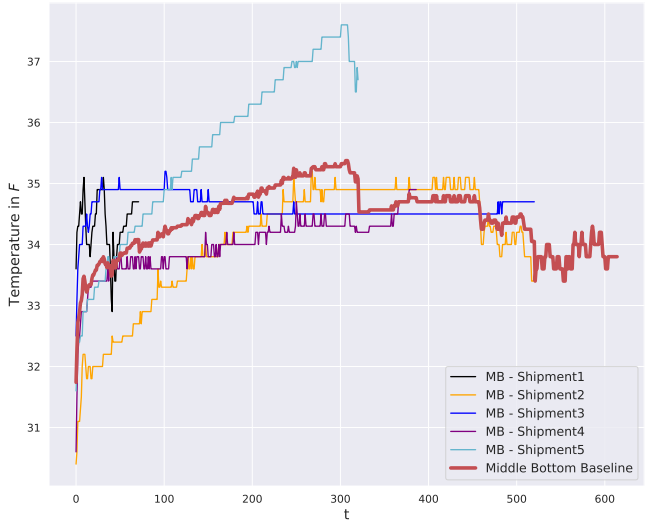

Figure 4.31 Middle-Bottom representation using UW-MAC.

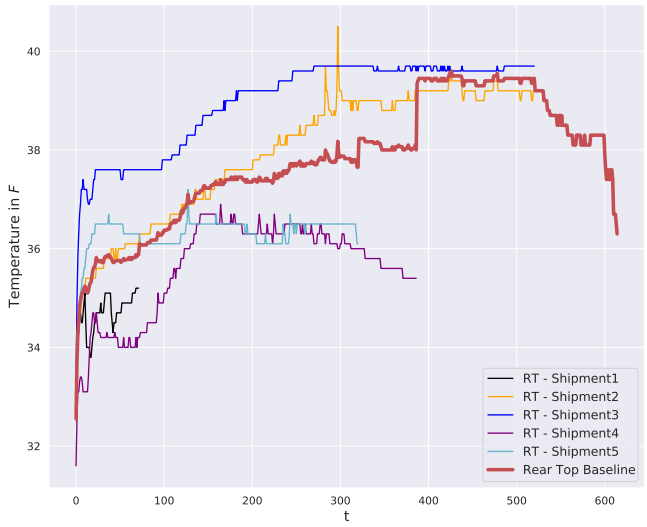

Figure 4.32 Rear-Top representation using UW-MAC.

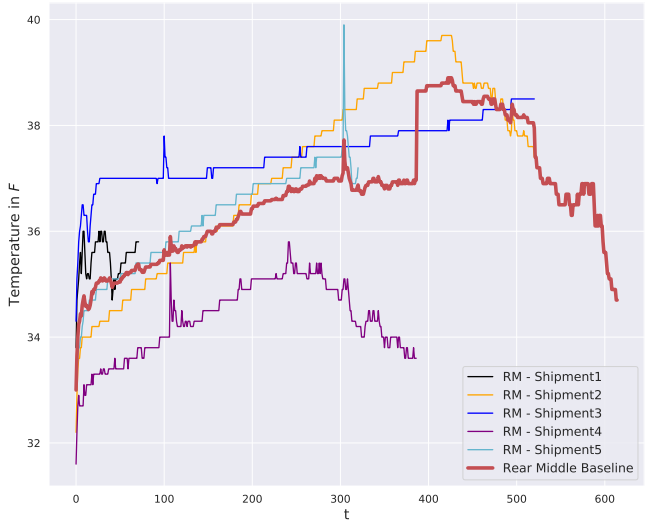

Figure 4.33 Rear-Middle representation using UW-MAC.

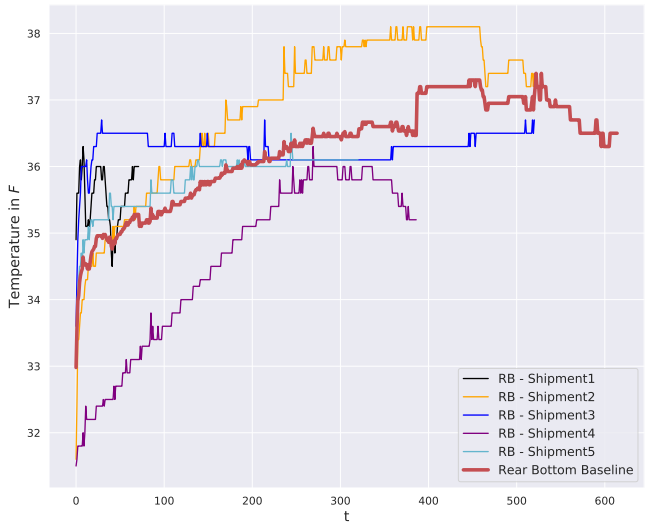

Figure 4.34 Rear-Bottom representation using UW-MAC.

and UW-MAC as functions "Sense2Vec" that takes sensors different sensor profiles and map them into a fixed-length time-series vector representative of all the profiles that are fed into the function. We believe that the differences in representations distributions, such as the means, the standard deviations, and the skewness directions give us more information about the temporal heterogeneity, complexity, similarity, and discrepancy of the collected multivariate time-series without inspecting the sensors' readings individually. Figure 4.35 shows the distributions of all sensors' vectorial representations for all the proposed algorithms.

4.7 Analysis

4.7.1 The Effects of Noise

To test the robustness of our algorithms against noise, an artificial noise has been added to one of the sensor profiles (i.e., the Front-Top sensor) across all shipments (1 through 5). The noise was sampled as a sequence of uncorrelated samples with zero mean and unit variance. Given that the multivariate time-series profiles have different lengths, we created different noise vectors with different lengths equal to the original sensor time-series, and separately added it to each sensor as

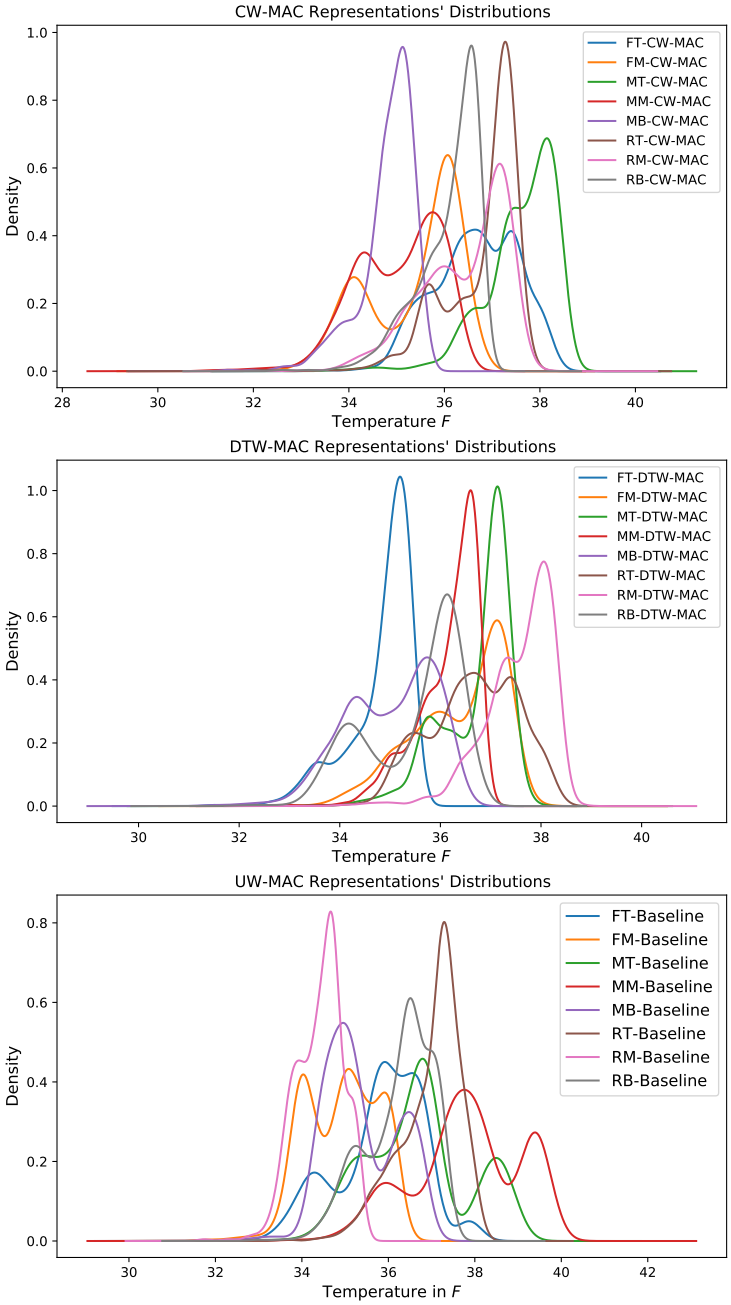

Figure 4.35 The distributions of all representations for CW-MAC, DTW-MAC, and UW-MAC.

we ran the algorithm five times on each Front-Top location across the five shipments. This process is shown by the equations below:

$$\mathbf{X}_{front-top}^{ship1^{n[0]}} = \mathbf{X}_{front-top}^{ship1^{n[0]}} + 0.3 * WN^{n^{[0]}} \tag{4.12}$$

$$\mathbf{X}_{front-top}^{ship2^{n[1]}} = \mathbf{X}_{front-top}^{ship2^{n[1]}} + 0.3 * WN^{n^{[1]}} \tag{4.13}$$

$$\mathbf{X}_{front-top}^{ship3^{n[2]}} = \mathbf{X}_{front-top}^{ship3^{n[2]}} + 0.3 * WN^{n^{[2]}} \tag{4.14}$$

$$\mathbf{X}_{front-top}^{ship4^{n[3]}} = \mathbf{X}_{front-top}^{ship4^{n[3]}} + 0.3 * WN^{n^{[3]}} \tag{4.15}$$

$$\mathbf{X}_{front-top}^{ship5^{n[4]}} = \mathbf{X}_{front-top}^{ship5^{n[4]}} + 0.3 * WN^{n^{[4]}} \tag{4.16}$$

where $n^{[\mathbf{0}]}$ is the length of the shortest Front-Top profile among all the different shipments.

Equations above are generalized and they can be applied to any location, even though we randomly chose the Front-Top location to demonstrate the algorithms' robustness against noise. The results confirm that all three proposed algorithms, CW-MAC, DTW-MAC, and UW-MAC, are robust to noise as the noise power is distributed temporarily across the time-series profiles by the clipping, weighted averaging, and moving operations. Figure 4.36 through 4.38 below show the CW-MAC, DTW-MAC, and UW-MAC representations, respectively. It can been seen clearly from the figures that these representations are similar to the one that was obtained from the profiles without the additional noise.

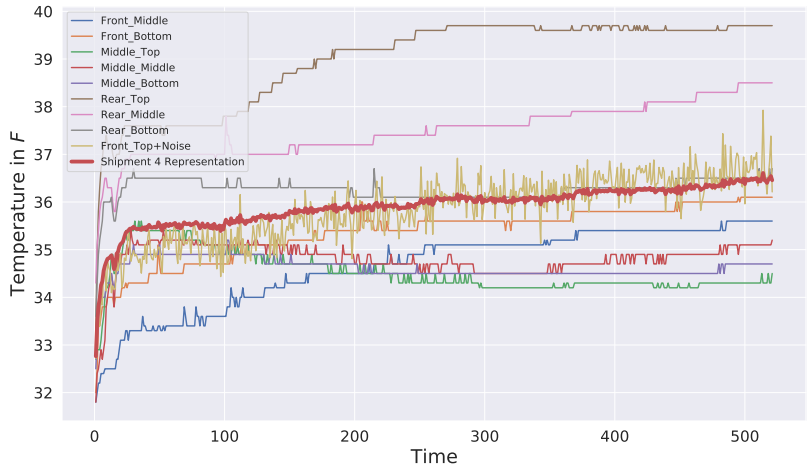

Figure 4.36 Shipment 4 time-series representation using CW-MAC after adding the noise to the Front−Top location.

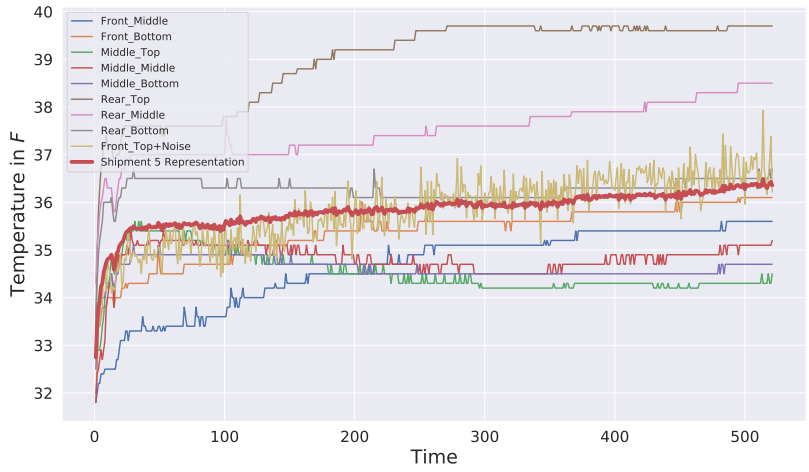

Figure 4.38 Shipment 4 time-series representation using UW-MAC after adding the noise to the Front−Top location.

4.7.2 The Effect of Different Distance Metrics

In the aforementioned subsections, we rely on similarity measures like the Pearson correlation coefficient, and dynamic time warping. However, the algorithm can accommodate other distance metrics to compare different sets of representations. Figure 4.39 through Figure 4.43 be-

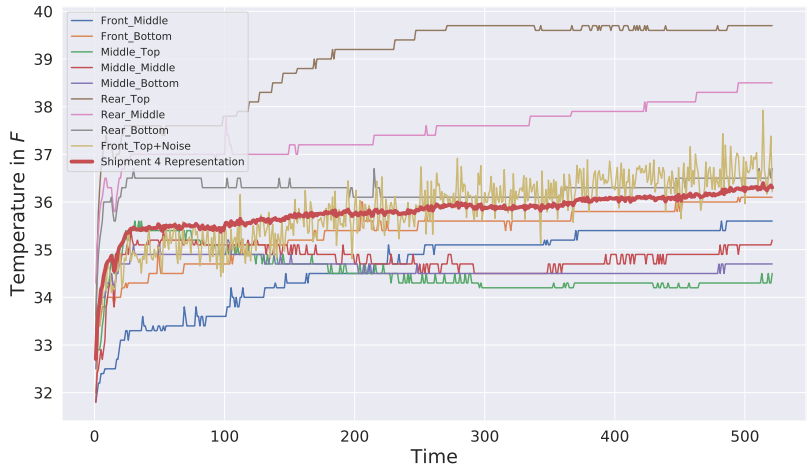

Figure 4.37 Shipment 4 time-series representation using DTW-MAC after adding the noise to the Front−Top location.

low demonstrate the results for shipments 1 through 5. It is clear that the algorithm still outputs a set of representations that are very similar regardless of the distance measure used. The distance between two sensors varies based on the distance selection, but since the values are subsequently normalized between zero and one as part of the algorithm, we find a minor change in the final representations. These results highlight the flexibility of our algorithm as we construct a sequence of steps to allow adopting more similarity measures or different weighing mechanisms, which can be computed based on the various distance measures. We use the following similarity measures to be compared with the DTW-MAC based-distance algorithm: the Euclidean distance $d_{Euclidean}$, the mean absolute error (MAE), the Canberra distance $d_{Canberra}$ [130], the Jeffreys and Matusita distance $d_{Jeffreys-Matusita}$ [131] where a lower value indicates that two time-series profiles are more similar. We also consider the cosine coefficients $Cos_{Similarity}$, which provides a higher value when the temperature readings from two sensors are more similar. The mathematical definitions of these measures are shown below:

$$d_{Euclidean} = \sum_{i=1}^{n^{[j]}} \sqrt{(X_i^j - Y_i^j)^2}; \tag{4.17}$$

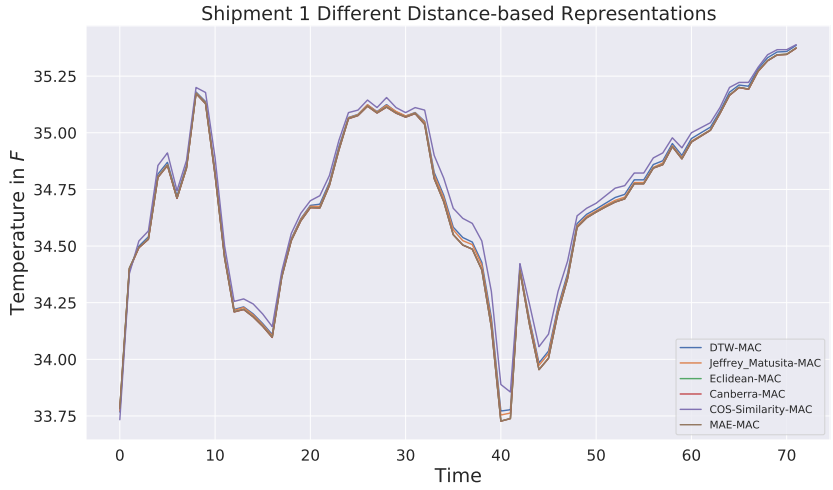

Figure 4.39 Shipment 1: comparing different time-series representations based different distances.

$$MAE = \sum_{i=1}^{n^{[j]}} X_i^j - Y_i^j; \tag{4.18}$$

$$d_{Canberra} = \sum_{i=1}^{n^{[j]}} \frac{X_i^j - Y_i^j}{X_i^j + Y_i^j}; \tag{4.19}$$

$$d_{Jeffreys-Matusita} = \sqrt{\sum_{i=1}^{n^{[j]}} (\sqrt{X_i^j} - \sqrt{Y_i^j})^2}; \tag{4.20}$$

$$Cos_{Similarity} = \frac{\sum_{i=1}^{n^{[j]}} X_i^j Y_i^j}{\sqrt{\sum_{i=1}^{n^{[j]}} (X_i^j)^2} \sqrt{\sum_{i=1}^{n^{[j]}} (Y_i^j)^2}}; \tag{4.21}$$

X_i^j, and Y_i^j denote the multivariate time-series data from two different sensors either in the same shipment or across different shipments. Thus, subscript i indicates the sensor ID (i.e., location) and superscript j indicates the shipment number for that sensor (i.e., first, second, etc.)

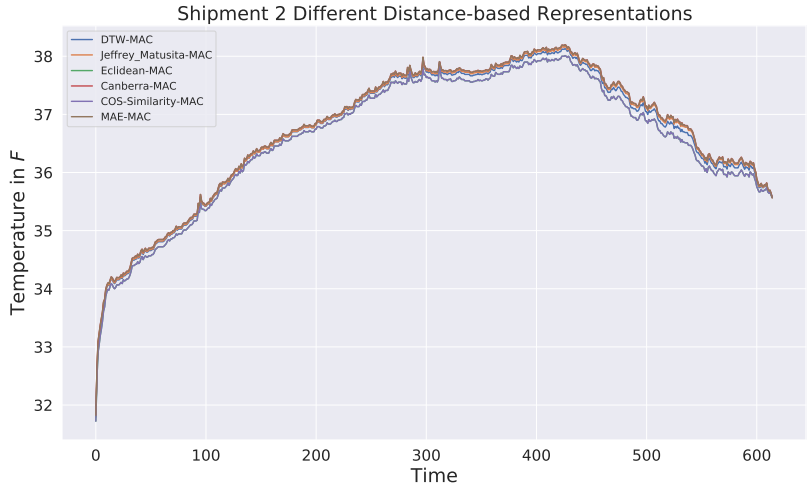

Figure 4.40 Shipment 2: comparing different time-series representations based different distances.

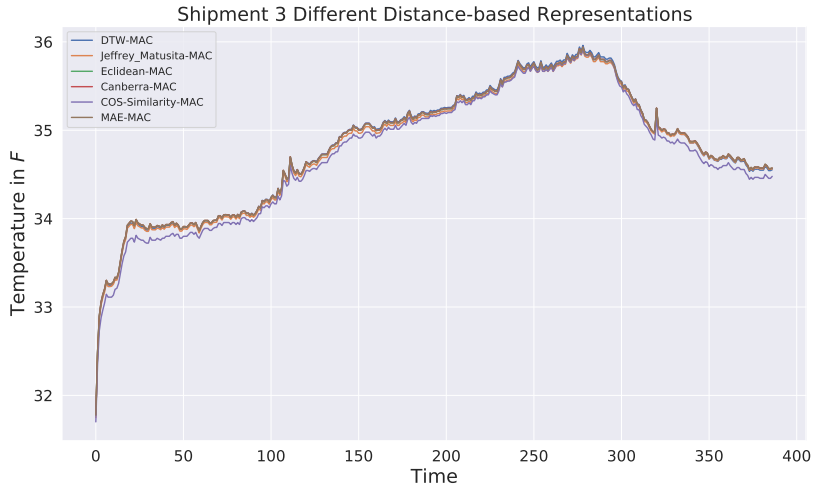

Figure 4.41 Shipment 3: comparing different time-series representations based different distances.

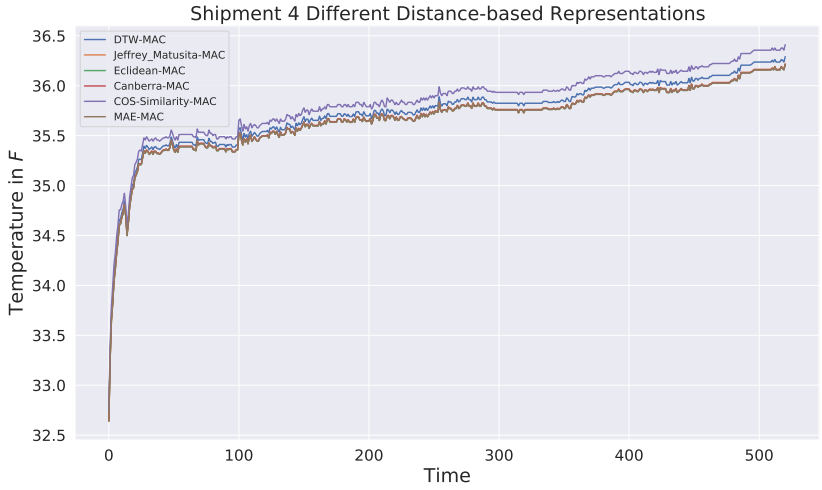

Figure 4.42 Shipment 4: comparing different time-series representations based different distances.

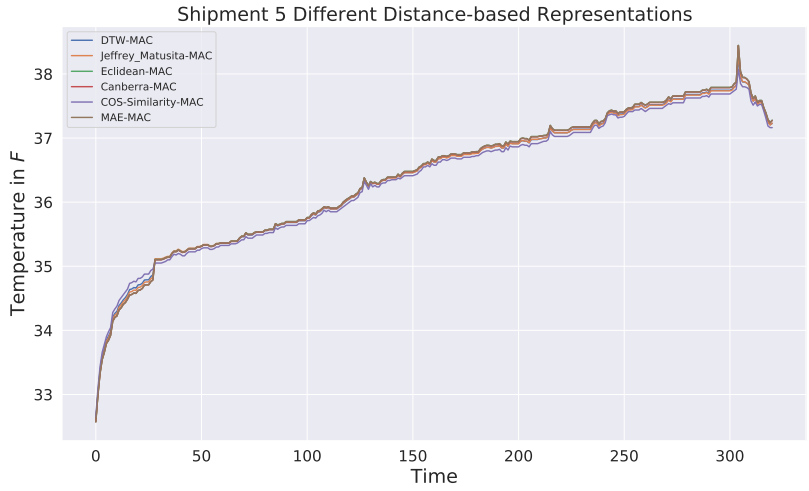

Figure 4.43 Shipment 5: comparing different time-series representations based different distances.

Chapter 5: Class2Vec: Time Series Representation and Classification

5.1 Note to Reader

Portions of this chapter have been previously submitted in our paper at IEEE Transactions on Knowledge and Data Engineering and have been reproduced with the permission from IEEE.

5.2 Summary

In this chapter, we introduce a novel representation method for better visualization and similarity detection in time series classification. Supervised Class2Vec is an application of the CW-MAC algorithm where each class is represented by a single blueprint profile learned from the training dataset. CW-MAC is used to transform multiple observations of univariate or multivariate time series of equal or different lengths into a compact feature-rich representation via clipping, statistical aggregation and concatenation. The Class2Vec algorithm uses dynamic time warping distances between different observations and the class blueprints to create a novel classification framework with an unprecedented compression of the time series training data. We evaluate this framework thoroughly on 70 different datasets including 11 different domains and multiple class distributions, hosted on the UCR Archive as the largest publicly available benchmark repository for time series analysis. Our results show that CW-MAC based Class2Vec outperforms the baseline algorithm with an overall average improvement of 1.53% in accuracy. Finally, the proposed model yields exceptional space savings due to the compression of the training space from N number of observations to K number of classes where $N >> K$ for any practical time-series dataset.

5.3 Introduction

Time series data appears in a wide range of data mining domains and applications ranging from finance (e.g. stock market) [132] to healthcare (e.g. electronic health record (EHR) system) [133] to anomaly detection [134] to audio [135], text [136], speech and human activity recognition [137]. While the analysis of time series ranked as one of the most challenging problems in data mining [116], time series visualization has not been studied enough in the literature [138]. The challenge comes from the fact that it is infeasible to find useful information and insight by displaying all the time series data on the screen due to length and size. Data visualization and summarizing techniques are very crucial for better understanding the data during the analysis phase of building a time series classifier for many applications. Recently, time series representation algorithms have been proposed for visualization. For example, CW-MAC [139] can efficiently summarize and visualize uni-multivariate time series. Although, the majority of previous works is limited to time series classification (TSC) problems where each time series has the same length [[126], [140]], CW-MAC algorithm has specifically developed to handle both equal and different lengths. Data mining and knowledge discovery across different problem types can be facilitated with the help of large time series repositories with additional information such as class labels [141]. UCR/UEA archive [141] was proposed as the largest time series repository to overcome the use of synthetic datasets by the researchers to test their algorithms. UCR/UEA archive has led to the advancement of hundreds of TSC algorithms [140], and increased the quality and the consistency of the evaluation of TSC applications including: Motion, sensor readings, images, spectrographs, ECG, electrical devices, and simulated. One of the most popular and traditional TSC classifiers is the Nearest Neighbor (NN) classifier with Dynamic Time Warping (DTW) distance. The authors of [128] stated that "after an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount." Compared to many other distance measures, feature representations, and specialized machine/ deep learning algorithms proposed for TSC, the DTW combined with 1-NN is remark-

ably not easy to beat [[126], [140], [142], [143], [144]]. However, without dynamic programming techniques, the DTW-1NN can be computationally expensive [126].

Time series classification methods are generally categorized into two main categories: distance and feature-based methods [126]. Under distance-based methods, a distance-based similarity (e.g. . Euclidean distance (ED), and DTW [145],etc) is found between two pairs of time series in the training set to determine if they belong to the same class. For feature-based methods, a statistical or symbolic [126] feature representation is computed and fed into a machine learning classifier that learns from the training data using the feature-based representation. In this work, we combine the two methods by first computing a feature-based statistical representation using a weighted sum of temporal Pearson's coefficient matrices for each class observation in the training set and then applying a distance-based similarity measure using DTW on the testing set to classify unseen time series. The CW-MAC representations satisfy the requirements for a good representation model as stated by [146], where the authors highlight four requirements for developing representation models for both accurate and fast similarity detection in time series:

1. The mapping of the time series must be warping-aware which allow using dynamic time warping for similarity detection.

2. The model should maintain low complexity in reducing the high dimensionality of time series data.

3. The new representation should capture the local features of the series by preserving as much information as possible from the original time series, known as the sensitivity to features.

4. The representation model should be sensitive to input parameters, such as the number of coefficients or symbols. Thus, it is desirable to build parameter-free representation models and dimensionality reduction methods.

The rest of this paper is structured as follows. In subsection 2, we highlight the main contributions of our paper. In subsection 3, we provide the related work from time series representation

and classification. In subsection 4, we provide the problem statement for the supervised Class2Vec algorithm we have implemented. In subsection 5 we describe the data, and the experimentation. In subsections 6, 7 and 8 we present and analyse the results. In subsection 9 we look in detail at the average performance of the proposed algorithm across different problems, datasets, and parameters. In subsection 10, we provide a detailed case study on different length time series. The paper is concluded in subsection 11 where we summarise our findings and discuss the future direction.

5.4 Contributions

The main hypotheses is that the average accuracy of CW-MAC based classifier outperforms the baseline across a wide range of datasets with different problem types, number of classes, length of time series, and training/testing set sizes.

- The contributions of this paper is the validation of the four requirements for developing robust representation models by [146] as follows:
  - The resulted representations are time series on which DTW can be applied for query processing and classification.
  - The resulted representations combine multiple time-series sequences that are similar in behavior, i.e belongs to the same class, but has equal or different lengths into good candidate sequences that can summarize the characteristic of the time series and reduce the size of the data set.
  - The novel application of the Pearson's correlation coefficient is to provide a robust way to summarily visualize the temporal similarities between time series sequences in each class.
  - Supervised CW-MAC does not require optimization to find the normalized weights.
- Class2Vec method generates feature-rich and extremely compact representations while preserving the underlying temporal dependency both within the time series and across different time series for each class label in the training dataset.

5.5 Related Work

Time2Vec introduced in [147] is a vector representation that leverages the Fourier sine series with appropriate frequencies to replace the time axis with Time2Vec. The authors show that Time2Vec can be effective for datasets with long sequences and time horizons, however, Time2Vec with LSTM complicates the optimization and the authors kept this issue as future work. Pearson's correlation coefficient (CW-MAC), Dynamic Time Warping (DTW-MAC), and moving average (UW-MAC) based are compressive representation algorithms that combine multiple related time-series datasets of variable lengths, as first proposed in [139] by leveraging various similarity metrics between different time series temperature profiles. CW-MAC is tested on a novel food transportation dataset [148] that were collected from 6 Continental states in US which consists of temperature recordings from wireless sensor networks implemented on different shipments of perishable commodities across the United States. Similarity/distance measures are generally used in retrieving the right time series for comparison with another one when searching large databases [149]. DTW distance measure [106] has been used in speed processing applications for its time warping advantages. DTW distance can be computed by finding the best alignment "warping path" between two time series despite their lengths and phase mis-matches which is an advantage over the well-known Euclidean distance metric [126]. DTW constructs a contiguous, and monotonic n-by-m matrix from which the best alignment path is computed. The best warping path is the one that minimizes the total cost of aligning the points of time series A with the points of time series B in the constructed matrix. The minimum cost alignment is computed using a dynamic programming algorithm. [143] and [150] have evaluated eight different distance measures on 38 data sets to study many alternatives to DTW like Edit distance with Real Penalty (ERP) [151], Longest Common Subsequence (LCSS) [152], and other measures to demonstrate they provide a negligible difference over the DTW measure performance. An extensive time series classification review by [140] has defined the criteria of four conditions in evaluating the time series classification algorithms' performance based on impact, data, code availability and feasibility. The review shows that three prominent algorithms are chosen based on the aforementioned criteria that can be com-

parable to DTW. Weighted DTW (WDTW) was proposed by [153] as modified version of DTW by including a new multiplicative weight to penalize the distance between points in the warping path. Another useful metric Time Warp Edit Distance (TWED) was introduced by [154] as an elastic metric and it is slightly different from Dynamic Time Warping (with no stiffness), and Euclidean distance with infinite stiffness by adding a control parameter elasticity along the time axis. Another metric for time series, called MSM (move-split-merge) was proposed by [149], where three consecutive operations named move with changing values, a split with elements being duplicated, and merge that combines two consecutive elements into one are applied to transform the original time series. The minimum distance is found based on the cheapest cost of operations among all combinations of the three blocks to transform the time series. The aforementioned methods have been used extensively for TSC as traditional models. With the advance of deep learning [11] in computer vision [6], speech recognition [155] natural language processing (NLP) [156]. Recurrent Neural Networks (RNNs) [2], Long short-term memory (LSTM) [59] and Gated Recurrent Unit (GRU) [60] gained popularity in learning a hierarchical level of representations and discriminative features from raw time series [157]. However, a comprehensive review by [158] shows that poor random initialization of the DNNs' weights and biases can significantly degrade the performance of DNNs. Also, the authors show that while the DNNs outperform the NN-DTW for TSC, their performance are not significantly different than COTE (Collective Of Transformation-based Ensembles) [159] and HIVE-COTE (extended COTE with a Hierarchical Vote system) [160].

5.6 Time Series Classification

5.6.1 Time Series Definitions

A time series $\mathscr{X}_t$ of size n can be defined as a collection of data points measured sequentially over equally spaced time intervals. i.e., $\mathscr{X}_t = (x_1, x_2, ..., x_n)$, where $x_t \in \mathbb{R}$ is an observation at time t. A time series can be univeriate if its observation is recorded over a single variable, and multivariate otherwise. Time series can also be classified as continuous or discrete based on the interval of measurement. Specifically, a time series can be considered continuous if its observa-

tions are measured continuously over a specific time interval, whereas discrete time series would contain observations at equally and discretely spaced time intervals such as minutes, hours and days [[31], [32]]. Time series are represented by graphs where the observations are plotted against the times of such observations. A dataset D = $[(\mathscr{X}_1, Y_1); (\mathscr{X}_2, Y_2); \ldots; (\mathscr{X}_N, Y_N)]$. $(\mathscr{X}_N, Y_N)$ is a collection of pairs $(\mathscr{X}_i, Y_i)$ where $\mathscr{X}_i$ could be a univariate or multivariate time series, and $\mathscr{Y}_i$ is the corresponding label.

The standard definition of the TSC task is to learn the mapping function between the input time series data space and the probability distribution over the output class values (i.e the labels).

5.6.2 Problem Statement

We categorize supervised Class2Vec as a Whole Series classifier, which is defined as comparing two time series by using a distance measure to find the discriminatory features [140]. We define time series classification as the problem of building a classifier to map a collection of labelled training time series into its true labels. In this work, we focus our attention to problems where each time series may or may not have the same lengths.

Given an input dataset $(\mathscr{X}, Y)$, $\mathscr{X}$ consists of different classes of time series signals, and class i has m sequences with equal or different lengths, hence the ordered lengths are $L = (n^{[0]}, n^{[q]}, ..., n^{[k]})$, where where $n^{[0]}$ is the length of the shortest sequence in class i. Class i from the input data $\mathscr{X}$ can be written as $\mathscr{X}_i$ = { $\mathscr{X}_0^{n^{[0]}}$, $\mathscr{X}_1^{n^{[1]}}$, ..., $\mathscr{X}_m^{n^{[k]}}$ }; $\mathscr{X}_m^{n^{[k]}} = (x_0, x_1, x_2, x_3, ..., x_k)$, where superscript indicates the length of each time series in a class i, and subscripts stands for the class number. $\mathscr{X}_m^{n^{[k]}} \in \mathbb{R}^{1\times k}$, and y_i are the class i labels, $y_i \in \mathbb{R}^{m\times 1}$. Please note that none of the time series sequences need to be of the same length as CW-MAC was designed to account for such a condition. The first step for supervised CW-MAC is to condition on the class label y_i. Given a training set $(\mathscr{X}_i, Y_i)$ associated with class i, assuming m sequences within the class, first, we perform ascending ordering on the time-series signals $\mathscr{X}_i$, for $i = 1, 2, 3, \ldots, m$, with superscripts 1 and m for the shortest and longest sequence respectively: $(\mathscr{X}_i^{[1]}, y_1)$; $(\mathscr{X}_i^{[2]}, y_2)$; $(\mathscr{X}_i^{[3]}, y_3)$; ...; $(\mathscr{X}_i^{[m]}, y_m)$. Then, a moving clipping mechanism is used to create uniform sets

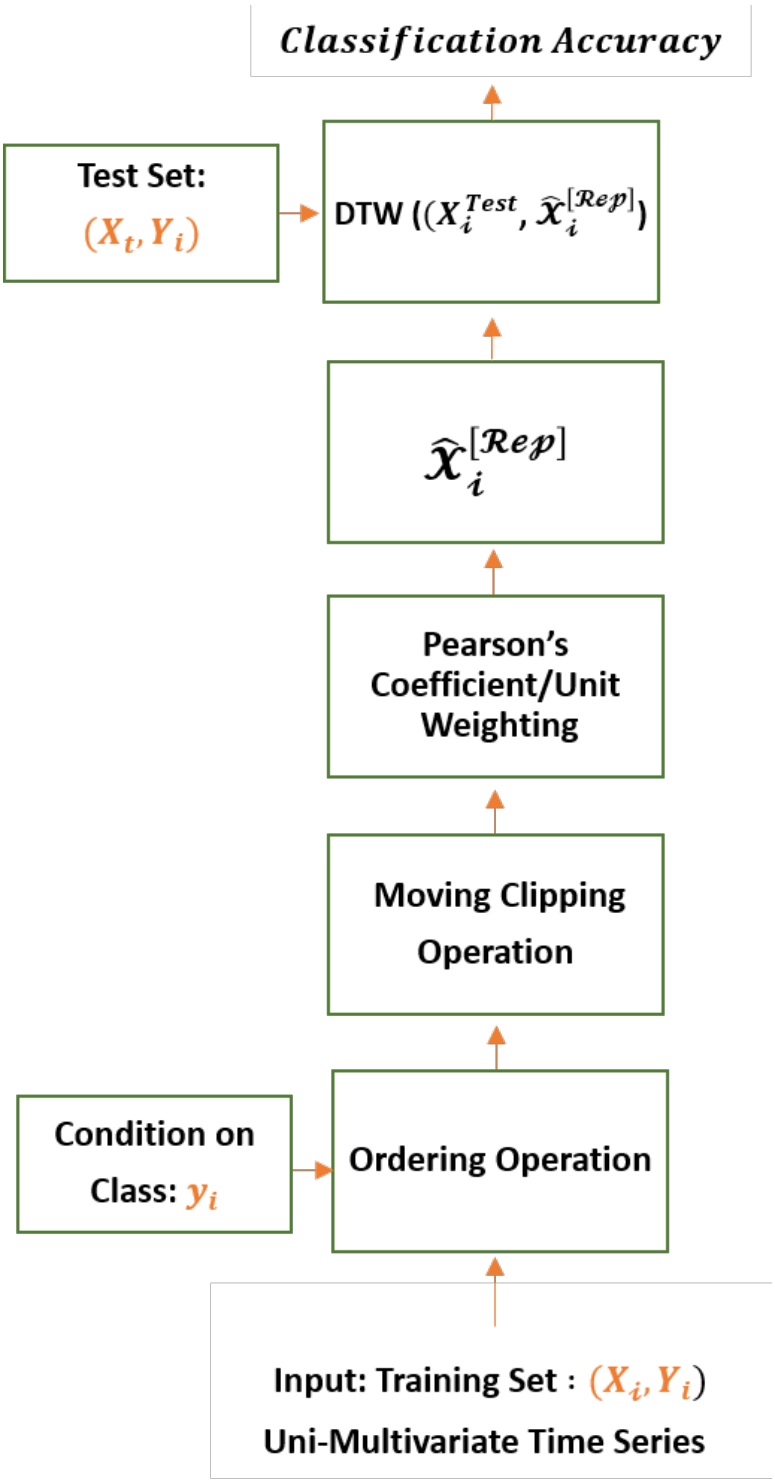

Figure 5.1 Time-series classification pipeline based on CW-MAC representation.

of disjoint sequences across multiple groups to calculate a normalized similarity metric. We compute m temporal Pearson's Correlation matrices starting by clipping the length of all $m-1$ signals to have the same length as the shortest signal. Then, we compute $k \times k$ cross-correlation matrix where σ_{ij} represents the Pearson's correlation coefficient between time-series i and time-series j. Pearson's product-moment correlation coefficients can be written according to [105] as: $\sigma_{xy} = \frac{\sum_{i=1}^{n^{[0]}}(x_i-\bar{x_i})(y_i-\bar{y_i})}{\sqrt{\sum_{i=1}^{n^{[0]}}(x_i-\bar{x_i})^2}\sqrt{\sum_{i=1}^{n^{[0]}}(y_i-\bar{y_i})^2}}$, where $\bar{x_i}$ is the average means of the ordered time-series signals of length $n^{[0]}$, and $n^{[0]}$ is the length of the shortest sequence in the group for the first correlation matrix which includes all m signals. Next, for each Pearson's correlation matrix, we compute normalized weight coefficients for each signal as follows:

$$\mathscr{W}_i^{[p]} = \frac{\sum_{j=1}^{m}\sigma_{(p+1)j} - 1}{(\sum_{i=1}^{m}(\sum_{j=1}^{m}\sigma_{ij} - 1))} \tag{5.1}$$

where $p = 1, 2, \ldots m$ represents the normalized weight coefficients for the first, second, $\ldots m^{th}$ time-series signals, respectively. Then, a weighted fusion and concatenation to create a representative vector for each group in class is performed as follows:

$$\hat{\mathscr{X}}_{\mathbf{class_i}} = \begin{bmatrix} \mathscr{X}_i[1][0{:}n^{[0]}] & \mathscr{X}_i[2][0{:}n^{[0]}] & \cdots & \mathscr{X}_i[m][0{:}n^{[0]}] \\ 0 & \mathscr{X}_i[2][n^{[0]}{:}n^{[1]}] & \cdots & \mathscr{X}_i[m][n^{[0]}{:}n^{[1]}] \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \mathscr{X}_i[m]n^{[m-1:m]} \end{bmatrix} \tag{5.2}$$

$$\mathbf{W} = \begin{bmatrix} \mathscr{W}_m^{[1]} & \mathscr{W}_m^{[2]} & \cdots & \mathscr{W}_m^{[m]} \\ 0 & \mathscr{W}_{m-1}^{[2]} & \cdots & \mathscr{W}_{m-1}^{[m]} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \mathscr{W}_1^{[m]} \end{bmatrix} \tag{5.3}$$

$$\mathscr{X}_{\mathbf{igroups}} = \begin{bmatrix} \hat{\mathscr{X}}^{[1]}_{group_1/y_i} \\ \hat{\mathscr{X}}^{[2]}_{group_2/y_i} \\ \vdots \\ \hat{\mathscr{X}}^{[m]}_{group_m/y_i} \end{bmatrix} \tag{5.4}$$

where that $\mathscr{W}_1^{[m]} = 1$. and $\mathscr{X}_i^{[m]n^{[m-1:m]}}$ is the remaining records of the longest sequence.

$$\mathscr{X}_{\mathbf{groups}} = \mathbf{W} \circ \hat{\mathscr{X}}_{\mathbf{class_i}} \tag{5.5}$$

The final compound signal for the first group can be obtained by concatenating vertically all the $\hat{\mathscr{X}}^{[i]}_{group_j/y_i}$ to form the one time-series signal that is the best representation for all the individual time-series signals.

$$\hat{\mathscr{X}}_i^{[Representative]} = [\hat{\mathscr{X}}^{[1]}_{group_1/y_i}, \hat{\mathscr{X}}^{[2]}_{group_2/y_i}, \ldots, \hat{\mathscr{X}}^{[m]}_{group_m/y_i}] \tag{5.6}$$

The number of representations in $\hat{\mathscr{X}}^{[Representative]}$ is equal to the number of classes in the dataset. Finally, we test the quality of the representations on the test set. We apply query processing where the input representation for class i needs to be matched to a large number of time-series sequences in the test set. To retrieve the class i from the test set using the proposed representations, we apply DTW as a measure of comparing the time-series sequences in the test set to the class representations. DTW warps the time axis of one (or both) sequences to achieve a better alignment.

To align two sequences, DTW constructs an $nx_i \times nx_j$ matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $d(\mathscr{X}_i, \mathscr{X}_j)$ between the two time series. $\mathscr{X}_i$, and $\mathscr{X}_j$ (typically the Euclidean distance is used, so $d(x_i, y_j) = \sqrt{\sum(\mathscr{X}_i - \mathscr{X}_j)^2}$). DTW warping path is subject to three

constraints according to [105]: boundary conditions, continuity and monotonicity; to minimize the overall warping cost as follows:

$$DTW(x,y) = min\left\{\frac{\sqrt{\sum_{l=1}^{L}(z_i)}}{L}\right\} \tag{5.7}$$

where L is used to compensate for the fact that warping paths may have different lengths. In order to find the minimum path, the warping path Z is contiguous: $Z = z_1, z_2, \ldots, z_L$ and $max(nx_i, nx_j) \leqslant L < (nx_i + nx_j\text{-}1)$. DTW uses dynamic programming to compute the cumulative distance $\zeta(i,j)$ from $d(i,j)$ "current position" in the matrix and the minimum of the cumulative distances of the adjacent elements:

$$\zeta(i,j) = d(\mathscr{X}_i, \mathscr{X}_j) + min\{\zeta(i-1,j-1) + \zeta(i-1,j) + \zeta(i,j-1)\tag{5.8}$$

5.7 Data and Experimental Design

5.7.1 Methodology

We evaluate our developed framework thoroughly on the largest publicly available benchmark for time series analysis: UCR archive time series datasets [141] have been widely established as a standard for evaluating time series classification algorithms. The time series in the archive are already z-normalized to have a zero mean and unity standard deviation. These datasets cover a wide spectrum of different problems, applications, number of classes, length of time series, as well as sizes of training and testing sets. Figure 5.2 (a) presents the distributions of problem types/domains, (b) classes, (c) time series lengths, and training and testing set frequencies on 70 datasets from the UCR repository. Many of these datasets were created at different places, and laboratories, and hence the archive is large and diverse. We used the default training and testing set splits provided by UCR. Readers can find the detailed description of the 70 datasets on (http://timeseriesclassification.com). The selected collection is varied in terms of data characteristics: the

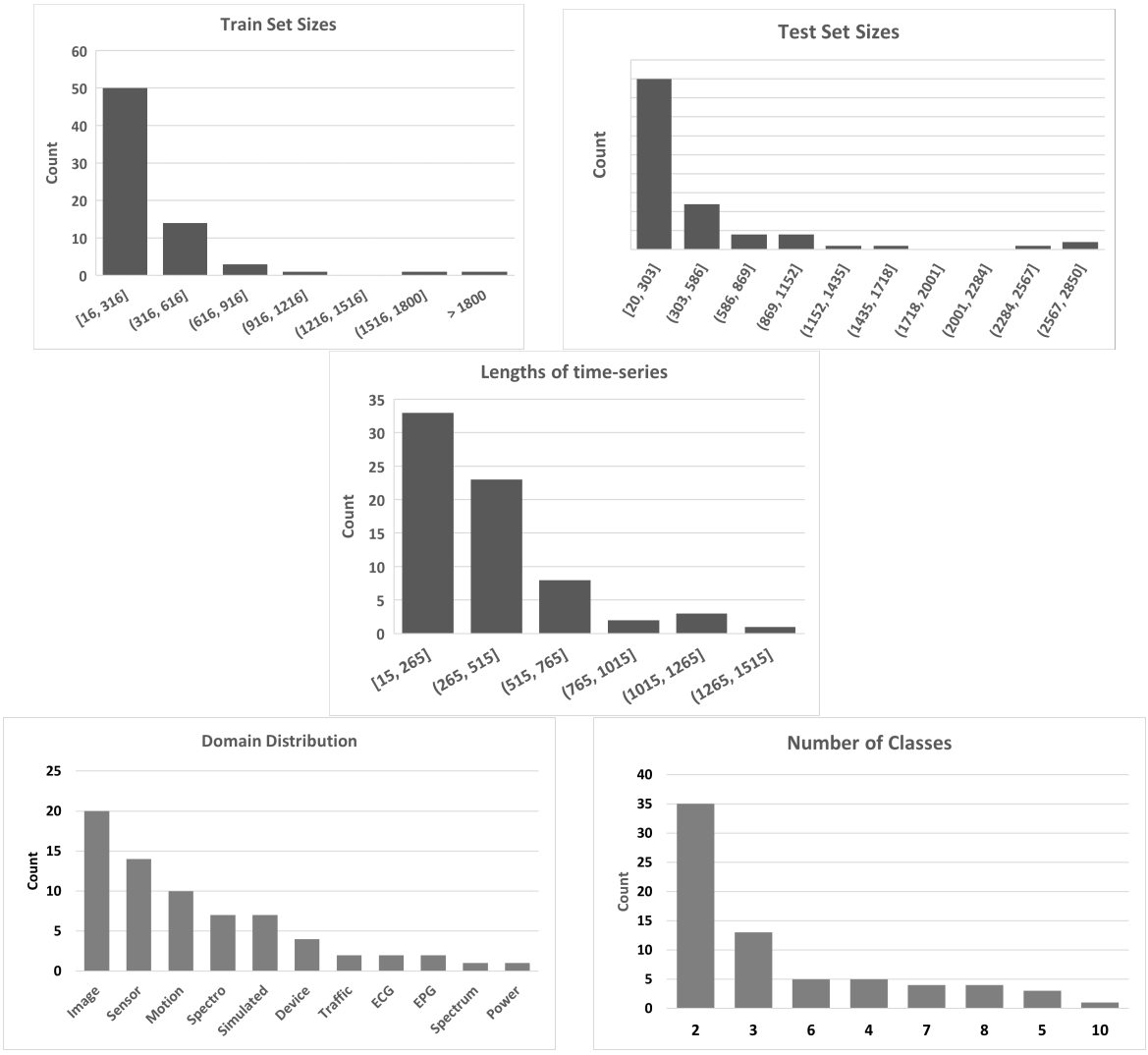

Figure 5.2 Summary information for the 70 datasets from the UCR repository used in the study.

lengths of the time series ranges from 15 (SmoothSubspace) to 1500 timestamps (SemgHandGenderCh2); train set sizes vary from 16 to 8926 observations; test set sizes vary from 20 to 2850 observations, and the number of classes is between 2 and 10. The bar graphs in Figure 5.2 show more detailed histograms. There are a large number of datasets with small and median training set sizes (twenty-six have 50 or fewer training examples, dozen have less than 100, and twenty-three between 100 and 500, and ten have greater than 500 observations). Image and sensor classification problems are the majority of data domains. Figure 5.2 gives the summary of domain distributions. We implemented DTW using the simple dynamic programming algorithm. Since our focus was not on computation performance, we did not use any faster tricks or algorithms to speed up the timing. We applied window-based constraints (Sakoe-Chiba band [161] in our experimentation) on the warping path so that the distance the path is allowed to drift is not deviated far from the constructed matrix diagonal. According to [162], the window size w can non-intuitively improve the classification accuracy of same/different length time series and prevent pathological warping, "where a relatively small subsection of one sequence maps to a much larger subsection of another". It should be noted that when the w is equal to zero then the best warping path is the diagonal of the DTW constructed matrix, and in that case DTW with $w = 0$ becomes the Euclidean distance. We used different window-sizes w based on experimentation and simple grid search to find the best w for DTW computations for each dataset. The parameter values we search for the window size w are shown in Table 3.5. We run the same conditions and seed on each problem for both CW-MAC and baseline classifiers. We use the same training and testing sets as were provided in the archive, however, for a few datasets that have a large number of observations and longer lengths, we choose a smaller subset of observations as computing the DTW on the entire testing set is computationally very expensive.

Table 5.1 Best performing accuracy split by domain type.

Domain Type	CW-MAC Accuracy	Baseline Accuracy
Device	0.513	0.416
ECG	0.767	0.789
EPG	1.000	1.000
Image	0.658	0.657
Motion	0.658	0.661
Power	0.900	0.911
Sensor	0.764	0.726
Simulated	0.702	0.705
Spectro	0.728	0.728
Spectrum	0.527	0.545
Traffic	0.789	0.780
Total Average	0.701	0.689

Table 5.2 Best performing accuracy split by number of classes.

Number of Classes	CW-MAC Accuracy	Baseline Accuracy
2	0.706	0.687
3	0.711	0.710
4	0.708	0.698
5	0.656	0.656
6	0.648	0.661
7	0.675	0.629
8	0.744	0.745
10	0.675	0.655
Total Average	0.701	0.689

Table 5.3 Average accuracy of CW-MAC and UW-MAC (baseline) over 70 problems.

No	Problem Type	Dataset	Train	Test	Length	Class	W	CW-MAC	Baseline
1	Image	ArrowHead	36	175	251	3	2	0.611	0.623
2	Spectro	Beef	30	30	470	5	4	0.533	0.533
3	Image	BeetleFly	20	20	512	2	8	0.700	0.750
4	Image	BirdChicken	20	20	512	2	64	0.650	0.550
5	Simulated	BME	30	150	128	3	16	0.640	0.640
6	Sensor	Car	60	60	577	4	32	0.633	0.617
7	Simulated	CBF	30	900	128	3	2	0.754	0.774
8	Traffic	Chinatown	20	343	24	2	2	0.904	0.904
9	Spectro	Coffee	28	28	286	2	64	0.929	0.929
10	Device	Computers	250	250	720	2	2	0.648	0.496
11	Image	DiatomSizeRed	16	306	345	4	32	0.722	0.801
12	Image	DistalPhalanxOAG	400	139	80	3	2	0.705	0.705
13	Image	DistalPhalanxOC	600	276	80	2	2	0.685	0.674
14	Image	DistalPhalanxTW	400	139	80	6	2	0.561	0.576
15	Sensor	DodgerLoopGame	20	138	288	2	128	0.616	0.551
16	Sensor	DodgerLoopW	20	138	288	2	128	0.957	0.957
17	Sensor	Earthquakes	322	139	512	2	32	0.748	0.252
18	ECG	ECG200	100	100	96	2	8	0.780	0.790
19	ECG	ECGFiveDays	23	861	136	2	2	0.754	0.787
20	Device	ElectricDevices	8926	1550	96	7	32	0.426	0.227
21	Image	FaceFour	24	88	350	4	32	0.727	0.648
22	Image	Fish	175	175	463	7	8	0.629	0.617
23	Sensor	FreezerRegular	150	2850	301	2	32	0.760	0.759
24	Sensor	FreezerSmall	28	2850	301	2	32	0.746	0.746
25	Motion	GunPoint	50	150	150	2	2	0.733	0.733
26	Motion	GunPointAgeSpan	135	316	150	2	2	0.646	0.576
27	Motion	GunPointMVF	135	316	150	2	2	0.617	0.617
28	Motion	GunPointOVY	136	315	150	2	2	0.889	0.886
29	Spectro	Ham	109	105	431	2	2	0.771	0.771
30	Image	Herring	64	64	512	2	8	0.563	0.563
31	EPG	InsectEPGRT	62	249	601	3	2	1.000	1.000
32	EPG	InsectEPGST	17	249	601	3	2	1.000	1.000
33	Sensor	ItalyPowerDemand	67	1029	24	2	4	0.899	0.826
34	Sensor	Lightning2	60	61	637	2	64	0.607	0.738
35	Sensor	Lightning7	70	73	319	7	16	0.685	0.699
36	Simulated	Mallat	55	320	1024	8	32	0.959	0.959
37	Spectro	Meat	60	60	448	3	2	0.867	0.867
38	Traffic	MelbourneP	1194	2439	24	10	0	0.675	0.655
39	Image	MiddlePOAG	400	154	80	3	2	0.571	0.571
40	Image	MiddlePOC	600	291	80	2	16	0.564	0.570

Table 5.3 (Continued)

No	Problem Type	Dataset	Train	Test	Length	Class	W	CW-MAC	Baseline
41	Image	MiddlePTW	399	154	80	6	2	0.487	0.494
42	Image	MixedSR	500	250	1024	5	64	0.716	0.712
43	Image	MixedSS	100	250	1024	5	64	0.720	0.724
44	Sensor	MoteStrain	20	1252	84	2	8	0.862	0.841
45	Spectro	OliveOil	30	30	570	4	2	0.867	0.867
46	Image	PhalangesOC	1800	858	80	2	2	0.629	0.632
47	Sensor	Plane	105	105	144	7	2	0.962	0.971
48	Power	PowerCons	180	180	144	2	2	0.900	0.911
49	Image	ProximalPOAG	400	205	80	3	2	0.771	0.771
50	Image	ProximalPOC	600	291	80	2	4	0.639	0.639
51	Image	ProximalPTW	400	205	80	6	2	0.620	0.620
52	Device	RefrigerationD	375	375	720	3	2	0.419	0.320
53	Spectrum	SemgHandG-Ch2	300	600	1500	2	32	0.527	0.545
54	Simulated	ShapeletSim	20	180	500	2	16	0.533	0.506
55	Device	SmallKitchenA	375	375	720	3	2	0.560	0.621
56	Simulated	SmoothSubspace	150	150	15	3	2	0.833	0.820
57	Sensor	SonyAIBORS1	20	601	70	2	8	0.867	0.879
58	Sensor	SonyAIBORS2	27	953	65	2	4	0.769	0.767
59	Spectro	Strawberry	613	370	235	2	32	0.576	0.573
60	Image	Symbols	25	995	398	6	16	0.885	0.899
61	Simulated	SyntheticControl	300	300	60	6	8	0.687	0.720
62	Motion	ToeSegmentation1	40	228	277	2	2	0.474	0.583
63	Motion	ToeSegmentation2	36	130	343	2	128	0.631	0.654
64	Sensor	Trace	100	100	275	4	2	0.590	0.560
65	Simulated	UMD	36	144	150	3	16	0.507	0.514
66	Motion	UWaveGestureLAll	896	320	945	8	32	0.844	0.828
67	Motion	UWaveGestureLX	896	320	315	8	32	0.609	0.625
68	Motion	UWaveGestureLY	896	320	315	8	32	0.563	0.566
69	Spectro	Wine	57	54	234	2	2	0.556	0.556
70	Motion	WormsTwoClass	181	77	900	2	2	0.571	0.545

5.7.2 Results and Discussion

We present the results of all the datasets in Table **??**. Both CW-MAC-based classifiers and UW-MAC (i.e. baseline) perform well on a total of 52 out of 70 datasets with classification accuracy greater than 60%. Within the 52 datasets, CW-MAC has an accuracy greater than 70%, 80% and 90% on 13, 9 and 6 datasets, respectively with 100% on 2 datasets. This result

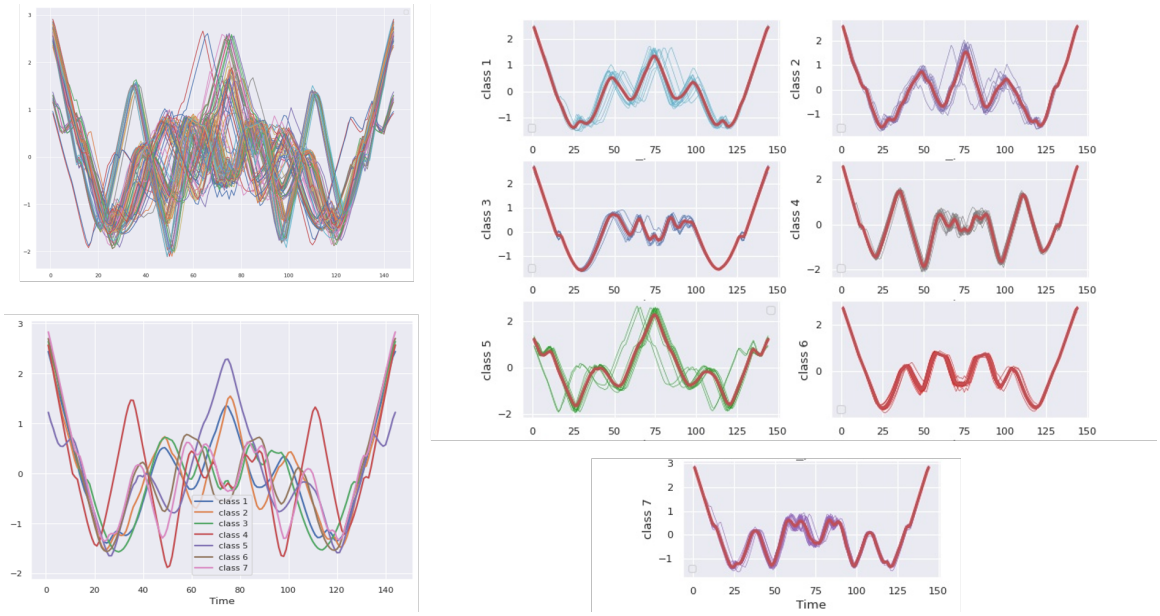

Figure 5.3 CW-MAC representations for Plane training dataset.

verifes the effectiveness of the CW-MAC algorithm in different problem domains. They also indicate that the CW-MAC algorithm captures the dynamics and statistics of the underlying processes. CW-MAC-based classifier mainly considers finding representations from a relatively simple combination of the time series within the same class based on their statistical significance quantified by the weighted Pearson's correlation coefficients. CW-MAC and UW-MAC-based classifiers do not perform well on 18 datasets with accuracies between 60% and 42% for CW-MAC, and between 60% and 32% for UW-MAC. The overall average accuracy difference between the CW-MAC-based classifier and the baseline for the domains: device, image, sensor, spectro, and Ttaffic is 2.93%, which highlights the average improvement of utilizing the CW-MAC time series representations with normalized Pearson's weights over the averaging baseline with unity weights. On the other hand, the baseline achieves a higher accuracy of 2.18% for ECG, and 0.28% for spectrum problem types, and average difference of 1.15% for the following problem types: ECG, Motion, Power, Simulated, and Spectrum. For datasets with 2, and 4 classes and type Sensors, 6 datasets (ItalyPowerDemand, MoteStrain, Earthquakes, Car, DodgerLoopGame, Trace) have a combined number of 3307 time series and an average length of 293 timestamps. The CW-MAC has significant improvements over the baseline on six datasets: Earthquakes, ElectricDevices, Computers,

BirdChicken, RefrigerationDevices, and FaceFour. Earthquakes dataset has the second-longest series, and by itself, CW-MAC has an overall improvement of approximately 50% in accuracy over the baseline. A possible reason is that UW-MAC can not learn properly the underlying structures and unique patterns of the time series for each class especially if the time series have unequal lengths. Besides, it has limited capacities in learning complicated characteristics by naive averaging. In our experiments, UW-MAC tends to generate simple representations,resulting in low accuracies results for datasets ElectricDevices, and RefrigerationDevices. For Device problem type, the average overall improvement of CW-MAC over the baseline is 38.80% on Computers, RefrigerationDevices, and ElectricDevices with 2,3, and 7 classes respectively. An average 1.9% improvement is observed on image type datasets: FaceFour, MixedShapesRegularTrain, DistalPhalanxOutlineCorrect, BirdChicken, and Fish. The maximum and the minimum improvement in the accuracies by CW-MAC-based classifier over the baseline is 4.68%, and 0.1% for datasets with 7 and 3 classes, respectively. The overall average difference between the two classifiers for the following: 2,3,4,7, and 10 classes is 1.91%, whereas the overall average difference between the two classifiers for the following 6 and 8 classes is insignificant. By summarizing these results, we can see that CW-MAC-based classifier is competitive in solving the time series classification tasks across different domains. Since only one representation per class is found from the training set and these representations are utilized in the testing time. We are able to achieve unprecedented gains in data compression and complexity for general time series classification. Besides, CW-MAC is able to improve the performance by leveraging the powerful normalized Pearson's weights capability. Our proposed classifier can combine both the benefits of data compression and feature representations, and hence it is expected to perform best when the representations are concatenated with the training set. Finally, graphical results from Figure 5.3 show that the algorithm facilitates discovering similarities across different classes to find proper representations while reducing the time-series dimensionality for back-end applications like classification.

Chapter 6: Open Research Topics

6.1 Future Research

The contribution of this research is not just an algorithm that works well on time series classification and outperforms the baseline, but an end-to-end pipeline that can also be easily combined with other machine learning-based algorithms, and representations methods when used as features to further improve the classification performance. Our method can be easily extended to be used in combination with other distance-based methods by simply finding distance-based representations [139] including the Euclidean distance [145], the mean absolute error, the Canberra distance [130], the Jeffreys and Matusita distance [131], and Cosine similarity. Then these representations could be added as additional features to the machine learning classifiers to improve their performance. This work considers classifying 70 datasets across 11 domains, and these domains are unique by some particular patterns of time series from which domain-specific features could be extracted or learned such that it could help the classification. For example, Electrocardiogram (ECG) time series representations could help in capturing the rising and falling patterns of heartbeats which could potentially help in classifying normal heartbeats from abnormal ones. Using the representations for time series clustering or for searching time series databases in the frequency domain is another possibility of future work. Finally, getting more insights into the circumstances under which CW-MAC-based classifier outperforms other machine learning classifiers and under which factors it lags, will also be promising future work.

6.2 Future Research Directions

The contributions of this research can be used to advance knowledge in the time-series representation learning domain on several future research trajectories as summarized below:

- Deep learning: Building an intelligent layer on top of CW-MAC, DTW-MAC, and UW-MAC to estimate the optimum number of required sensors that can provide reliable temperature monitoring inside a shipping container (or any multi-sensory monitoring application). Another variation of this research venue is to build multiple CW-MAC, DTW-MAC, and UW-MAC layers and create a shared point of information among them via a smart network to select the locations of the sensors that provide the best mapping of location−based prediction. For this purpose, multiple methods can be used including deep learning sequence networks such as recurrent neural networks (RNNs), long-short-term memory networks (LSTMs), attention networks, transformers, and the state-of-the-art sequence to sequence models.

- Cost of sensor implementation: By reducing the number of sensors required for complete cold-chain cycle monitoring, the implementation costs can be reduced significantly especially for large retailers in the specific case of a food transportation application.

- Food quality prediction: Build novel algorithms based on CW-MAC, DTW-MAC, and UW-MAC layers to predict food quality and shelf-life using the representative time-series profile for the whole shipment.

- Time-space: In this work, we only looked at the time dimension whereas an interesting area of research is to include the space dimension, and model both time and space dimension within the time-space-series modeling framework.

- Classification: Utilize the learned representations to run different classifiers to identify if two sensors come from the same shipment or if a sensor profile satisfied a binary quality control criteria for a retailer.

- Clustering: Use different clustering algorithms to cluster the sensors' temperature profiles in the dataset based on similarities in their readings, distribution, hidden/latent factors, distance, and locations.

6.3 Possible Sensor and Data Applications

- Dimensionality reduction: Researchers can employ standard dimensionality reduction algorithms like the principal component analysis (PCA) or even deep learning approaches like the autoencoders or variational autoencoders to learn low-dimensional features in the latent space of the representative temporal data vector to improve subsequent tasks of regression, classification, and clustering.

- Environmental disasters: The location-based prediction can be extended into other applications and fields. For instance, measuring the flood levels in hard to reach places in North Africa and many other places in the world, and predicting their height and direction require installing stream-gauges that are costly in terms of operation and monitoring. Thus, using our approach to predict the hard to reach locations using known accessible locations within the region vicinity can reduce the cost of implementation and save a lot of human lives, animals, and properties.

- Healthcare: Healthcare is another great potential application area where each patient's diagnostics data naturally constitute a multi-variate time-series dataset spanning a wide range of time intervals. In this instance, when one needs to analyze the trends at a certain healthcare facility or geographical area, this data can be processed using our proposed method which can simultaneously improve the visualization while preserving the individual privacy of the patient due to the aggregated cumulative representative form.

6.4 Potential Data Analytics Applications

6.4.1 Time Series Clustering

- The sensors in the dataset can be clustered within the same shipments into different groups based on similar behavior, distribution, hidden/latent factors, distance, and locations.

- Different portions of the time series can also be clustered into different groups to identify the significant events and periods in the cold−chain cycle. For instance, the time series temperature portion of the pre−cooling process across different shipments could be grouped as having a distinctly different temperature profile compared to the rest of the transportation.

6.4.2 Time Series Representation Learning

- The multivariate time series dataset is challenging, nonlinear, and nonstationary with variable lengths. Thus, proper learning representation could help in combining different length sensor profiles into a single representative candidate without the additional padding or clipping mechanisms which can introduce noise.

- Another area of interest is in developing learning representation algorithms to learn complex salient features with deep neural networks for time series sequence modeling to increase prediction accuracy.

6.4.3 Time Series Dimensionality Reduction

- Compare frequency domain−based distance measures with DTW and study the effect of Fast Fourier Transform coefficient on reconstructing the original signal.

Chapter 7: Concluding Remarks

In this dissertation, we provide the reader with a thorough overview of time-series data representation in terms of both its foundations (PART I) and its applications (PART II), with novel contributions to the development of this field from data collection to processing to developing new algorithms for better visualization and classification.

Chapter 1 laid out a thorough history of machine learning, which dates back to the 1950s. At the time, researchers already recognized its importance as a proper way of learning for computers without explicit instructions. However, it was not until the 2006s, with recent advancements in Graphics Processing Units (GPUs) and the proliferation of data that have brought machine learning algorithms like neural networks to the forefront of attention. Since 2010, the field has been completely reshaped with the advancement of deep learning to learn abstract representations and automate the feature-learning process. The consecutive occurrence of both collecting better and more data and building more effective models thanks to increases in computational power have significantly contributed to the field.

In Chapter 2, we introduce our first contribution to science in the form of a novel location-aware time-temperature dataset for perishable transportation across the continental United States with an IoT wireless network implementation. We provided a comprehensive overview of the data and its descriptive statistics with some preliminary time series visualization. Another reason why this data is novel and important is that, unlike many other studies which involved a singular entity, it is the result of a significant collaboration between all the stakeholders in the cold chain from growers to distributors to retailers to academics, which can play an important role for researchers and educators in food engineering, cold-chain, machine learning, and data mining, as well as in other disciplines related to food and transportation.

In Chapter 3, we conducted a comprehensive statistical and temporal analysis to analyze the dynamic factors for temporal heterogeneity, complexity, similarity, and discrepancy on the novel location-aware multivariate time series dataset introduced in the previous chapter. The dataset represents the temperature variations across 6 different shipments with 9 sensors in each shipment monitoring and recording sensory data at 15-minute intervals across multiple days. Through careful hand-analysis and experiments, we concluded that there is still a lot of unknowns when it comes to what happens in the cargo hold throughout the shipment. The dataset is challenging, and it will motivate the food transportation research community to delve into developing more sophisticated regression and classification algorithms for univariate and multivariate time series, better clustering methods, and learning representations with dimensionality reduction. The dataset also provides novel location-aware data with significant predictive potentials to deploy cost-effective intelligent food temperature monitoring systems with high-resolution recording on the minimal number of sensors throughout the entirety of the cold chain.

In Chapter 4, we introduced three algorithms: Correlation Weighted - Moving Average Coefficient (CW-MAC), Dynamic Time Warping Moving Average Coefficient (DTW-MAC), and Unity Weighted Moving Average Time Coefficient (UW-MAC) to generate fixed-length representation from a collection of real-world multivariate time-series datasets with different lengths. CW-MAC, DTW-MAC, and UW-MAC use a layering approach with internal steps called: moving clipping, Pearson's weighting, DTW's weighting, and mathematical concatenations to be computed for each group of multivariate time-series, which is related to the same entity. The proposed algorithmic framework serves as a novel method for better visualization and analytics of multivariate heterogeneous time-series data and helps in reducing the size of the datasets, especially for large scale data. Preliminary results indicate that the proposed algorithms capture the dynamics and statistics of the underlying processes. We also show that CW-MAC, DTW-MAC, and UW-MAC are all robust to noise as the noise power is distributed temporarily across the time-series profiles due to the mechanisms of the proposed algorithm. Furthermore, we implemented different distances and compared their representations and found that the final representations are still very

close in both shape and dynamics. All of the analysis that we have conducted on this novel and temporally rich data introduced in Chapter 2 has highlighted the main challenges of working with temperature profiles along the cold chain in terms of modeling, statistical analysis, and IoT sensor data analytics specifically for sensor analytics and food engineering.

In PART II, we answered the following research questions: What are the main requirements to generate good representations from time-series data? What knowledge is embedded in these representations? If we can generate high-quality representations for better visualization of time-series from numerous sources, can it enable useful follow-up applications like time series classification?

In Chapter 5, we introduced the classification framework which enables the novel CW-MAC and DTW-MAC representations to be used for time series classification and tested it across 70 datasets and 11 different domain applications. We first presented the problem statement of taking the representation from the historical trajectories of the time series observations to solving the classification task as our selected application. The CW-MAC-based classifier uses representations with priors about the series. Then, we aggregate the average accuracies on two different levels: domain/problem types and classes. A major novelty of this work lies in utilizing the representations not only for visualizations, but also for time series classification. A CW-MAC-based classifier is more capable of learning distinct features using statistical aggregation than simple averaging. We constructed comprehensive experiments for verifying the effectiveness and robustness of the proposed representations with DTW as end-to-end classifiers. Interestingly, besides the conventional performance comparisons, we have also found that the proposed representation is able to very effectively reduce the search space by an unprecedented compression of the training data from n examples to c, where c is equal to the number of classes. Finally, our classifier does not introduce any additional computational complexity in the training time, but during testing, the computational efficiency needs to be improved over the use of standard DTW, which is left as an extension of this work. Hence, the next section is dedicated to discussing the possible applications this dissertation can inspire.

In Chapter 6, we discussed future work and open questions in this field. The contributions of this research can be used to advance knowledge in the time-series representation learning domain on several future research trajectories in different directions such as deep learning; cost of sensor implementation; food quality prediction; time-space and time series classification and clustering; time Series representation learning and dimensionality reduction, environmental disasters; and privacy preservation in healthcare applications.

Ultimately, we hope this dissertation will encourage more researchers to work on the applications of time series and apply the proposed representations to new and different domains or tasks. We believe that could lead us towards building better data visualization and analytics techniques specifically in industry implementations on food engineering, cold-chain analytics and machine learning and data mining applications in transportation and healthcare.

References

[1] Alla Abdella and Ismail Uysal. A statistical comparative study on image reconstruction and clustering with novel vae cost function. *IEEE Access*, 8:25626–25637, 2020.

[2] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105:2295–2329, 2017.

[5] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[6] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems 2. chapter Handwritten Digit Recognition with a Back-propagation Network, pages 396–404. Morgan Kaufmann Publishers Inc., 1990.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[8] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010.

[9] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3581–3589, Cambridge, MA, USA, 2014. MIT Press.

[10] R. Wirth and J. Hipp. Crisp-dm: towards a standard process modell for data mining. 2000.

[11] F Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[12] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] B. Karlik and A. Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. 2011.

[15] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE COMMUNICATIONS SURVEYS TUTORIALS*, 20:2595–2621, 2018.

[16] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[18] Dmytro Mishkin and Jiri Matas. All you need is a good init. *CoRR*, abs/1511.06422, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034. IEEE Computer Society, 2015.

[20] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, 2015.

[21] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50. Springer-Verlag, 1998.

[22] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12:145–151, 1999.

[23] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for bench marking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[25] Rui Xu and D. Wunsch, II. Survey of clustering algorithms. *Trans. Neur. Netw.*, 16:645–678, 2005.

[26] DeLiang Wang. Unsupervised learning: Foundations of neural computation. *American Association for Artificial Intelligence*, 2001.

[27] Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl Johann Simon-Gabriel, and Bernhard Schölkopf. From optimal transport to generative modeling: the vegan cookbook. Technical report, 2017.

[28] QIANG LIU (Member IEEE) GEN ZHANG JIANJING CUI Erxue Min, XIFENG GUO and JUN Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE. Translations*, 6:2169–3536, 2018.

[29] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.*, 59(4-5):291–294, 1988.

[30] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[31] Jan G. De Gooijer and Rob J. Hyndman. 25 years of time series forecasting. 2006.

[32] Antonio Rafael Sabino Parmezan, Vinícius M. A. de Souza, and Gustavo E. A. P. A. Batista. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Inf. Sci.*, 484:302–337, 2019.

[33] Everette S. Gardner. Exponential smoothing: The state of the art–part ii. 2006.

[34] John H. Cochrane. Time series for macroeconomics and finance. *Graduate School of Business, University of Chicago*, 1997.

[35] Keith W. Hipel and A. Ian. McLeod. *Time series modelling of water resources and environmental systems*. Elsevier Amsterdam ; New York, 1994.

[36] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., USA, 1990.

[37] James M. Lucas and Michael S. Saccucci. Exponentially weighted moving average control schemes: Properties and enhancements. 1990.

[38] Coundefinedkun Hamzaçebi. Improving artificial neural networks' performance in seasonal time series forecasting. *Inf. Sci.*, 178(23):4550–4559, December 2008.

[39] Guoqiang Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[40] E.J. Hannan and H. Edward James. *Multiple Time Series*. WILEY SERIES in PROBABILITY and STATISTICS: PROBABILITY and STATISTICS SECTION Series. Wiley, 1970.

[41] James Stock and M.W. Watson. Vector autoregressions. *Journal of Economic Perspectives*, 15(4):101 – 116, 2001.

[42] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, May 2009.

[43] P. Cortez. Sensitivity analysis for time lag selection to forecast seasonal time series using neural networks and support vector machines. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[44] Robert R. Andrawis, Amir F. Atiya, and Hisham El-Shishiny. Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting*, 27(3):672–688, July 2011.

[45] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery.

[46] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[47] J. M. Kihoro, Ro Otieno, and Charles Ouma Wafula. Seasonal time series forecasting: A comparative study of arima and ann models. 2006.

[48] John Fulcher. Chapter v application of higher-order neural networks to financial time-series prediction. 2007.

[49] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, 1997.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[51] Anastasia Borovykh, Sander M. Bohte, and Cornelis W. Oosterlee. Conditional time series forecasting with convolutional neural networks. 2017.

[52] I. Koprinska, D. Wu, and Z. Wang. Convolutional neural networks for energy time series forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2018.

[53] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff. Convtimenet: A pre-trained deep convolutional neural network for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[54] C. Liu, W. Hsaio, and Y. Tu. Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(6):4788–4797, June 2019.

[55] F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:67718–67725, 2019.

[56] R. Yang, L. Feng, H. Wang, J. Yao, and S. Luo. Parallel recurrent convolutional neural networks-based music genre classification method for mobile devices. *IEEE Access*, 8:19629–19637, 2020.

[57] J. Elman. Finding structure in time. *Cogn. Sci.*, 14:179–211, 1990.

[58] P. Werbos. Backpropagation through time: What it does and how to do it. 1990.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[60] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555:1–9, 2014.

[61] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. pages 1724–1734, October 2014.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[63] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[65] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[66] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[67] Sepp Hochreiter and Yoshua Bengio. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[68] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *LECTURE NOTES IN COMPUTER SCIENCE*, page 669, 2001.

[69] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2012.

[70] S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[71] A. Essien and C. Giannetti. A deep learning framework for univariate time series prediction using convolutional lstm stacked autoencoders. In *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6, July 2019.

[72] N. Hossain, S. R. Hossain, and F. S. Azad. Univariate time series prediction of reactive power using deep learning techniques. In *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, pages 186–191, Jan 2019.

[73] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David A Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. In *Scientific Reports*, 2016.

[74] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural networks : the official journal of the International Neural Network Society*, 116:237–245, 2018.

[75] King Ma and Henry Leung. A novel lstm approach for asynchronous multivariate time series prediction. pages 1–7, 07 2019.

[76] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. pages 799–804, 01 2005.

[77] Alex Graves. Generating sequences with recurrent neural networks, 2013.

[78] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Laurent Callot, and Tim Januschowski. Neural forecasting: Introduction and literature overview, 2020.

[79] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2017.

[80] Jean-François Toubeau, Jeremie Bottieau, François Vallee, and Zacharie De Greve. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34:1203–1215, 03 2019.

[81] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks, 2017.

[82] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4650–4661. Curran Associates, Inc., 2019.

[83] Yunzhe Tao, Lin Ma, Weizhong Zhang, Jian Liu, Wei Liu, and Qiang Du. Hierarchical attention-based recurrent highway networks for time series prediction. pages 1–10, 2018.

[84] Tryambak Gangopadhyay, Sin Yong Tan, Zhanhong Jiang, Rui Meng, and Soumik Sarkar. Spatiotemporal attention for multivariate time series prediction and interpretation. pages 1–17, 2020.

[85] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 77–87. Curran Associates, Inc., 2017.

[86] Filippo Maria Bianchi, Lorenzo Livi, Karl Øyvind Mikalsen, Michael Kampffmeyer, and Robert Jenssen. Learning representations of multivariate time series with missing data. *Pattern Recognition*, 96:106973, 2019.

[87] D. J. Trosten, A. S. Strauman, M. Kampffmeyer, and R. Jenssen. Recurrent deep divergence-based clustering for simultaneous feature learning and clustering of variable length time series. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3257–3261, May 2019.

[88] Zhiguang Wang, Wei Song, Lu Liu, Fan Zhang, Junxiao Xue, Yangdong Ye, Ming Fan, and Mingliang Xu. Representation learning with deconvolution for multivariate time series classification and visualization. pages 1–8, 2016.

[89] Thach Le Nguyen, Severin Gsponer, Iulia Ilie, and Georgiana Ifrim. Interpretable time series classification using all-subsequence learning and symbolic representations in time and frequency domains. *CoRR*, abs/1808.04022, 2018.

[90] J. Pereira and M. Silveira. Learning representations from healthcare time series data for unsupervised anomaly detection. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–7, 2019.

[91] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, pages 703–716, Cham, 2019. Springer International Publishing.

[92] Dominique T. Shipmon, Jason M. Gurevitch, Paolo M. Piselli, and Stephen T. Edwards. Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. pages 1–9, 2017.

[93] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[94] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 478–487. JMLR.org, 2016.

[95] Michael Kampffmeyer, Sigurd Lse, Filippo M. Bianchi, Lorenzo Livi, Arnt-Bre Salberg, and Robert Jenssen. Deep divergence-based approach to clustering. *Neural Networks*, 113:91 – 101, 2019.

[96] Reiner Jedermann, Luis Ruiz-Garcia, and Walter Lang. Spatial temperature profiling by semi-passive rfid loggers for perishable food transportation. *Computers and Electronics in Agriculture*, 65(2):145 – 154, 2009.

[97] Ricardo Badia-Melis, Ultan Mc Carthy, and Ismail Uysal. Data estimation methods for predicting temperatures of fruit in refrigerated containers. *Biosystems Engineering*, 151:261 – 272, 2016.

[98] Samuel Mercier and Ismail Uysal. Neural network models for predicting perishable food temperatures along the supply chain. *Biosystems Engineering*, 171:91 – 100, 2018.

[99] Samuel Mercier, Jeffrey K. Brecht, and Ismail Uysal. Commercial forced-air precooling of strawberries: A temperature distribution and correlation study. *Journal of Food Engineering*, 242:47 – 54, 2019.

[100] A. Abdella and I. Uysal. Sense2vec: Representation and visualization of multivariate sensory time series data. *IEEE Sensors Journal*, pages 1–1, 2020.

[101] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[102] Maria Cecilia do Nascimento Nunes, Mike Nicometo, Jean Pierre Emond, Ricardo Badía Melis, and Ismail Uysal. Improvement in fresh fruit and vegetable logistics quality: berry logistics field studies. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372, 2014.

[103] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[104] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[105] Joseph Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[106] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[107] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[108] R.N. Bracewell and R. Bracewell. *The Fourier Transform and Its Applications*. Electrical engineering series. McGraw Hill, 2000.

[109] Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *CIKM '00*, 2000.

[110] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.

[111] D. Dickey and W. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74:427–431, 1979.

[112] Jian-Bin Kao and Jehn-Ruey Jiang. Anomaly detection for univariate time series with statistics and deep learning. *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pages 404–407, 2019.

[113] D. M. Diab, B. AsSadhan, H. Binsalleeh, S. Lambotharan, K. G. Kyriakopoulos, and I. Ghafir. Anomaly detection using dynamic time warping. In *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 193–198, 2019.

[114] Chunkai Zhang, Shaocong Li, Hongye Zhang, and Yingyang Chen. Velc: A new variational autoencoder based model for time series anomaly detection, 2020.

[115] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art, 2020.

[116] Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5:597–604, 2006.

[117] Ultan Mc Carthy, Ismail Uysal, Ricardo Badia-Melis, Samuel Mercier, Colm O'Donnell, and Anastasia Ktenioudaki. Global food security – issues, challenges and technological solutions. *Trends in Food Science Technology*, 77:11 – 20, 2018.

[118] E.J. Hannan. *Multiple Time Series*, volume 38 of *Wiley Series in Probability and Statistics*. Wiley, 2009.

[119] J. Fulcher, M. Zhang, and Shuxiang Xu. Application of higher-order neural networks to financial time-series prediction. *Artificial Neural Networks in Finance and Manufacturing*, pages 80–108, 01 2006.

[120] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. 28:1310–1318, 2012.

[121] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181 – 1191, 2020.

[122] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '18, page 95–104, New York, NY, USA, 2018. Association for Computing Machinery.

[123] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[124] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

[125] Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S. Dhillon. Similarity preserving representation learning for time series clustering. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2845–2851. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[126] Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Min. Knowl. Discov.*, 30(2):283–312, March 2016.

[127] Hoang Anh Dau, Nurjahan Begum, and Eamonn Keogh. Semi-supervision dramatically improves time series clustering under dynamic time warping. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 999–1008, New York, NY, USA, 2016. Association for Computing Machinery.

[128] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 262–270, New York, NY, USA, 2012. Association for Computing Machinery.

[129] Rodica Neamtu, Ramoza Ahsan, Elke A. Rundensteiner, Gábor N. Sárközy, Eamonn J. Keogh, Hoang Anh Dau, Cuong Nguyen, and Charles Lovering. Generalized dynamic time warping: Unleashing the warping power hidden in point-wise distances. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 521–532. IEEE Computer Society, 2018.

[130] G. N. Lance and W. T. Williams. Computer Programs for Hierarchical Polythetic Classification ("Similarity Analyses"). *The Computer Journal*, 9(1):60–64, 05 1966.

[131] L. Bruzzone, F. Roli, and S. B. Serpico. An extension of the jeffreys-matusita distance to multiclass cases for feature selection. *IEEE Transactions on Geoscience and Remote Sensing*, 33(6):1318–1321, 1995.

[132] L. Anghinoni, L. Zhao, Q. Zheng, and J. Zhang. Time series trend detection and forecasting using complex network topology analysis. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018.

[133] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1), Jun 2019.

[134] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A review on outlier/anomaly detection in time series data, 2020.

[135] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[136] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[137] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[138] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[139] A. Abdella and I. Uysal. Sense2vec: Representation and visualization of multivariate sensory time series data. *IEEE Sensors Journal*, pages 1–1, 2020.

[140] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.*, 31(3):606–660, May 2017.

[141] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/ eamonn/time$_s eries_d ata/$.

[142] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 1033–1040, New York, NY, USA, 2006. Association for Computing Machinery.

[143] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, August 2008. Copyright: Copyright 2017 Elsevier B.V., All rights reserved.

[144] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Min. Knowl. Discov.*, 29(3):565–592, May 2015.

[145] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, SIGMOD '94, page 419–429, New York, NY, USA, 1994. Association for Computing Machinery.

[146] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. A time series representation model for accurate and fast similarity detection. *Pattern Recognit.*, 42:2998–3014, 2009.

[147] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time, 2019.

[148] Alla Abdella, Jeffrey K. Brecht, and Ismail Uysal. Statistical and temporal analysis of a novel multivariate time series data for food engineering. *Journal of Food Engineering*, page 110477, 2021.

[149] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. The move-split-merge metric for time series. *IEEE Trans. on Knowl. and Data Eng.*, 25(6):1425–1438, June 2013.

[150] Xiaoyue Wang, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26, 12 2010.

[151] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 792–803. VLDB Endowment, 2004.

[152] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664–675, October 1977.

[153] Young-Seon Jeong, Myong K. Jeong, and Olufemi A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recogn.*, 44(9):2231–2240, September 2011.

[154] P.-F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):306–318, Feb 2009.

[155] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, 2013.

[156] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[157] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, Sep 2020.

[158] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, Mar 2019.

[159] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: The collective of transformation-based ensembles. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1548–1549, Los Alamitos, CA, USA, may 2016. IEEE Computer Society.

[160] J. Lines, Sarah Taylor, and A. Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1041–1046, 2016.

[161] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[162] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*. Citeseer, 2004.

Appendix A: Copyright Permissions

The permission below is for the use of our published papers contents at IEEE Access (chapter 1) and IEEE Sensors Journal (chapter 4).

- **Does IEEE require individuals working on a thesis or dissertation to obtain formal permission for reuse?**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

Textual Material

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Full-Text Article

If you are using the entire IEEE copyright owned article, the following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

The permission below is for the use of our papers content at Elsevier Journal of Food Engineering (chapter 2 and 3).

Statistical and temporal analysis of a novel multivariate time series data for food engineering

Author: Alla Abdella,Jeffrey K. Brecht,Ismail Uysal
Publication: Journal of Food Engineering
Publisher: Elsevier
Date: June 2021

About the Author

Received the B.Sc. degree in Electrical and Electronic Engineering from the University of Benghazi, Benghazi, Libya in 2011, and the M.S. degree in Electrical Engineering from Florida Institute of Technology, Melbourne, FL, USA, in 2017. He is currently pursuing a Ph.D. degree in Electrical Engineering at the University of South Florida (USF), FL, USA. Alla works with UST company and AETNA Insurance as a data scientist. He also worked on different projects with the University of Florida, the University of Central Florida, and the COMCAST company. From 2017, he was a Research Assistant with the USF. His research interest includes deep learning neural networks, and applications in unsupervised machine learning, time-series representation and classifications, NLP, and computer vision. Mr. Abdella's awards and honors include valedictorian scholarship for M.S. and Ph.D. degrees.