

KADİR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
PROGRAM OF PhD IN COMPUTER ENGINEERING

**A HYBRID DEEP LEARNING FRAMEWORK FOR
UNSUPERVISED ANOMALY DETECTION IN
MULTIVARIATE SPATIO-TEMPORAL DATA**

YILDIZ KARADAYI

DOCTOR of PHILOSOPHY THESIS

ISTANBUL, AUGUST, 2020

YILDIZ KARADAYI

Ph.D. Thesis

2020



A HYBRID DEEP LEARNING FRAMEWORK FOR UNSUPERVISED ANOMALY DETECTION IN MULTIVARIATE SPATIO-TEMPORAL DATA

YILDIZ KARADAYI

PHD THESIS

Submitted to the School of Graduate Studies of Kadir Has University in partial fulfillment of the requirements for the degree of PhD in the Program of Computer Engineering.

ISTANBUL, AUGUST, 2020

DECLARATION OF RESEARCH ETHICS /
METHODS OF DISSEMINATION

I, YILDIZ KARADAYI, hereby declare that;

- this PhD Thesis is my own original work and that due references have been appropriately provided on all supporting literature and resources;
- this PhD Thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- I have followed “Kadir Has University Academic Ethics Principles” prepared in accordance with the “The Council of Higher Education’s Ethical Conduct Principles”

In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations.

Furthermore, both printed and electronic copies of my work will be kept in Kadir Has Information Center under the following condition as indicated below:

☒ The full content of my thesis/project will not be accessible for 3 years. If no extension is required by the end of this period, the full content of my thesis/project will be automatically accessible from everywhere by all means.

YILDIZ KARADAYI

25/08/2020

KADIR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES

ACCEPTANCE AND APPROVAL

This work entitled **A HYBRID DEEP LEARNING FRAMEWORK FOR UNSUPERVISED ANOMALY DETECTION IN MULTIVARIATE SPATIO-TEMPORAL DATA** prepared by **YILDIZ KARADAYI** has been judged to be successful at the defense exam held on **25 August 2020** and accepted by our jury as **PhD Thesis**.

APPROVED BY:

Assoc. Prof. Dr. Mehmet N. Aydın (Advisor) Kadir Has University _____

Assist. Prof. Dr. A. Selçuk Öğrenci (Co-Advisor) Kadir Has University _____

Assoc. Prof. Dr. Tamer Dağ Kadir Has University _____

Assoc. Prof. Dr. Habib Şenol Kadir Has University _____

Assist. Prof. Dr. N. Ziya Perdahçı Mimar Sinan Fine Arts University _____

Assist. Prof. Dr. Figen Özen Haliç University _____

I certify that the above signatures belong to the faculty members named above.

Prof. Dr. Emine Füsün Alioğlu

Dean of School of Graduate Studies

DATE OF APPROVAL: 25/08/2020

TABLE OF CONTENTS

ABSTRACT	i
ÖZET.....	ii
ACKNOWLEDGEMENTS.....	iii
LIST OF TABLES	v
LIST OF FIGURES	vi
1. INTRODUCTION.....	1
2. RELATED WORK	7
2. BACKGROUND	11
2.1 Convolutional Neural Networks (CNNs)	11
2.2 Long Short-Term Memory (LSTM) Networks	12
2.3 Autoencoders	14
2.3.1 Anomaly Detection with Autoencoders	16
3. PROPOSED METHODOLOGY	18
3.1 Definitions and Problem Formulation	18
3.2 Anomaly Detection Procedure	21
3.3 Proposed Hybrid Autoencoder Framework	23
3.3.1 MC-CNN-Encoder	23
3.3.2 LSTM-Decoder	26
4. EXPERIMENT SET I	29
4.1 Dataset Description	29
4.2 Data Preparation.....	30
4.3 Algorithms Used for Comparison.....	34
4.4 Framework Tuning.....	36
4.5 Results	38
4.5.1 Anomaly detection results.....	38
4.5.2 Quantitative evaluation of models	43
4.5.3 Quantitative evaluation of framework variants.....	44
5. EXTENSION OF THE FRAMEWORK	46
5.1 Spatio-Temporal Encoding	47
5.2 Spatio-Temporal Decoding.....	48

6. EXPERIMENT SET II	50
6.1 Dataset Description	51
6.2 Data Preparation	52
6.3 Framework Tuning	55
6.4 Algorithms Used for Comparison	58
6.4.1 Statistical Models	58
6.4.2 Regression Based Models	59
6.4.3 Distance-based Models	60
6.4.4 Isolation Based Model	60
6.4.5 Deep Learning Models	61
6.5 Anomaly Detection	63
6.5 Results	64
6.5.1 Framework Performance	64
6.5.2 Detection Time Comparison	68
7. CONCLUSIONS	75
REFERENCES	80
CURRICULUM VITAE	89

A HYBRID DEEP LEARNING FRAMEWORK FOR UNSUPERVISED ANOMALY DETECTION IN MULTIVARIATE SPATIO-TEMPORAL DATA

ABSTRACT

Unsupervised anomaly detection for spatio-temporal data has extensive use in a wide variety of applications such as earth science, traffic monitoring, fraud, and disease outbreak detection. Most real-world time series data have a spatial dimension as an additional context which is often expressed in terms of coordinates of the region of interest (such as latitude - longitude information). However, existing techniques are limited to handle spatial and temporal contextual attributes in an integrated and meaningful way considering both spatial and temporal dependency between observations. In this paper, a hybrid deep learning framework is proposed to solve the unsupervised anomaly detection problem in multivariate spatio-temporal data. The proposed framework works with unlabeled data and no prior knowledge about anomalies are assumed. It is more effective in extracting representative features from the multivariate spatio-temporal data compared to the state-of-the-art spatio-temporal anomaly detection approaches. Extensive set of experiments are conducted using two different contextual datasets. The first dataset is the buoy dataset of 2005 from National Data Buoy Center where the Hurricane Katrina is used as ground truth. In this experiment, the proposed hybrid framework achieves more than 10% improvement in accuracy along with increased robustness over the methods used in comparison. The second dataset is the public COVID-19 regional data provided by the website of Italian Department of Civil Protection. In this experiment, results show that proposed framework achieves significant improvement on anomaly detection performance for spatio-temporal data as compared to the state-of-the-art deep learning architecture.

Keywords: Spatio-temporal anomaly detection, unsupervised learning, deep learning, multivariate time series analysis, CNN, LSTM, autoencoders.

UZAMSAL-ZAMANSAL ÇOK BOYUTLU VERİLERDE DENETİMSİZ ANOMALİ TESPİTİ İÇİN HİBRİT DERİN ÖĞRENME ÇERÇEVESİ

ÖZET

Uzamsal-zamansal veriler için denetlenmeyen anormallik tespiti, yer bilimi, trafik izleme, dolandırıcılık ve hastalık salgını tespiti gibi çok çeşitli uygulamalarda yaygın kullanıma sahiptir. Çoğu gerçek dünya zaman serisi verisi, genellikle ilgilenilen bölgenin koordinatları (enlem - boylam bilgisi gibi) cinsinden ifade edilen ek bir bağlam olarak bir uzamsal boyuta sahiptir. Bununla birlikte, mevcut teknikler, gözlemler arasındaki hem uzamsal hem de zamansal bağımlılığı göz önünde bulundurarak, uzaysal ve zamansal bağlamsal nitelikleri entegre ve anlamlı bir şekilde ele almakla sınırlıdır. Bu makalede, çok değişkenli uzay-zamansal verilerdeki denetimsiz anormallik algılama problemini çözmek için hibrit bir derin öğrenme çerçevesi önerilmiştir. Önerilen çerçeve, etiketlenmemiş verilerle çalışır ve anormallikler hakkında önceden bilgi sahibi olunmaz. Çok değişkenli uzamsal-zamansal verilerden temsili özelliklerin çıkarılmasında, son teknoloji uzamsal-zamansal anormallik algılama yaklaşımlarına kıyasla daha etkilidir. Kapsamlı deneyler, iki farklı bağlamsal veri kümesi kullanılarak gerçekleştirilir. İlk veri seti, Katrina Kasırgasının yer gerçeği olarak kullanıldığı Ulusal Veri Şamandırası Merkezi'nden alınan 2005 şamandıra veri setidir. Bu deneyde, önerilen hibrit çerçeve, karşılaştırmada kullanılan yöntemlere göre daha fazla sağlamlığın yanı sıra doğrulukta% 10'dan fazla iyileştirme sağlar. İkinci veri seti, İtalyan Sivil Koruma Dairesi'nin web sitesi tarafından sağlanan halka açık COVID-19 bölgesel verileridir. Bu deneyde sonuçlar, önerilen çerçevenin son teknoloji derin öğrenme mimarisine kıyasla mekansal-zamansal veriler için anormallik algılama performansında önemli iyileşme sağladığını göstermektedir.

Anahtar Sözcükler: Uzay-zamansal anormallik tespiti, denetimsiz öğrenme, çok değişkenli zaman serisi analizi, neural networks, CNN, LSTM, autoencoders.

ACKNOWLEDGEMENTS

I cannot express enough thanks to my advisor Assoc. Prof. Dr. Mehmet N. Aydın and my co-advisor Assist. Prof. Dr. A. Selçuk Öğrenci for their support and encouragement. I offer my sincere appreciation for the learning opportunities and encouragement provided by them.

My completion of this project could not have been accomplished without the support of my husband, Ibrahim Metin Karadayı. Your encouragement is much appreciated, and you deserve my deepest gratitude. To Atalay, my son – thank you for allowing me time away from you to research and write. You deserve much quality time to spend with your mother!



To my family

LIST OF TABLES

Table 4.1	Total number of subsequences.....	33
Table 4.2	Timestamps of detected anomalies.....	39
Table 4.3	Quantitative evaluation of algorithms.....	43
Table 4.4	OAHII1 measures of framework variants.....	44
Table 6.1	Information on Italian regions.....	53
Table 6.2	The framework data structure.....	58
Table 6.3	Outbreak detection date of models.....	69



LIST OF FIGURES

Figure 2.1 Illustration of an Autoencoder.....	15
Figure 3.1 Overall architecture of proposed framework.....	18
Figure 3.2 Spatio-temporal data matrix structure	20
Figure 3.3 The schematic diagram of MC-CNN-Encoder.....	26
Figure 3.4 LSTM-Decoder component.....	28
Figure 4.1 The location of buoys.....	30
Figure 4.2 Sample data from buoys.....	32
Figure 4.3 The architecture of the framework.....	37
Figure 4.4 The path of Hurricane Katrina.....	38
Figure 4.5 Anomaly score comparison.....	42
Figure 4.6 Boxplot of OAHII1 measures.....	45
Figure 6.1 Correlation matrix visualized as heatmap.....	55
Figure 6.2 The architecture of the extended framework.....	57
Figure 6.3 Framework performance for the region Lazio.....	66
Figure 6.4 Framework performance for the region Campania.....	67
Figure 6.5 Framework performance for the region Sicilia.....	68
Figure 6.6 Generated alarms for the region Lazio.....	71
Figure 6.7 Generated alarms for the region Campania	72
Figure 6.8 Generated alarms for the region Sicilia.....	73

1. INTRODUCTION

An anomaly is an observation whose properties are significantly different from the majority of other observations under consideration which are called the normal data. Anomaly detection refers to the problem of finding these observations in data that do not conform to expected or normal behavior. These non-conforming observations are often referred to as anomalies, outliers, abnormalities, exceptions or contaminants in different application domains. Of these, anomalies and outliers are two terms used most in the context of anomaly detection; sometimes interchangeably (Chandola et al. 2009). Outlier analysis is an important research area in data mining and machine learning communities. The recognition of such unusual patterns provides useful insights for many application domains. Some examples are network intrusion detection (Lakhina et al. 2005, Thottan & Ji 2003), fraud detection (Akhilomen 2013, Ghosh and Reilly 1994, Aleskerov et al. 1997, Chan and Stolfo 1998, Derrig 2002), medical image diagnostics (Hazel 2000, Prastawa and Bullitt 2004, Leemput et al. 2001), fault detection and system diagnosis (Fujimaki et al. 2005, Fujimaki 2008, Desforges et al. 1998, Peng and Chu 2004), and earth science (Wu et al. 2010, Zhao et al. 2003, Rogan et al. 2003, Lanorte et al. 2015), where a large amount of spatio-temporal data about weather patterns, climate change and oceans are available.

By the advancement of hardware and sensor technologies for data collection, generation of contextually rich data has become part of many processes. Data from today's applications are temporal (recorded with a time stamp) in nature such as sensor data and financial data. In addition to time context, many temporal data have also spatial context such as geographic location. When the spatial data have temporal component as part of the contextual attributes, it is referred to as spatio-temporal data (Aggarwal 2017). A spatial-temporal outlier (ST-Outlier) is an object whose behavioral (non-spatial and non-temporal) attributes are significantly different from those of the other objects in its spatial and temporal neighborhoods (Gupta et al. 2014). Spatio-temporal data are extremely common in many problem settings where collecting data from various spatial locations and at different times for the nature of the problem are important. In such

settings, identification of ST-Outliers can lead to the discovery of unexpected and interesting knowledge, such as local instability or fraudulent events.

For a problem domain, obtaining the labelled training data for all types of anomalies is often too expensive if not impossible (Hodge and Austin 2004). This highlights the need for unsupervised techniques to find spatio-temporal anomalies. Moreover, spatio-temporal datasets are generally multivariate and have many contextual structures in them (spatial and temporal regularities) which makes them particularly difficult for modelling and well suited for unsupervised learning models. In the unsupervised scenarios, the type of anomalies and the ratio of anomalous events within the given dataset are generally not known. In such scenarios, the normal behavior of the underlying system needs to be modeled in the presence of noise and anomaly which pose extra difficulty.

In the past few years, some density and clustering based unsupervised spatio-temporal anomaly detection algorithms have been developed. The most prominent ones are ST-Outlier detection algorithm (Birant and Kut 2006) and ST-BDBSCAN (Duggimpudi et al. 2017) methods which are basically an extension of DBSCAN (Ester et al. 1996), a density based spatial clustering algorithm. LDBSCAN (Duan et al. 2007), a locality-based outlier detection algorithm, is the merge of both DBSCAN and Local Outlier Factor (LOF) (Breunig et al. 2000). IsolationForest (Liu et al. 2008; Liu et al. 2012) is a powerful approach for anomaly detection in multivariate data without relying on any distance or density measure. In particular, it is an unsupervised, tree-based ensemble method that applies the novel concept of isolation to anomaly detection. It detects anomalies based on a fundamentally different model-based approach: an anomalous point is isolated via recursive partitioning by randomly selecting a feature and then randomly selecting a split value for the selected feature. The output is the anomaly score for each data point. Although it establishes a powerful multivariate non-parametric approach, it works on continuous-valued data only. Despite the inherent unsupervised setting, they may still not detect anomalies effectively due to the following reasons:

- In multivariate time series data, strong temporal dependency exists between time steps. Due to this reason, distance/clustering-based methods, may not perform well

since they cannot capture temporal dependencies properly across different time steps.

- On the other hand, for the case of high-dimensional data, even the data locality may be ill-defined because of data sparsity (Aggarwal 2017).
- Despite the effectiveness of IsolationForest technique, it isolates data points based on one random feature at a time which may not represent spatial and temporal dependencies between data points properly in the spatio-temporal dataset.
- The definition of distance between data points in multivariate spatio-temporal data with mixed attributes is often challenging because of the difficulties in treating the various types of attributes in a homogeneous and consistent way. This difficulty may have an adverse effect on outlier detection performance of distance-based clustering algorithms.

To address these challenges, a hybrid deep autoencoder framework is proposed. It is composed of a multi-channel convolutional neural network (CNN) and Long Short-Term Memory (LSTM) network, named as MC-CNN-LSTM. The architecture of the proposed hybrid framework is based on the autoencoder scheme which has the greater power in representing arbitrary data distributions for anomaly detection than commonly used alternatives such as PCA or matrix factorization (Aggarwal 2017). The multi-channel CNN network is deployed on the encoder side for learning spatial structures of each spatio-temporal multivariate data frame, and LSTM network is deployed on the decoder side for learning temporal patterns from the encoded spatial structures. It is designed to solve the unsupervised anomaly detection problem in non-image multivariate spatio-temporal data.

Major contributions of this thesis can be summarized as follows:

- In aforementioned neighborhood methods, the biggest challenge is to combine the contextual features along the spatial and temporal dimensions in a meaningful way. In the proposed approach, the spatial and temporal contexts are processed by different deep learning components as these contextual variables refer to different types of dependencies.

- All distance/clustering-based algorithms are proximity-based algorithms and dependent on some notion of distance. However, many distance functions used for proximity quantification (such as Euclidean and Manhattan) are not meaningful for all datatypes. For the case of categorical data, it is generally more natural to study similarities rather than distances which are heavily data dependent. Unlike proximity-based methods, the proposed framework can be easily implemented on data with mixed-type attributes and scales well to multivariate datasets with high number of features.
- The proposed hybrid deep learning architecture, named as MC-CNN-LSTM framework, is designed to extract the inter- and intra-correlations within the spatio-temporal neighborhood. The CNN and LSTM layers are combined into one unified framework that is jointly trained. Based upon the spatial features provided by multi-channel CNN encoder, LSTM decoder network extracts better temporal features.
- The framework is designed to be trained in a truly unsupervised fashion without any labels indicating normal or abnormal data. The architecture is robust enough to learn the underlying dynamics even if the training dataset contains noise and anomalies.
- According to the literature on the area, it is the first unsupervised anomaly detection deep learning framework designed specifically for non-image multivariate spatio-temporal data.

The proposed framework works under the following assumptions:

- No prior knowledge about the dataset.
- No prior knowledge about anomalies (such as the distribution of anomalies and types of anomalies).
- Model is trained on unlabeled training dataset which already contains anomalies and noise. It is assumed that the training set contains mostly normal instances and only a few anomalies as, by definition, anomalies are rare events.

To verify the performance of the framework, two sets of experiments are conducted on two different contextual datasets. For the first set of experiments, the year 2005 Gulf of

Mexico (West) buoy dataset from National Data Buoy Center¹ is used. The Hurricane Katrina best track path data (Knabb et al. 2006) is used as ground truth for the experiments. In this set of experiments, the proposed approach is compared with the following state-of-the-art spatial and spatio-temporal anomaly detection algorithms: LOF (Breunig et al. 2000), LDBSCAN (Duan et al. 2007), ST-Outlier detection algorithm (Birant and Kut 2006), ST-BDBSCAN (Duggimpudi et al. 2017), and to a powerful model-based anomaly detection method, IsolationForest (Liu et al. 2008). Results demonstrate that the proposed hybrid deep learning framework achieves superior results by achieving at least 10% improvement in accuracy over the methods used in comparison.

For the second set of experiments, the public COVID-19 regional data provided by the website of Italian Department of Civil Protection is used. The proposed model is tested against the dataset at a resolution of the regional level. One unified model is trained with the data from northern regions and with that model the progression of the COVID-19 epidemic in Italy is tracked and any anomalous trends in central and southern regions are detected. The main assumption here is that the data generated by northern regions' experience going through the epidemic can be used to derive an anomaly detection model.

In this set of experiments, the proposed framework is compared with the following anomaly detection algorithms: statistical models (Shewhart Control Chart, CUSUM, ARIMA), distance based models (LOF and LDBSCAN), IsolationForest, and state-of-the-art deep learning models (CNN Autoencoder, LSTM Autoencoder and a newly proposed spatiotemporal anomaly detection framework based on CNN and ConvLSTM). Results show that proposed framework achieves significant improvement on early outbreak (anomaly) detection performance for spatio-temporal data as compared to the state-of-the-art deep learning architectures.

¹ <https://www.ndbc.noaa.gov/>

This thesis is organized as follows: Related Work section provides an overview of related work in the literature and investigates the different approaches on the anomaly detection problem to address the challenges introduced above. Background section gives necessary theoretical background on deep learning and neural network architectures used in the anomaly detection problems. Proposed Methodology section represents the main contribution of the thesis. First, it gives the formulation of spatio-temporal multivariate anomaly detection problem. Then, it introduces the proposed deep learning framework along with the anomaly detection approach. The section “Experiment Set I” gives information on experimental setup and the preparation of Katrina dataset, then gives results of the experiment. The section “Extension of The Framework” gives details on how the framework is further extended to increase the anomaly detection performance. The section “Experiment Set II” gives information on experimental setup and the preparation of COVID-19 dataset provided by the website of Italian Department of Civil Protection, then gives results of the experiment executed on the extended version of the framework. Finally, Conclusions section provides a summary of the results presented in the thesis and proposes future research directions.

2. RELATED WORK

Outlier or anomaly detection has been studied extensively in the context of time series and spatial data separately. Time-series outlier detection studies find outliers considering only temporal context (Yaminshi and Takeuchi 2002; Cheng et al. 2009; Gupta et al. 2014). For data with spatial context, a number of context-based anomaly detection techniques have been proposed (Shekhar et al. 2003; Lu et al. 2003; Shekhar et al. 2001; Kou et al. 2006; Sun and Chawla 2004). For spatiotemporal outlier detection, both spatial and temporal continuity are used for modeling. Hence, spatio-temporal outlier detection methods are significantly more challenging because of the additional difficulty of modeling the temporal and spatial components jointly (Aggarwal 2017; Cheng and Li 2004; Cheng and Li 2006).

Birant and Kut (2006) propose a neighborhood-based ST-Outlier detection algorithm. They propose a three-step approach to identify the spatio-temporal outliers. First, they use a modified version of DBSCAN algorithm to identify the spatial neighborhoods within the dataset. They define spatial outliers based on these neighborhoods. Then, they check the temporal context of spatial outlier objects by comparing them to temporal neighbor objects.

Cheng and Li (2006) propose a four-step approach to identify spatio-temporal outliers: classification (clustering), aggregation, comparison and verification. Their aim is to address the semantic and dynamic properties of geographic phenomena for ST-outlier detection. At the clustering step, the prior knowledge of the data is used to form some regions that have significant semantic meanings. The aggregation is also called filtering since the outliers (noises) will be filtered by changing the spatial scale. The main idea here is that if there are spatial outliers, they usually disappear if the scale of processing is reduced, clustering results will be different with different scales. With a decrease in scale, the difference between the objects and their spatial neighbors will decrease and the small regions that contain outliers will be removed. At comparison step, the results obtained at

two spatial scales are compared in order to detect the potential spatial outliers. At the verification step, the outliers detected in the previous step can be suspected as ST-outliers. Therefore, the verification step checks the temporal neighbors of the suspected ST-outliers detected in the previous step. If the attribute value of such a ST-outlier is not significantly different from its temporal neighbors, this is not a ST-outlier. Otherwise, it is confirmed as a ST-outlier.

Adam et al. (2004) propose a spatio-temporal outlier detection approach methodology based on Voronoi diagrams. Their methodology is based on building micro and macro neighborhoods using the spatial and semantic relationships among the objects. They first build Voronoi diagrams using spatial properties of each object to find micro neighborhood. By using spatial and semantic relationships between objects, they find the macro neighborhood, which is an extended neighborhood, of each object. Using these neighborhoods, they detect outliers based on distance (Euclidean distance) values among various points. A data object is said to be a spatio-temporal outlier if it differs sufficiently from other points in the macro neighborhood. The macro neighborhood consists of all the micro neighborhoods merged into it under the spatial and the semantic relationship restrictions.

Gupta et al. (2013) introduce the notion of context-aware anomaly detection in distributed systems by integrating the information from system logs and time series measurement data. In addition to temporal context, they use system specific performance metrics such as number of tasks running, memory usage, and CPU usage to create additional contextual data. They propose a two-stage clustering methodology using a principle component analysis (PCA) (Jolliffe 2002) based method and K-Means (Aggarwal and Reddy 2013) algorithm to extract context and metric patterns. They instantiate their framework for Apache Hadoop platform. They extract additional context variables using the job history logs. They first cluster instances based on non-temporal context variables to extract context patterns, then apply time context using time series metrics variables to detect outliers.

The aforementioned methods have something in common: They do not use the spatial and temporal components jointly to detect anomalies. They first apply spatial (or non-temporal) context to find spatial outliers using a distance-based technique. Then, spatial outliers are compared with other spatial objects using temporal neighborhoods to detect if they are temporal outliers too. They all use either spatial clustering-based outlier detection algorithms such as DBSCAN (Ester et al. 1996), or locality-based outlier detection algorithms such as LOF (Breunig et al. 2000) to find neighborhoods. They can only detect point anomalies like extreme cases, but they cannot detect collective anomalies in the context of spatio-temporal data. The main challenge with distance-based methods is that they are computationally too expensive for large multivariate datasets.

Besides traditional distance and density-based methods, deep learning-based anomaly detection approaches have recently gained a lot of attention. Long Short-Term Memory (LSTM) networks have been used for anomaly detection on time series data as it achieves better generalization capability than traditional methods. An LSTM network-based Encoder-Decoder scheme for anomaly detection was recently proposed (Malhotra et al. 2016) where the model learns to reconstruct ‘normal’ time series data and uses reconstruction error to detect anomalies. A recent deep learning model combining a convolutional neural network (CNN), LSTM, and deep neural network (DNN) were designed for detection of traffic anomalies in web servers (Kim and Cho 2018). They used a fully labeled dataset for supervised learning and solved a univariate time series classification problem. Another deep learning framework combining a convolutional encoder, attention based convolutional Long-Short Term Memory (ConvLSTM) network, and a convolutional decoder for detecting spatio-temporal anomalies for multivariate time series data was proposed by Zhang et al. (2019). In addition to network complexity, their approach requires the calculation of multiscale system signature and residual signature matrices to perform anomaly detection thereby increasing the overall complexity in calculations.

Christiansen et al. (2016) propose a CNN-based anomaly detection system to detect obstacles in an agriculture field. Their anomaly detection framework uses features extracted by a CNN detector. Oh and Yun (2018) propose an autoencoder based approach

to detect abnormal operations of a complex machine using operational sound. They adopt a general convolutional autoencoder architecture and use residual errors to detect anomalies. Munir et al. (2019) propose a novel approach for anomaly detection in streaming sensors data which takes advantage of both statistical and deep learning-based techniques. They use statistical ARIMA model and a convolutional neural network (CNN) model to forecast the next time-stamp in a given time-series. The forecasted value is further passed to an anomaly detector module that marks a data point as normal or anomalous based on the distance between the predicted value and the actual value.

Despite their effectiveness of those mentioned deep learning approaches, they are either supervised or semi-supervised models which need labelled normal or abnormal data for training. The proposed hybrid deep learning framework is designed to be trained in a truly unsupervised fashion without any labels indicating normal or abnormal data. The architecture is robust enough to learn the underlying dynamics even if the training dataset contains noise and anomalies. The main distinction between other deep learning based methods and the proposed approach is that they perform on either univariate or multivariate time series data, and none of them is actually designed for spatio-temporal multivariate datasets with both spatial and temporal contextual attributes.

2. BACKGROUND

In this section, the key components of the proposed hybrid deep learning framework are introduced. First, some background information on CNNs and LSTM networks are given. Then, autoencoders, which are a special type of neural network structures, and anomaly detection approach using autoencoders are introduced.

2.1 Convolutional Neural Networks (CNNs)

Fully convolutional network (FCN) has shown compelling quality and efficiency for image classification and segmentation. It has been showed that a fully convolutional network, which is trained end-to-end, pixels-to-pixels, given the category-wise semantic segmentation annotation, exceeds the state-of-the-art without further creating hand-crafted features (Long et al. 2015, Krizhevsky et al. 2012).

The layers of a convolutional neural network are arranged to represent spatial information within data in a grid structure. These spatial relationships are inherited from one layer to the next creating representative features based on a small local spatial region in the previous layer. This spatial transformation is achieved by three types of layers which are commonly present in a convolutional neural network: convolution, pooling, and activation function layers. Each layer in the convolutional network represents a three-dimensional grid structure of size $h \times w \times d$, where height (h), width (w) and depth (d). h and w are spatial dimensions, and d refers to the number of channels or the number of feature maps in the hidden layers. In addition to hidden layers, a final set of fully connected layers is often used in an application specific way to generate the desired output (Aggarwal 2018).

Let $x^l = [x_{ijk}^l]$ be a 3-dimensional tensor representing feature maps in the l th layer. The indices i, j, k indicate the positions along the height, width, and depth of the filter maps. The weights of the p th feature map in the l th layer are represented by the 3-dimensional tensor as $W^{(p,l)} = [w_{ijk}^{(p,l)}]$. The transformation from the l th to the $(l+1)$ st convolutional layer after activation function applied can be written as follows:

$$y_{ijp}^{(l+1)} = f\left(b_p^l + \sum_{r=1}^M \sum_{s=1}^M \sum_{k=1}^{d_l} w_{rsk}^{(p,l)} x_{i+r-1,j+s-1,k}^l\right) \quad (2.1)$$

where f is the activation function, such as $\tanh$ or ReLU, b_p^l is the bias for the p th feature map, $w_{rsk}^{(p,l)}$ represents the parameters of the p th filter, and M is the size of the filter. The convolutional operation is a simple dot product over the entire volume of the filter, which is repeated over spatial dimensions (i, j) and filters (indexed by p). d_{l+1} is the number of feature maps, or number of channels, in the $(l + 1)$ st layer, where $p \in \{1..d_{l+1}\}$.

The pooling operation works on small spatial regions and produces another layer with the same depth. The pooling layers decrease the spatial dimensions of each activation map and effectively reduce the number of parameters and computational complexity of the network. Max pooling and average pooling are the mostly used pooling techniques in CNNs. Max pooling operation with pooling size of $R \times R$ returns the maximum of the values in the square region of pool size in each layer. Stride value defines how much the pooling window will move in each spatial dimension, which is another way that convolution can reduce the spatial footprint of the feature maps. Previous studies (Scherer et al. 2010) show that max pooling is superior to average pooling for image classification and it also leads to faster convergence rate and improves generalization performance by preventing overfitting. Hence, the max pooling strategy is applied in convolutional neural network component of the proposed framework.

2.2 Long Short-Term Memory (LSTM) Networks

LSTM network is a type of recurrent neural network (RNN) with controllable memory units that enable the network to learn when to forget previous hidden states and when to update hidden states with new information (Hochreiter and Schmidhuber, 1997). They have been proved to be very effective in many long-range sequential modeling applications such as machine translation, speech and action recognition (Sutskever et al. 2014, Graves and Jaitly 2014, Srivastava et al. 2015, Wu et al. 2015). Generally, LSTM recursively maps the input data at the current time step to a sequence of hidden states, and thus the learning process of LSTM should be in a sequential manner.

LSTM networks address the vanishing gradient problem commonly found in ordinary RNNs by incorporating the concept of gating into their state dynamics. The inputs provided to the LSTM memory unit are fed through gates which are the input, output, and forget gates. At each time step, an LSTM unit updates a hidden state vector h_t and a cell state vector c_t responsible for controlling state updates and outputs using current input, previous cell state and hidden state values. Specifically, given a sequential data of T timestamps $(x_1, x_2, \dots, x_T)$, an LSTM memory unit maps the inputs to an output sequence $(y_1, y_2, \dots, y_T)$ by computing the gate functions recursively for each step from $t = 1$ to $t = T$ with the following equations:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (2.2)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2.3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (2.4)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (2.5)$$

$$h_t = o_t \circ \tanh(c_t) \quad (2.6)$$

where x_t and h_t are the input and hidden state vectors, the notations i_t, f_t, o_t, c_t are the activation vectors of the input gate, forget gate, output gate, and memory cell respectively. W_{ab} is the weight matrix between a and b , and b_a is the bias term of a , σ is the sigmoid function. $\circ$ denotes the element wise product. The memory cell unit c_t is the sum of two terms: the previous memory cell unit modulated by forget gate and a function of the current input and previous hidden state. Input and forget gates can be thought of as knobs that control the learning process: the LSTM learns to selectively forget its previous memory through f_t and considers its current input and previous hidden state through i_t . The output gate o_t learns how much of the current memory cell to transfer to the hidden state. This gated structure seems to enable the LSTM to learn complex and long-term temporal dynamics for contextual data.

2.3 Autoencoders

Autoencoders are commonly used for dimensionality reduction of multidimensional data as a powerful non-linear alternative to PCA or matrix factorization (Hinton and Salakhutdinov 2006; Bengio 2009; Hecht-Nielsen 1995). If linear activation function is used, the autoencoder becomes virtually identical to a simple linear regression model or PCA/matrix factorization model. When a nonlinear activation function is used, such as rectified linear unit (ReLU) or a sigmoid function, the autoencoder goes beyond the PCA, capturing multi-modal aspects of the input distribution (Aggarwal 2017; Japkowicz et al. 2000). It is shown that, carefully designed autoencoders with tuned hyperparameters outperform PCA or K-Means methods in dimension reduction and characterizing data distribution (Vincent et al. 2010; Rifai et al. 2011). They are also more efficient in detecting subtle anomalies than linear PCAs and in computation cost than kernel PCAs (Hinton and Salakhutdinov 2006; Sakurada and Yairi 2014).

A traditional autoencoder is a feed-forward multi-layer neural network which is trained to copy its input into the output. An autoencoder is trained to encode the input x into some representation z so that the input can be reconstructed from that representation. Hence the target output of the autoencoder is the autoencoder input itself. In order to prevent identity mapping, deep autoencoders are built with low dimensional hidden layers by creating non-linear representation of input data (Hinton and Salakhutdinov 2006). An autoencoder is trained by unsupervised learning to learn how to build its original input with a minimum reconstruction error. Figure 2.1 depicts a typical autoencoder network structure with one hidden layer. They are composed of two parts: an encoder and a decoder. They basically learn a non-linear mapping from the input to itself through encoding and decoding steps. An autoencoder takes an input vector $x \in R^d$, and first maps it to a latent representation $z \in R^{d'}$ through a deterministic mapping:

$$z = f_{\theta}(x) = Wx + b \quad (2.7)$$

where the function f_{θ} represents encoding steps and parameterized by $\theta = \{W, b\}$. W is a $d' \times d$ weight matrix and b is a bias vector. The lower dimensional latent representation of the input is then mapped back to a reconstructed vector $x' \in R^d$ in the input space:

$$x' = g_{\theta'}(z) = W'z + b' \quad (2.8)$$

where the function $g_{\theta'}$ represents decoding steps and parameterized by $\theta' = \{W', b'\}$. The autoencoders training procedure consists of finding a set of parameters $\{W, b, W', b'\}$ that make the reconstructed vector x' as close as possible to the original input x . The parameters of the autoencoder are optimized by minimizing a loss function that measures the quality of the reconstructions. The loss function of an autoencoder is sum-of-squared differences between the input and the output. Therefore, an autoencoder tries to minimize the following loss function during the training:

$$L(x, x') = \sum_{x \in \varphi} \sum_{i=1}^d ||x_i - x'_i||^2 \quad (2.9)$$

where φ is the training dataset.

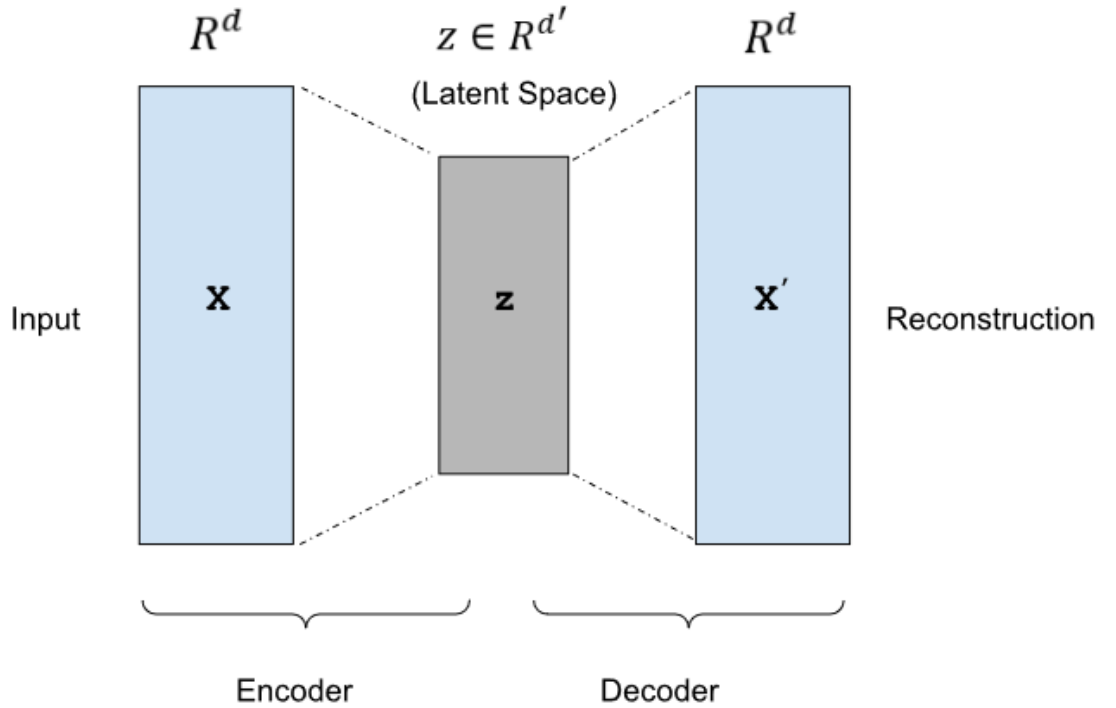


Figure 2.1 Illustration of an Autoencoder.

2.3.1 Anomaly Detection with Autoencoders

The main idea behind autoencoder based anomaly detection is to measure how much the reconstructed data deviates from the original data. An autoencoder has an unsupervised learning objective whose primary task is to copy the input to the output (Goodfellow et al. 2016). Therefore, an autoencoder is trained to reconstruct data by minimizing this objective function, or loss function. For anomaly detection, reconstruction error is used as the anomaly score. Data points which generate high reconstruction errors can be categorized as anomalous data points based on a threshold value. When autoencoders are used for anomaly detection, they are trained using semi-supervised learning. In this scenario, only normal data instances are used in training process. The training dataset should be cleaned from anomalous data points and outliers as much as possible for a successful model generation. Therefore, for the anomaly detection model to be successful in semi-supervised learning approach, it is critical to design a pre-processing phase that can filter out or correct such noise from the data when possible. This can be achieved using a variety of unsupervised domain-specific methods, such as statistical and distance-based methods. For example, in the context of time series data, autoregressive models, for spatial data, distance and density-based models can be used for noise removal. Techniques such as Principle Component Analysis (PCA) and Singular Value Decomposition (SVD) methods can also be used in order to improve the representation quality of data and remove discordant data points (Gupta et al. 2014; Aggarwal 2017). After the training process, the autoencoder will generally reconstruct normal data with very small reconstruction error. As the autoencoder has not encountered with the abnormal data during the training, it will fail to reconstruct them and generate high reconstruction errors which can be used as anomaly score. Below, Algorithm 1 shows the anomaly detection procedure of autoencoders using reconstruction errors.

There are some practical issues in using autoencoders with contaminated training data (dataset with normal and anomalous data points). Since anomalies are treated as normal data points during the training phase, there will be inevitably more errors in the model compared to training with only normal data points. If we try to overcome these errors by tuning the network with more layers and neurons, we may face the problem of overfitting which is the significant problem in the case of deep neural networks. A sufficiently

complex deep autoencoder may even learn how to represent each anomaly with sufficient training by generating low reconstruction errors which would be a problem in anomaly detection (Aggarwal 2018).

Algorithm 1 Autoencoder-based Anomaly Detection

INPUT: Normal dataset X for training,
 anomalous dataset $y^{(i)}$ for testing, $i = 1, \dots, N$
 threshold α (optional)

OUTPUT: Data points in the test set with calculated
 reconstruction errors: $\|y - \hat{y}\|$
 labels (optional)

$\theta', \theta \leftarrow$ train the autoencoder using dataset X

for $i=1$ **to** N **do**

$\hat{y}^{(i)} = g_{\theta'}(f_{\theta}(y^{(i)}))$

reconstruction_error(i) = $\|y^{(i)} - \hat{y}\|$

if threshold α is given **and**
 reconstruction_error(i) > α **then**

label = $y^{(i)}$ is an anomaly

else

label = $y^{(i)}$ is not an anomaly

end if

end for

3. PROPOSED METHODOLOGY

In this section, the proposed hybrid deep learning framework for unsupervised anomaly detection problem, which is shown in Figure 3.1, is explained in detail. First, problem definition and data structure of multivariate spatio-temporal data are given. Then, anomaly detection problem is formulated on multivariate spatio-temporal data. Finally, each component of proposed framework is explained.

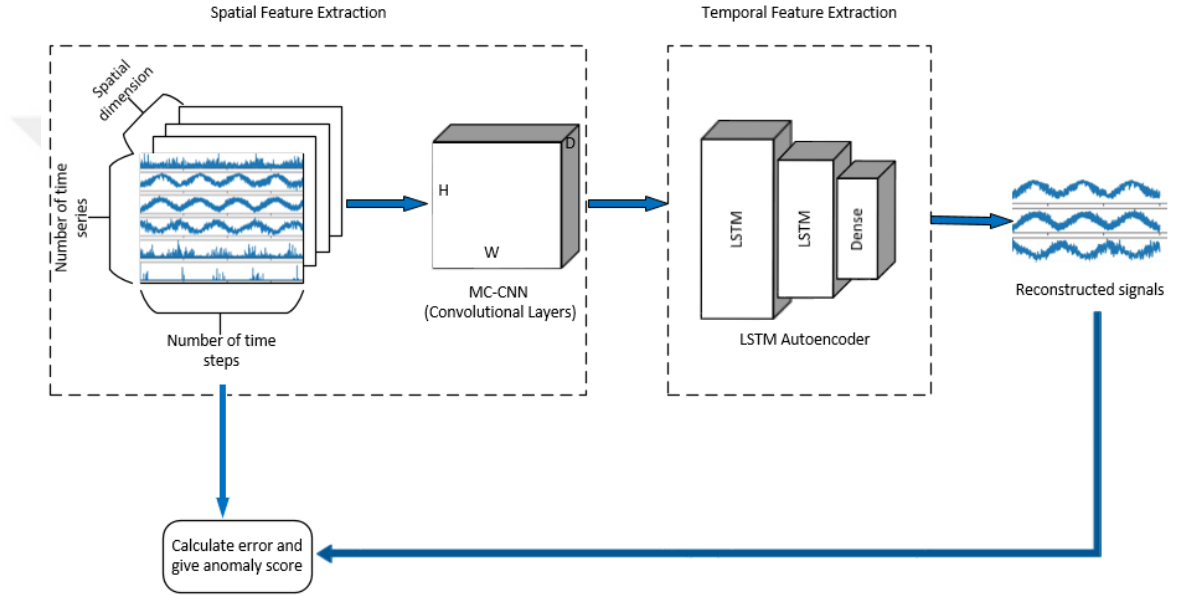


Figure 3.1 Overall architecture of proposed framework.

3.1 Definitions and Problem Formulation

A univariate time series is a sequence of real valued data points with timestamps. A multivariate time series is a set of univariate time series with the same timestamps. In this study, we focus on multivariate time series that are measured at successive points in time, spaced at uniform time intervals.

Let $X = \{x^{(n)}\}_{n=1}^N$ denote a multivariate time series dataset composed of N data points. Let each data point $x^{(n)}$ has T time steps, and each observation at time step t , is a d dimensional vector. The dataset X has dimensions of (d, T) , where $x^{(n)} \in \mathbb{R}^{d \times T}$. Each data point $x^{(n)}$ is a two-dimensional data matrix and can be represented as:

$$\mathbf{x}^{(n)} = \begin{pmatrix} x_{11}^n & \cdots & x_{1T}^n \\ \vdots & \ddots & \vdots \\ x_{d1}^n & \cdots & x_{dT}^n \end{pmatrix} \quad (3.1)$$

The superscript n represents the ordered number of each data point within the dataset X . $\mathbf{x}^{(n)}$ is a multivariate time series data point with a contextual time attribute. Each $\mathbf{x}^{(n)}$ in the dataset X is ordered based on the timestamp. As the number n increases, the time context changes, and time dimension, or timestamps, moves ahead.

In a spatio-temporal dataset, each multivariate data point $\mathbf{x}^{(n)}$ comes from a different spatial location, or region, which has different spatial attributes (such as latitude and longitude). We denote the multivariate spatio-temporal dataset as $D_{ST} = \{(X^{(i)}, S^{(i)})\}_{i=1}^m$ which contains multivariate time series data points from m different spatial regions. Each spatial region S_i , where $S_i \in S$, has a set of multivariate time series data represented by the dataset $X^{(i)}$. $N^{(i)}$ represents the number of data points (or observations) in each spatial region S_i . In other words, it is the size of $X^{(i)}$, which may be different for each region in real-world scenarios.

X_{ST} , which is the multivariate spatio-temporal data matrix, can be represented as a 3-dimensional tensor as shown in Figure 3.2. It is built using multivariate time series data from m different spatial regions or S_i s, where $i = 1 \dots m$. The sliding window technique which is used to build the 3-dimensional data matrix X_{ST} , is given in Algorithm 2. It is composed of m multivariate time series data points from m different spatial regions and representing observations from the same time window with the same timestamps. T , which is called the “input window-size”, represents the number of timestamps in the multivariate data point, and d represents the number of univariate time series. m represents the number of nearest spatial neighborhood to include in the anomaly detection process. The best m can be found empirically for each problem domain. The m number of nearest neighboring regions are selected from S different regions based on the pairwise spatial distance between regions.

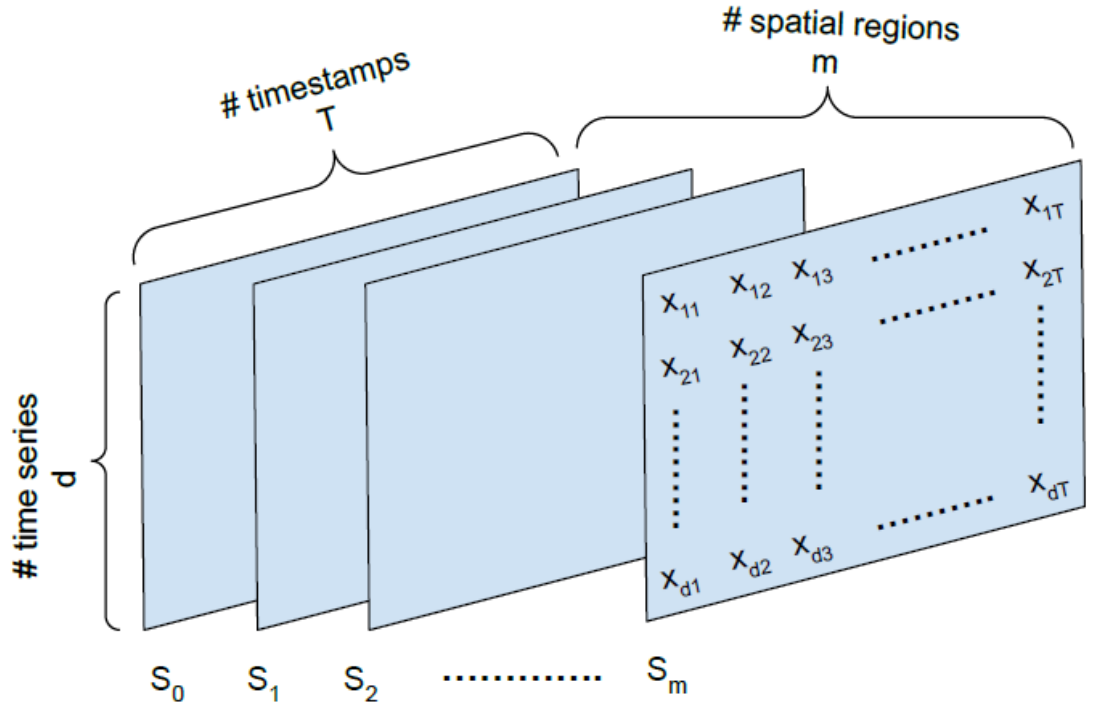


Figure 3.2 Spatio-temporal data matrix structure for anomaly detection procedure.

The total number of multivariate subsequences from each spatial location can be quite different. In a real-world application, the number of missing observations from different sensors in each time frame can be different due to sensor failure or data loss. As a result, the number of subsequences in each dataset $X^{(i)}$ is not the same, which poses a great difficulty to train a robust deep learning framework to extract representative features from the dataset.

Algorithm 2 Sliding window algorithm used in subsequence generation

```
# dataset: region dataset
# T : Number of timesteps, window size (T)
# region_list: region id list
# s : sliding step size
# depth: the number of spatial neighbors
# distance_matrix: distance matrix of regions
# output : multivariate spatio-temporal tensor data

process_dataset (dataset,
                 T,
                 region_list,
                 s,
                 depth,
                 distance_matrix)

1:  output = list()
2:  for each region in region_list:
    # get region data from dataset
3:  data ← dataset[region]
4:  start_indx = 0
5:  end_indx = start_indx + T
    # step through the region data
6:  while (end_indx < length(data)):
    # get start and end timestamps of data slice of
    size T
7:  start_date = getStartTimestamp (data, start_indx)
8:  end_date = getEndTimestamp(data, end_indx)
    # get subsequence from region data:
    # seqs : 3D spatio-temporal data
9:  seqs[n, 0] = data[start_indx..end_indx]
    # get subsequences from depth-1 nearest
    neighbours:
10:  seqs[n, 1:depth] = getDataFromNeighbours(dataset,
T, start_date_time, end_date_time, region_code, depth-1)
    output.append(seqs)
10.  start_indx = start_indx + s
11.  end_indx = start_indx + T
12.  n = n + 1 # end-while
13.  return output
```

3.2 Anomaly Detection Procedure

The spatio-temporal anomaly detection problem is formulated as detecting anomalous multivariate observations in dataset D_{ST} which differentiate significantly from their spatial and temporal neighbors. Given the spatio-temporal 3-dimensional data matrix X_{ST} , the goal is to reconstruct the multivariate subsequences from the spatial region S_i , which is the region of interest. Anomalous data points have large reconstruction errors because they do not conform to the subspace patterns in the data. Therefore, for the proposed autoencoder

based anomaly detection framework, the aggregate error of reconstruction over all d dimensions can be used as the anomaly score. All $x_i^{(n)}$ multivariate data points, or subsequences, with high reconstruction errors from the spatial region S_i are considered to be anomalies.

The anomaly detection process is performed using the reconstruction error as anomaly score. Let $x = \{x^{(1)}, x^{(2)}, \dots, x^{(T)}\}$ be a multivariate time series data point from the multivariate time series dataset X , where $x \in \mathbb{R}^{d \times T}$, T is the length of the time series window, and d is the number of features (univariate time series). Each point $x^{(i)}$ is a d -dimensional vector of readings for d variables at time instance t_i . The root mean squared error (RMSE) is used to calculate the reconstruction error for a given time step t_i as given by the following equation:

$$e(i) = \sqrt{\frac{1}{d} \sum_d (x_i - \hat{x}_i)^2} \quad (3.2)$$

where x_i is the real value and $\hat{x}_i$ is the reconstructed value at time step t_i . The reconstruction error for each time step in the output data is calculated. The result is the aggregated error of output data which has a window size of T . These aggregated errors of data points can be used as anomaly score and high scoring data points are regarded as spatio-temporal anomalies. The proposed hybrid deep learning framework uses unlabeled spatio-temporal data points for training. Training data might be contaminated by anomalous data which is the case in most of the real word scenarios. The multi-channel CNN architecture helps to overcome the problem of overfitting. It processes data which comes from m spatial neighborhood simultaneously and extracts representative features of it. This approach helps to build a model that represents the great majority of data points and not the anomalous points which are rare. Anomalous observations can be thought of as noise and the framework is designed to be robust enough not to learn representation of those points. The framework is trained to reconstruct data by minimizing a loss function (mean squared error) that measures the quality of the reconstructions. The main hypothesis is that after training, the model is able to reconstruct the input data that conform to expected or normal behavior within the dataset, while it fails to reconstruct anomalous data, since it has not learned any features to represent them.

3.3 Proposed Hybrid Autoencoder Framework

The proposed autoencoder framework consists of two major components: a multi-channel convolutional neural network-based encoder for spatial feature extraction (MC-CNN-Encoder) and a Long Short-Term Memory based decoder for temporal feature extraction (LSTM-Decoder). The framework is connected in a linear structure and trained jointly to minimize the reconstruction error. The CNN layers are an important part of the framework and work as encoder to extract spatial features. The multi-channel structure helps to capture the spatial dependencies in neighboring spatial regions. The reconstruction error of a data point is used for anomaly score. Our model design is inspired by the network combining CNN and LSTM into one unified architecture for text prediction problem (Sainath et al. 2015), which shows that LSTM performance can be improved by providing better features. The overall architecture of the framework is illustrated in Figure 3.1.

The performance of the designed framework depends on various components and training parameters. The number of layers and units in each layer, the number of kernels used in each convolutional layer, size of each kernels, striding strategy, dropout ratio and type and size of pooling operators are all important parameters in the anomaly detection process as they can affect the learning performance and speed. They play an important role in extracting more representative features of the training data. To determine the optimal architecture and the best training parameters for the overall network, including the optimum parameters used to preprocess the raw data such as window size and step size, the characteristics of the input data has to be analyzed carefully. In this thesis, the optimum parameters of the framework for each set of experiment are selected based on the best empirical performance of grid search on training dataset.

3.3.1 MC-CNN-Encoder

For the spatial encoder component, an effective multi-channel convolutional encoder is proposed. Each channel of the encoder takes multivariate time series data from different spatial neighbors as input. Using multichannel approach, spatio-temporal features from the spatial neighborhood of each data point can be extracted. As in image data,

multivariate spatio-temporal data are processed by 2D kernels and 2D pooling operations. In contrast to other deep learning models with CNNs which were applied on time series with 1D kernels (Zheng et al. 2014, Zheng et al. 2016, Wang et al. 2016), 2D kernels are applied onto spatial dimensions to extract spatial features.

The way how traditional CNN network works is modified by applying it to multivariate spatio-temporal data in the following way: A long multivariate time series data from each spatial location is preprocessed using parameters of window size (T), sliding step size (s_{step}), and depth of the input (d). d represents the number of channels of the first convolutional layer and used to select nearest spatial neighbors as explained in Algorithm 2 in section 4.2. The number of nearest spatial neighbors to use in anomaly detection process is controlled by the depth of the input data. The optimum number of depth can be selected based on the nature of dataset and cardinality of spatial context. The resulting pre-processed data point, which is a multivariate subsequence, is a 3D tensor with third dimension representing the channels, and each channel is fed with the nearest neighbor's multivariate data as first channel being assigned to current data point. Spatial neighborhood of each data point is calculated based on spatial context attribute (such as latitude and longitude) using a distance measure (such as Euclidian).

MC-CNN-Encoder component is composed of two convolutional blocks, and each block is made up of four cascading layers: 2D convolution kernel layer, followed by an activation layer, followed by 2D max-pooling layer and a channel-wise batch normalization layer. In contrast to image data, the input of the proposed encoder is multivariate time series data instead of pixels from RGB images. The input shape of the first convolution block is a three dimensional tensor (d, T, s) where d is the number of behavioral features (univariate time series), T is the window size or number of time steps processed at once, and s is the number of channels, which represents the size of spatial neighborhood. For each convolutional block, 2D convolutional layer captures how spatial features at each lower level change over time, 2D max-pooling enables efficient computation of long-ranging temporal patterns, and batch normalization improves robustness towards various environmental conditions. Specifically, input data are shaped

as a tensor $X^{t,0} \in \mathbb{R}^{d \times T \times s}$ and then fed into the first convolutional layer. The output of the first filter (convolution) layer is given by:

$$y = W^1 * X^{t,0} + b^1 \quad (3.3)$$

where $*$ denotes the 2D convolution operation, $W^1 \in \mathbb{R}^{k_1 \times k_1}$ denotes filter of size $k_1 \times k_1$, $b^1 \in \mathbb{R}^{s_1}$ is the bias term, and $y \in \mathbb{R}^{d \times T \times s_1}$ denotes the output of the convolution operation after activation function was applied. s_1 is the number of feature maps (or depth) and it is controlled by number of filters in the convolutional layer. The first convolutional layer has 16 filters of size (2×2) with zero-padding and no striding (or with strides (1×1)). After applying rectified linear unit (ReLU) (Nair and Hinton 2010) activation function, a channel-wise batch normalization is applied before max-pooling step. Batch normalization (BN) has been proved effective in recent CNN applications and it is applied to speed up the convergence of overall framework and help improve generalization (Ioffe and Szegedy 2015). Dimensionality reduction is achieved by max pooling step with pool size (2×2) . The final output of the first convolutional block is given by:

$$s = \text{ReLU}(y) \quad (3.4)$$

$$h = \text{MaxPooling}(s) \quad (3.5)$$

$$X^{t,1} = \text{BN}(h) \quad (3.6)$$

where $X^{t,1} \in \mathbb{R}^{d_1 \times T_1 \times s_1}$ denotes the output of the first convolutional block, MaxPool is the max pooling operation with pool size (2×2) and strides (2×2) to cut temporal dimensions by half at each max pooling layer.

The second convolution block has convolutional layer with 32 filters of size (3×3) without striding and with zero padding. It is made up of same activation, max pooling and batch normalization layers as in the first block. The output of the second convolution block can be written as a tensor $X^{t,2} \in \mathbb{R}^{d_2 \times T_2 \times s_2}$, where d_2 and T_2 denotes the downsampled temporal dimensions, s_2 denotes the number of channels (spatial dimensions) set by number of filters in the second convolutional layer.

The MC-CNN-Encoder takes 3-dimensional multivariate spatio-temporal input data and generates a higher order representation of it. The dimension of the output of the MC-CNN-Encoder is $(timestamps \times features \times feature\ maps)$. Thus, we need to add a layer to reshape the output before passing to the LSTM layer. The output of the MC-CNN-Encoder is flattened over spatial dimensions into a feature vector and sent to the LSTM Decoder. As we go deeper in the network, the width and height dimensions (the temporal and feature dimensions representing multivariate time series) tend to shrink as a result of max pooling operation, while the number of feature maps (spatial dimension) for each Conv2D layer controlled by number of filters increases. This architecture helps to extract complex spatio-temporal dependencies for each data point in the dataset. To be understood easily, the detailed architecture of MC-CNN-Encoder component is illustrated in Figure 3.3.

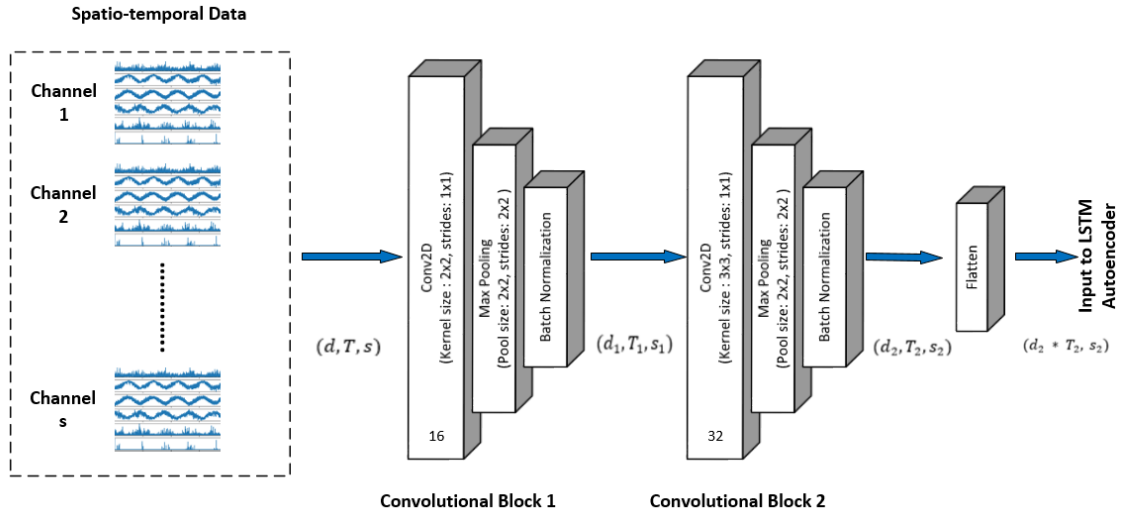


Figure 3.3 The schematic diagram of multi-channel convolutional neural network encoder (MC-CNN-Encoder).

3.3.2 LSTM-Decoder

The spatial feature maps generated by the MC-CNN-Encoder are temporally dependent on previous time steps. A LSTM-Decoder is designed to represent those temporal dependencies. It decodes the MC-CNN-Encoder output and reconstructs the sequence data using hidden states (feature maps). It is composed of an LSTM block and a fully

connected neural network (FCNN) layer. The LSTM block comprises of two LSTM layers followed by a dropout layer. The output of the LSTM block is fed to the FCNN layer where the output is mapped into the feature set we want to reconstruct. The-LSTM Decoder component is illustrated in Figure 3.4.

During the process of anomaly detection, the MC-CNN-Encoder learns a 3-dimensional vector representation of the multivariate spatio-temporal input data. Then, the LSTM-Decoder uses the reshaped representation to extract temporal features and reconstruct the input data. The LSTM-block is trained to learn higher level temporal features of the input data, whereas the FCNN layer is trained to reconstruct the target univariate time series. In the LSTM block, two hidden layers are used with 64 and 16 hidden units each and followed by a dropout layer to prevent overfitting.

For the final dense (FCNN) layer, the number of units is set to d' and it equals to the number of univariate time series (features) that we want to reconstruct. The number of hidden units in the dense layer can be adjusted according to the problem context at hand. For an anomaly detection problem, we are only interested with the reconstruction of a subset of original spatio-temporal multivariate dataset and not the fully reconstructed version of it. The overall framework is trained to produce the target multivariate time series $X = \{x_1, \dots, x_{T'}\}$ of length T' which is the size of the reconstruction window. The length of T' can be equal to or smaller than the input window size T and should be tuned for each problem. Each sequence $x_i \in \mathbb{R}^{d'}$ is an d' -dimensional vector where $d' \leq d$.

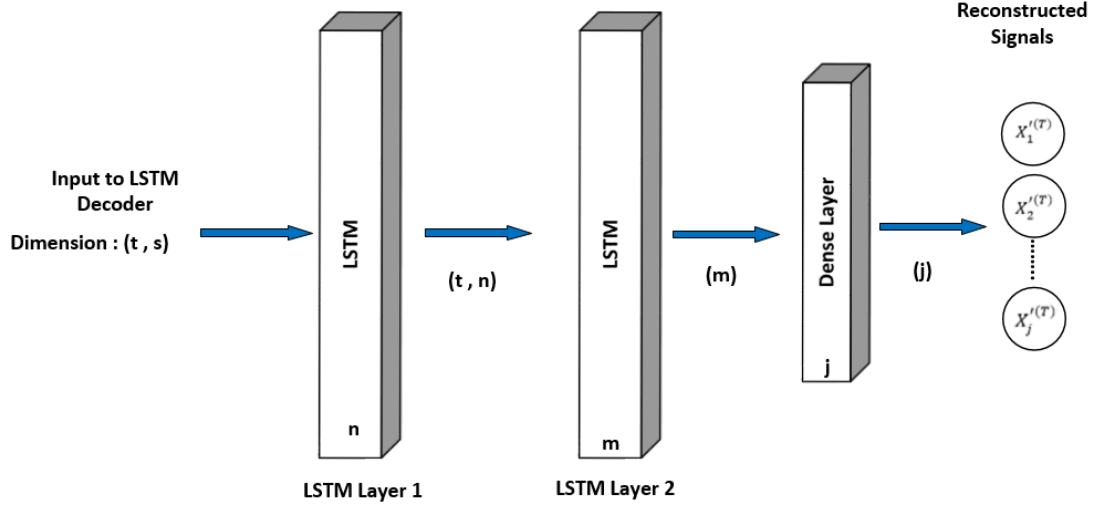


Figure 3.4 LSTM-Decoder component: Each layer represents inference steps to reconstruct original signals from spatio-temporal latent feature space.

Given a multivariate input signal $x \in \mathbb{R}^{d \times T}$ from the spatio-temporal multivariate dataset ST , the input to the LSTM-Decoder is $z \in \mathbb{R}^{d_3 \times s_2}$, where $d_3 = d_2 \times T_2$ and s_2 is the number of the filters (feature maps) in the final convolution layer. The output of the final LSTM layer is the hidden state h_T which is the lower dimensional vector representation of the original input signal. Final FCNN layer uses this hidden state vector as input to reconstruct the target time series. The final output of the framework is the reconstruction of the target multivariate subsequences, which can be written as follows:

$$\hat{x} = f(w * h_T + b) \quad (3.7)$$

where w is a weight matrix, b is a bias vector, and f is a non-linear activation function. The dimension of h_T is equal to the number of neurons in the last LSTM layer. The model is trained to minimize the following mean squared error (MSE) loss function:

$$MSE = -\frac{1}{N} \sum_i^N (x - \hat{x})^2 \quad (3.8)$$

where N is the total number of data points in the batch run, $\hat{x}$ is the predicted time series for the target time period, and x is the target time series. By increasing the size of the reconstruction window, we can detect long ranging and collective anomalies.

4. EXPERIMENT SET I

To verify and evaluate the proposed hybrid deep learning framework for spatio-temporal anomaly detection, a set of experiments are conducted on buoy dataset. The year 2005 Gulf of Mexico (West) buoy dataset, which is the year of Hurricane Katrina, is used for these experiments. A buoy is a floating object in water and used to mark the location and to record oceanographic data. Our main motivations in selecting this dataset and doing this case study can be given as:

- Because of the difficulty of finding labelled spatio-temporal multivariate real-world dataset for anomaly detection, we decided to use the dataset full of with well-known natural phenomena (anomalies) such as hurricanes.
- Associating the spatio-temporal outliers detected between August 25th and August 30th with the path of Hurricane Katrina helps us to compare our approach with other algorithms.
- To get a better understanding about the performance of algorithms, we used OAHII (Outliers Association with Hurricane Intensity Index) measures developed by Duggimpudi et al. (2017) using the same buoy dataset. Using this measure, we are able to quantitatively compare the anomaly scores generated by algorithms.

4.1 Dataset Description

The buoy dataset has many meteorological measurements such as wind direction, wind speed, gust speed, wave height, air temperature, water temperature, tide etc. However, buoy measurements from the year 2005 have many missing values or missing features. We have selected buoys which have sufficient historical data from the 2005 hurricane season for comparison purposes. For each buoy data reading, there are two contextual attributes: a time stamp (temporal context) attribute and latitude-longitude (spatial context) attribute which is static for each buoy. Beside these contextual attributes there are six behavioral attributes which are mostly available for each buoy used for this experiment: wind direction, wind speed, gust speed, barometric pressure, air temperature, and water temperature. After having eliminated buoys with many missing features and/or values and the ones that were damaged during the hurricane season we have selected 21 buoys and their data were included in our experiment. Those buoys are: 41001, 42001,

data) on maximum number of consecutive missing values that could be filled. Even after interpolation, there are still huge number of missing attributes for some of buoys as can be seen in Figure 4.2.

The wind direction attribute is defined as “the direction the wind is coming from in degrees clockwise from true N” by National Data Buoy Center and it takes values between 0 and 360. In this study, we have divided the full 360 degrees value into 8 equally sized slices of 45 degrees and label encoded each slice. Then, wind speed attribute of observations from buoys were one-hot encoded using these labels based on which slice direction its value falls in. Meanwhile, we have applied the min-max scaling technique on 5 attributes, which are water temperature (WTMP), air temperature (ATMP), wind speed (WSPD), gust speed (GST) and barometric pressure (BAR), to map all values to the range [0,1].

We consider all data readings starting from January 1st, 2005 00:00 till August 31st, 2005 23:59; separating data from January till the end of June as training set, data from July as validation set and data from August as test set. We need all selected attribute values available for all data points to train models. We have discarded any observations with missing attributes, resulting a total of 93982 readings from 21 different buoys. In final dataset, the total number of data points are different for each buoy as the number of missing attributes differs significantly. In the final dataset, the number of data readings from each buoy is shown in Table 4.1, in column “# Observations”.

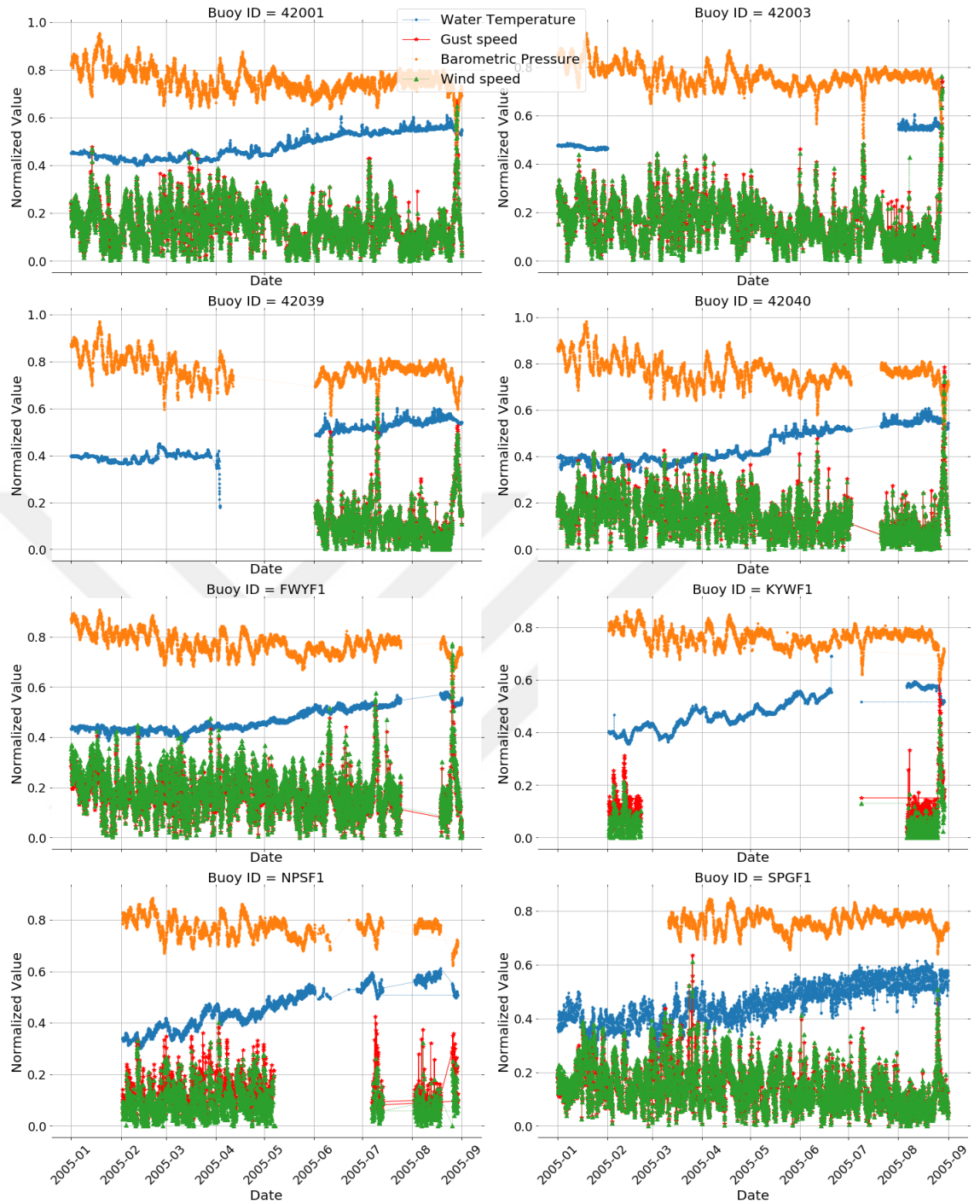


Figure 4.2 Sample data from buoys 42001, 42003, 42039, 42040, FWYF1, KYWF1, NPSF1, and SPGF1 after interpolation. Attributes shown: Water temperature (WTMP), Gust speed (GST), Barometric Pressure (BAR), and Wind speed (WSPD).

Training is executed using overlapping sequences obtained with a sliding window algorithm with a window size T and step size of s . The sliding window technique given in Algorithm 2 is updated to mitigate the missing attribute problem. The updated version of the sliding window algorithm which is applied to the pre-processed buoy dataset to extract subsequences is given in Algorithm 3. The length of each subsequence is equal to the window size. The performance evaluation of the framework for different window sizes is given in Section 4.5. Using sliding window technique, for a long sequence with length L , the number of extracted subsequences can be given as $\left\lfloor \frac{L-T+1}{s} \right\rfloor$. This gives the maximum number of subsequences we can possibly extract. As shown in Figure 4.2, each buoy has different number of missing data which results in data sequences with different length. Because of the high number of missing readings, we have tolerated missing data points on some time stamps by keeping the time difference limit between the end points of a subsequence at $2 \times T$. After having applied Algorithm 2 with window size set to 12 and step size to 1, we have generated training, validation and test sets which have different number of subsequences for each buoy as shown in Table 4.1.

Table 4.1 Total number of subsequences belonging to each buoy created for training, validation and test steps.

Buoy ID	# Observations	# Training subsequences	# Validation subsequences	# Testing subsequences
42001	5822	4325	731	730
42002	5807	4295	732	725
42003	1421	755	0	642
42036	5452	4304	372	729
42038	5811	4316	732	715
42039	2178	682	730	723
42040	5348	4297	274	723
BURL1	5760	4327	732	665
BYGL1	3953	2195	697	574
CDRF1	5625	4105	732	730
FWYF1	5231	4325	565	305
GDIL1	5782	4321	732	693
KYWF1	880	399	0	444
LONF1	5825	4326	732	731
MLRF1	5827	4328	732	731
NPSF1	2638	2044	100	397
SANF1	5818	4323	731	728
SGOF1	5573	4268	539	730
SMKF1	3622	2124	732	730
SPGF1	4172	2674	732	730
VCAF1	1437	108	706	573

Algorithm 3 Sliding window algorithm used in subsequence generation

```
# p: current spatial context (eg. sensor)
# L: Full time series of current spatial context (multivariate)
# T: window size
# s: sliding step size
# d: number of channels (depth of 3d spatio-temporal data matrix)
# seqs: 3D tensor data for the current spatio-temporal data point
1: function seqs = ExtractSubsequences(p, L, T, s, d)
2:   start_indx = 0
3:   end_indx = start_indx + T
4:   n = 0   # n is the number of subsequences
5:   neighbour_list = getNearestNeighbours(p, d-1)
6:   while (end_indx < length(L)):
7:     # get timestamp of data at start_indx and end_indx
8:     start_date_time = getTimestamp(L, start_indx)
9:     end_date_time = getTimestamp(L, end_indx)
10:    # check if there is a big time gap between start and end indexes
11:    If (end_date_time - start_date_time) < (2*T):
12:      # get the subsequence from the long time series data
13:      seqs[n, 0] = L[start_indx..end_indx]
14:      # get data from (d-1) nearest neighbours and attach to seqs
15:      seqs[n, 1:d] = GetDataFromNeighbours(T,
16:                                           start_date_time,
17:                                           end_date_time,
18:                                           neighbour_list)
19:      start_indx = start_indx + s
20:      end_indx = start_indx + T
21:      n = n + 1
22:   return seqs
```

4.3 Algorithms Used for Comparison

To validate the effectiveness of the proposed approach, spatio-temporal outliers detected by the proposed framework and by the following state-of-the art anomaly detection techniques are examined: ST-Outlier detection algorithm, ST-BDBSCAN, LOF, LDBSCAN, and Isolation Forest. ST-Outlier detection and LDBSCAN algorithms are implemented in Python 3.6.8 programming language. For LOF³ and IsolationForest⁴ methods, implementations available in the scikit-learn⁵ are used, which is a free machine

³ <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html>

⁴ <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html>

⁵ <https://scikit-learn.org/stable/>

learning library for Python programming language. The Euclidean distance for all proximity-based clustering algorithms since it has generated better results compared to other distance metrics. As ST-Outlier detection algorithm does not return a score, it is not included in the quantitative evaluation tests.

Model parameters were selected through randomized search over a predefined list for each parameter. For LDBSCAN algorithm, following values were assigned to the parameters since they gave the best result, as also stated in Duggimpudi et al. (2017): $LOFUB = 4$, $MinPts_{LOF} = 180$, $MinPts_{LDBSCAN} = 225$, and $pct = 0.27$. For LOF algorithm, following values were used: 50 for number of neighbors to use for k-nearest neighbor calculations, and 0.1 for contamination ratio. For ST-Outlier detection algorithm, following values were assigned to the parameters: $Eps1 = 100$ (distance parameter for spatial attribute), $Eps2 = 0.2$ (distance parameter for non-spatial attribute), $MinPts = 5$, which is the minimum number of points within $Eps1$ and $Eps2$ distance of a point. For IsolationForest, we have used 100 base estimators (number of trees), and contamination was set to 0.1.

Besides the above baseline methods, the depth of the components of the framework for the sample dataset were also evaluated: Number of layers in MC-CNN-Encoder component and number of layers in LSTM-decoder component. The quantitative evaluation of variants of the framework is given in Section 4.5.3.

In order to compare the proposed approach quantitatively to models which return an outlieriness score, three different *OAHII* (Outliers Association with Hurricane Intensity Index) measures developed by Duggimpudi et al. (2017) were used. *OAHII* measures show how much the generated scores for each observation are correlated with the impact of the ground truth spatio-temporal anomaly which is the Hurricane Katrina for the test data. While *OAHII1* measures the Pearson correlation between the outlieriness score and actual impact of the hurricane on each data point and impact, *OAHII2* measures the recall power of models using the impact vector as weight factor in weighted correlation, a derivative of Pearson correlation, and *OAHII3* measures the precision power of models using the scores vector as weight vector in weighted correlation.

4.4 Framework Tuning

The proposed framework is implemented using the Keras⁶ deep learning library using Python, running on top of the Tensorflow⁷ backend. All hidden layers are equipped with the Rectified Linear Unit (ReLU) non-linearity which allows the networks converge faster (Simonyan and Zisserman 2014). For the final fully connected neural network layer, sigmoid non-linear activation function is used. Optimization was performed by ADAM optimizer algorithm (Kingma and Ba 2014). The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Layer weights were initialized with the Glorot uniform initializer (Glorot and Bengio 2010). The framework is trained in mini-batches of size 32 for 20 epochs, with learning rate=0.001, beta_1=0.9, beta_2=0.999, and epsilon=1e-07. The final framework has 35,092 parameters to optimize. The optimum parameters for training epochs, batch size, learning rate and activation function were selected through grid search. The detailed architecture of the framework is illustrated in Figure 4.3.

Although window size is an important hyper parameter for time series analysis, the proposed framework is not very sensitive to it as it is explained in the next section. The performance of the proposed framework is evaluated using window sizes 12, 6, and 3. The anomaly detection results given in Section 4.5.1 were generated by the framework with window size set to 12 ($T=12$) and sliding step size set to 1.

⁶ <https://keras.io/>

⁷ <https://www.tensorflow.org/>

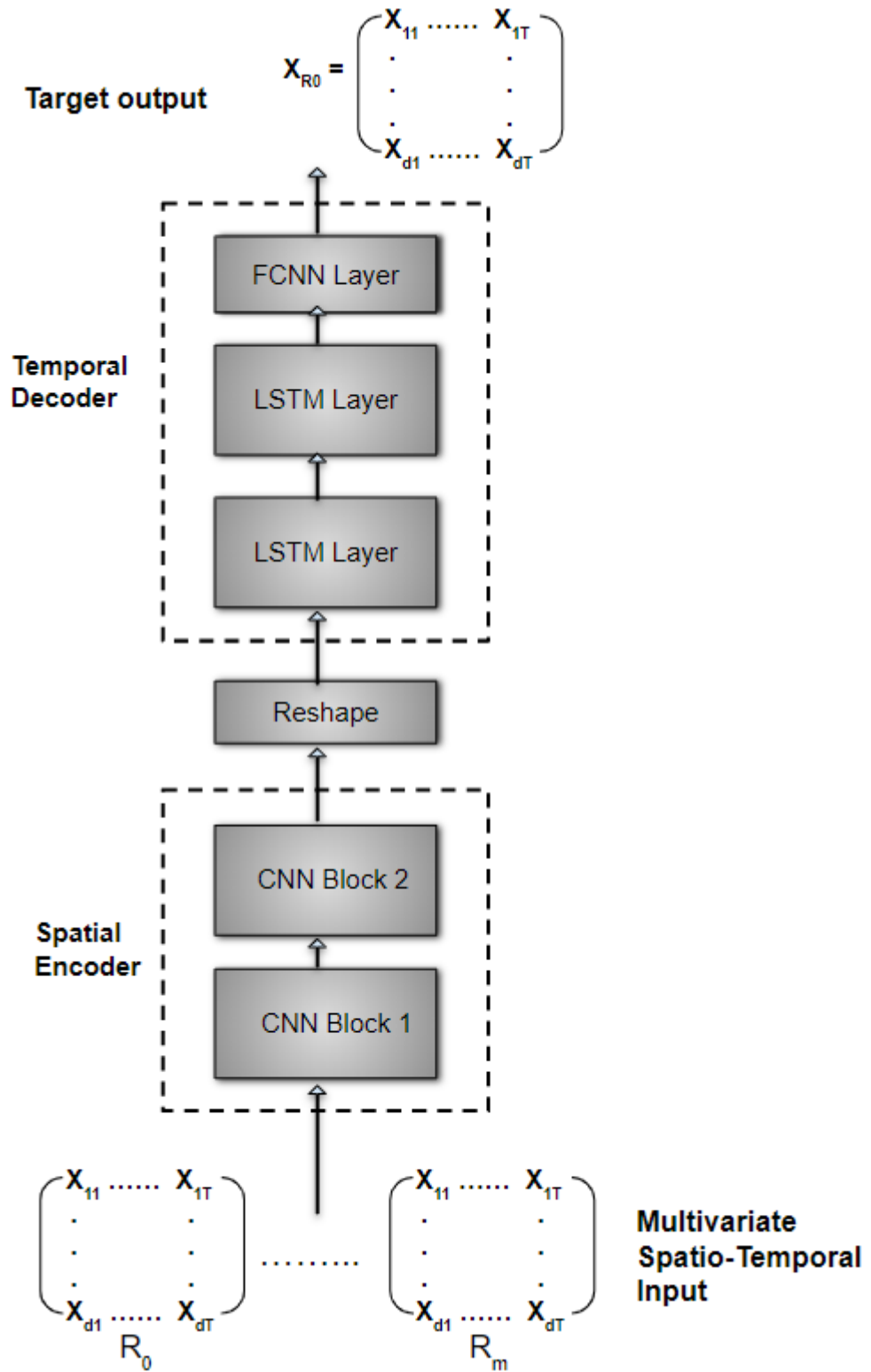


Figure 4.3 The architecture of the hybrid deep autoencoder framework.

4.5 Results

4.5.1 Anomaly detection results

In Figure 4.4, the location of buoys and path of Hurricane Katrina along with the time stamps are shown. The track of Katrina is plotted according to “best track” positions and intensities measured in every six hours, which starts on 00:00 UTC 25 August and ends on 18:00 UTC 29 August (Knabb et al. 2006). The cyclone became Katrina, the 11th tropical storm of the 2005 Atlantic hurricane season, at 12:00 UTC 24 August when it was over the central Bahamas. Katrina is estimated to have reached hurricane status near 21:00 UTC 25 August, less than two hours before its center made landfall on the southeastern coast of Florida. As it entered the Gulf of Mexico on 26 August, Katrina embarked upon two periods of rapid intensification between 26 and 28 August. Katrina strengthened from a low-end Category 3 hurricane to a Category 5 in less than 12 hours reaching the peak by 12:00 UTC 28 August. The large wind field on 28 August, with hurricane-force winds extended out to about 90 nautical miles from the center, made Katrina extremely intense and exceptionally large (Knabb et al. 2006).

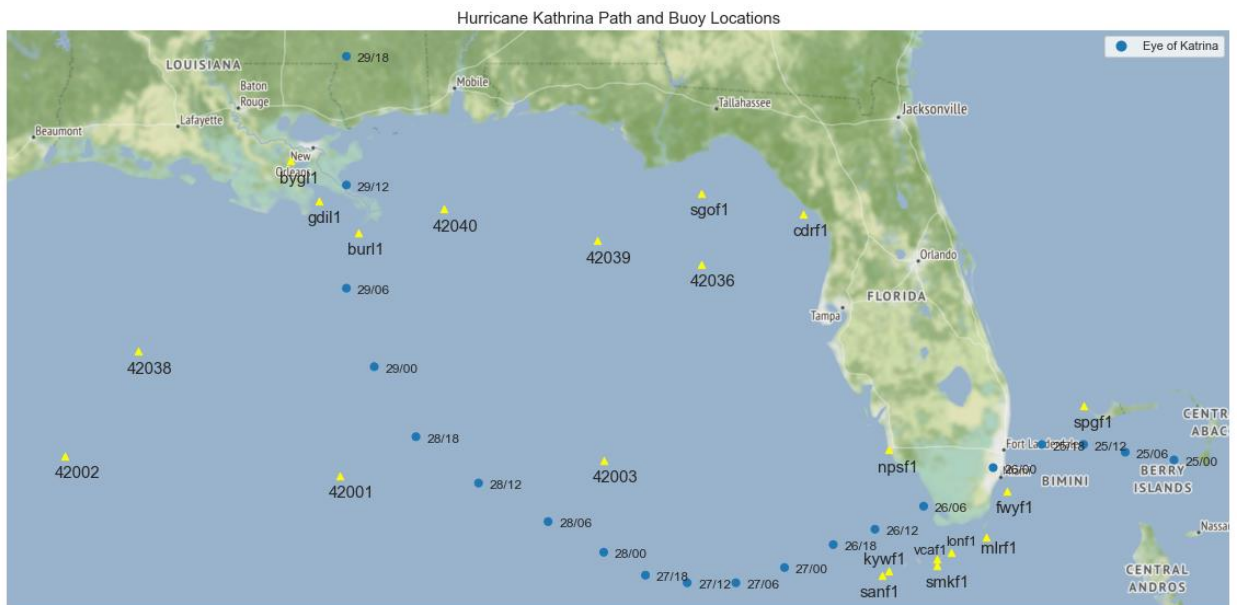


Figure 4.4 Buoy locations and path of Hurricane Katrina.

The center of Katrina was very close when it passed by the buoy 43003 on 06:00 UTC 28 August, the buoy 42001 on 18:00 UTC 28 August, and the buoy 42040 on 09:00 UTC 29 August. The timestamps of top 5 scoring spatio-temporal anomalous data points for buoys 42001, 42003, and 42040 detected by algorithms are given in Table 4.2. The anomalies which are in space-time vicinity of the Hurricane Katrina are given in blue for performance comparison. The proposed framework (MC-CNN-LSTM) with window size 6 detected all spatio-temporal anomalies caused by Hurricane Katrina as it passed by buoys. When we examine the results, LDBSCAN seems to give the highest score to data points which are the most related with Hurricane Katrina path after our model, followed by ST-Outlier detection algorithm, LOF and IsolationForest.

Table 4.2. Timestamps of detected anomalies. Anomalous data points which are in the spatio-temporal neighborhood of the Hurricane Katrina are given in blue.

Buoy ID	IsolationForest	LOF	LDBSCAN	ST-Outlier Detection	MC-CNN-LSTM (T=6)
42001	2005-08-29 09:00	2005-08-28 20:00	2005-08-28 20:00	2005-08-27 08:00	2005-08-28 20:00
	2005-08-29 10:00	2005-08-28 21:00	2005-08-28 21:00	2005-08-28 19:00	2005-08-28 21:00
	2005-08-29 11:00	2005-08-28 19:00	2005-08-28 22:00	2005-08-28 20:00	2005-08-29 00:00
	2005-08-29 12:00	2005-08-29 09:00	2005-08-29 00:00	2005-08-28 21:00	2005-08-28 22:00
	2005-08-29 19:00	2005-08-29 11:00	2005-08-28 23:00	2005-08-31 23:00	2005-08-28 19:00
42003	2005-08-27 05:00	2005-08-07 20:00	2005-08-27 17:00	2005-08-24 20:00	2005-08-28 01:00
	2005-08-27 03:00	2005-08-27 00:00	2005-08-28 01:00	2005-08-24 21:00	2005-08-28 04:00
	2005-08-27 04:00	2005-08-27 06:00	2005-08-27 18:00	2005-08-24 22:00	2005-08-28 02:00
	2005-08-28 05:00	2005-08-27 08:00	2005-08-27 16:00	2005-08-28 04:00	2005-08-28 03:00
	2005-08-28 04:00	2005-08-27 09:00	2005-08-27 20:00	2005-08-28 05:00	2005-08-28 00:00
42040	2005-08-29 20:00	2005-08-29 16:00	2005-08-29 13:00	2005-08-29 12:00	2005-08-29 13:00
	2005-08-29 16:00	2005-08-29 10:00	2005-08-29 12:00	2005-08-29 13:00	2005-08-29 12:00
	2005-08-29 17:00	2005-08-29 09:00	2005-08-29 11:00	2005-08-29 14:00	2005-08-29 10:00
	2005-08-29 18:00	2005-08-29 08:00	2005-08-29 14:00	2005-08-29 15:00	2005-08-29 11:00
	2005-08-29 19:00	2005-08-29 23:00	2005-08-29 10:00	2005-08-31 19:00	2005-08-29 09:00

Although this heuristic analysis gives a high-level understanding about performance of algorithms, it is better to take all data points and their scores into account for the duration of Hurricane Katrina. Actual Katrina path reported by Knabb et al. (2006) gives latitude and longitude information of the center reported for every six hours. We have applied linear interpolation to transform it to an hourly basis data to match buoys' hourly readings data. For each buoy whose location is highly relevant to Katrina path, we have plotted the anomaly score of each data point against the distance to the center of the Katrina for the time frame starting on 00:00 UTC 25 August and ending on 23:59 UTC 29 August. In order to show comparison in detail, anomaly scores of proposed framework (MC-CNN-LSTM), LDBSCAN and IsolationForest algorithms are plotted against the distance between buoys and the center of the Katrina in Figure 4.5. Anomaly score values and distances are min-max normalized in the range of [0, 1] before plotting.

In Figure 4.5, we can observe that the anomaly scores of LDBSCAN and IsolationForest are not stable and results contain many false positives and false negatives. For the LDBSCAN algorithm, false alarms based on generated anomaly scores are quite visible for buoys 42038, LONF1, and SANF1. For buoy 42038, LDBSCAN generates three highest scores for data points when Katrina center is not the closest. For buoys LONF1 and SANF1, LDBSCAN exhibits high anomaly score values followed by sharp drops which followed by high score values again for data points when Katrina center is very close. On the other hand, IsolationForest, which is the next best performing algorithm according to the results of quantitative evaluation given in section 4.5.2, shows weaker performance than the proposed framework in capturing the anomalous trend in the test data. Its performance degrades significantly for buoys FWYF1, SANF1, and SMKF1. Although these buoys are on the direct path of the Katrina, anomaly scores generated by IsolationForest are not stable and exhibits high values for data points when Katrina center is not very close. Meanwhile, the anomaly scores generated by MC-CNN-LSTM is smoother than LDBSCAN and IsolationForest. In other words, the anomaly scores generated by MC-CNN-LSTM do not exhibit sharp drops and high peaks in consecutive data points. The proposed framework outperforms LDBSCAN and IsolationForest in capturing the overall anomalous trend in the test data. In summary, MC-CNN-LSTM generates less false alarms and does not miss real alarms, as generated anomaly score

values show higher correlation with the distance between buoys and the center of the Katrina.



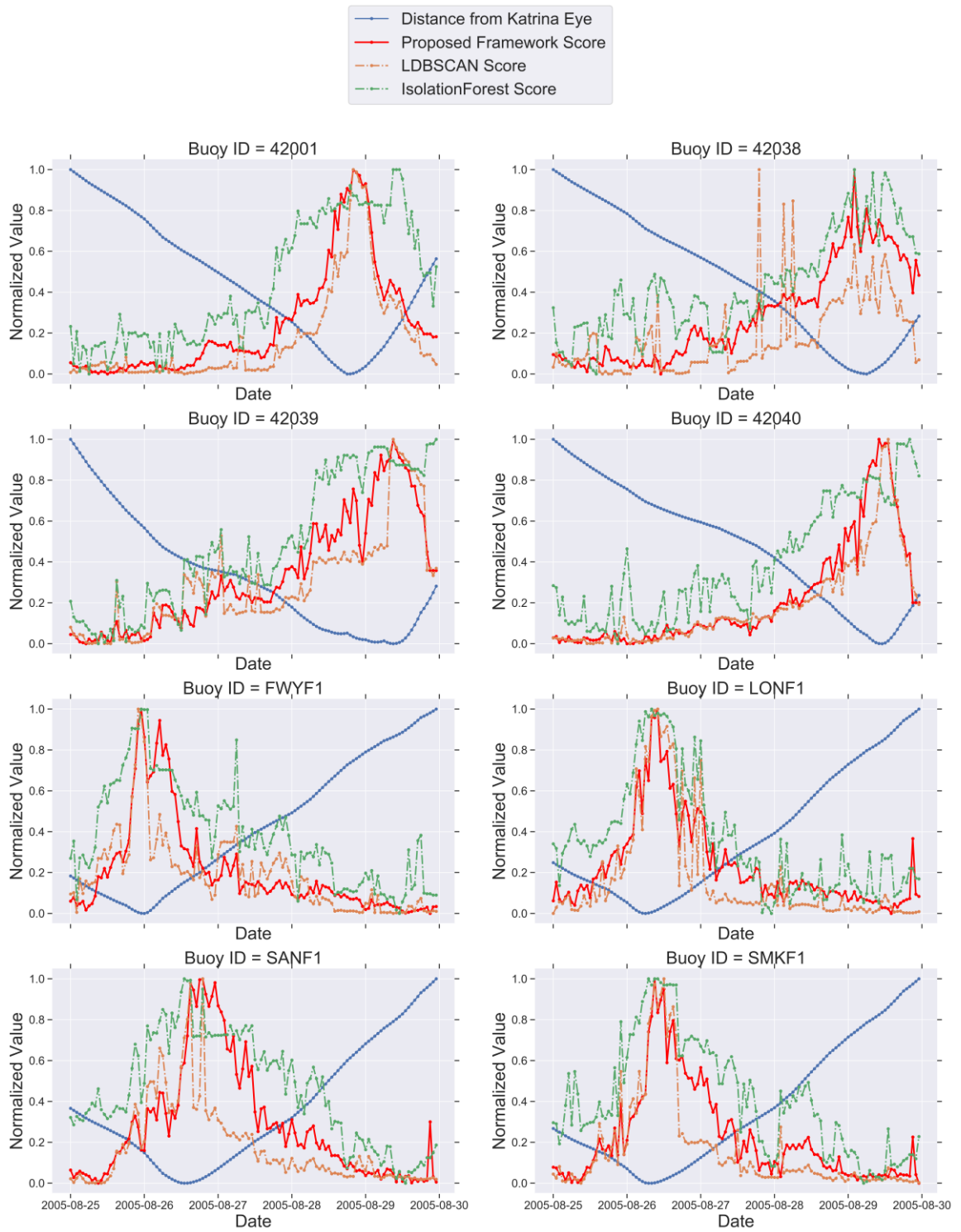


Figure 4.5 Anomaly score comparison of proposed framework, IsolationForest and LDBSCAN algorithm. Normalized anomaly scores are plotted against the normalized distance between buoys and the center of the Katrina.

4.5.2 Quantitative evaluation of models

To demonstrate more convincing evaluation, we have used a quantitative measure, *OAHII* (Outliers Association with Hurricane Intensity Index) developed by Duggimpudi et al. (2017). *OAHII* shows how much correlation exists between the outlierness scores generated by an algorithm and the actual impact of the hurricane (which is the ground truth for the given dataset). The ground truth is represented by the actual Katrina path data reported Knabb et al. (2006), which contains the best track positions and intensities reported once every 6 hours. It contains the latitude, longitude, time, wind speed and pressure information around a set of hurricane eye locations during the period from August 23rd to August 31st. We have applied linear interpolation to the ground truth data to fill missing hurricane readings and to transform it to an hourly basis like the buoy dataset.

The anomaly scores of the proposed hybrid deep learning framework (DLF) and IsolationForest (IF), LDBSCAN, LOF, and ST-BDBSCAN algorithms are compared, excluding ST-Outlier detection algorithm as it does not generate a score. Proposed framework is tested against data prepared with different window sizes to demonstrate that it is the highest performing model at each configuration. We used Euclidean distance for LOF and LDBSCAN algorithms as it gives better results. However, we gave the result for ST-BDBSCAN generated by using Mahalanobis distance since it yielded in good results as indicated by Duggimpudi et al. (2017). Results of the three *OAHII* measures, *OAHII1*, *OAHII2*, and *OAHII3* are shown in Table 4.3.

Table 4.3. Quantitative evaluation of algorithms

Measure	IF	LOF	LDBSCAN	ST-BDBSCAN	DLF (T=3)	DLF (T=6)	DLF (T=12)
<i>OAHII1</i>	0.722	0.386	0.681	0.595	0.810	0.823	0.866
<i>OAHII2</i>	0.741	0.327	0.662	0.664	0.791	0.822	0.856
<i>OAHII3</i>	0.660	0.210	0.484	0.632	0.741	0.770	0.843

OAHII measures show the effectiveness of the proposed framework over the state-of-the-art spatio-temporal anomaly detection algorithms. It is superior to other approaches in terms of both anomaly detection accuracy, which is given in Table 4.2, and correlation of scores to the ground truth, which is given in Table 4.3. According to the results given in

Table 4.3, the proposed framework performs better than the next best algorithm in detecting anomalies scoring more than 10% in all OAHII measures.

4.5.3 Quantitative evaluation of framework variants

To further investigate the effects of hyper parameters, we have tested the framework under different configurations by changing the number of CNN and LSTM layers. The following variants of the proposed framework were tested: CNN(1)-LSTM(1), CNN(1)-LSTM(2), CNN(1)-LSTM(3), CNN(2)-LSTM(1), CNN(2)-LSTM(2), CNN(2)-LSTM(3), CNN(3)-LSTM(1), CNN(3)-LSTM(2), CNN(3)-LSTM(3), numbers in parenthesis representing the number of layers. Only *OAHII1* measure is considered for this experiment. Data is preprocessed using the window size $T = 12$ and with different sliding step sizes $s = 1, 2, 3, 4, 6, 12$. The quantitative evaluation of variants is given in Table 4.4.

Table 4.4. *OAHII1* measures of framework variants with window size $T = 12$ and different sliding step sizes $s = 1, 2, 3, 4, 6, 12$.

Model Variants	T=12, s=1	T=12, s=2	T=12, s=3	T=12, s=4	T=12, s=6	T=12, s=12
CNN(1)-LSTM(1)	0.815	0.833	0.864	0.857	0.863	0.862
CNN(1)-LSTM(2)	0.864	0.841	0.834	0.866	0.853	0.841
CNN(1)-LSTM(3)	0.841	0.847	0.851	0.857	0.797	0.826
CNN(2)-LSTM(1)	0.815	0.828	0.818	0.817	0.849	0.871
CNN(2)-LSTM(2)	0.862	0.811	0.829	0.842	0.817	0.842
CNN(2)-LSTM(3)	0.851	0.860	0.832	0.803	0.841	0.839
CNN(3)-LSTM(1)	0.806	0.799	0.805	0.807	0.816	0.869
CNN(3)-LSTM(2)	0.830	0.819	0.841	0.821	0.839	0.846
CNN(3)-LSTM(3)	0.841	0.848	0.848	0.818	0.816	0.822

The best *OAHII1* measure for each framework variant is given in bold. Overall results show that for the given problem domain the network is not very sensitive to number of layers and increasing the layers does not really improve the performance. The boxplot of each framework variant is given in Figure 4.6. According to mean score of each model variant represented by the blue line in the box plot, the accuracy (which is the *OAHII1* measure) does not deviate much between different models with different network parameters. The average accuracy calculated over all framework variants with different step sizes is 0.836 and more than 10% better than the next best scoring algorithm, which is IsolationForest according to quantitative evaluation results given in Table 4.3. The core architecture of the network and the way we use the spatial and temporal context prove to be a powerful approach in spatio-temporal anomaly detection problem.

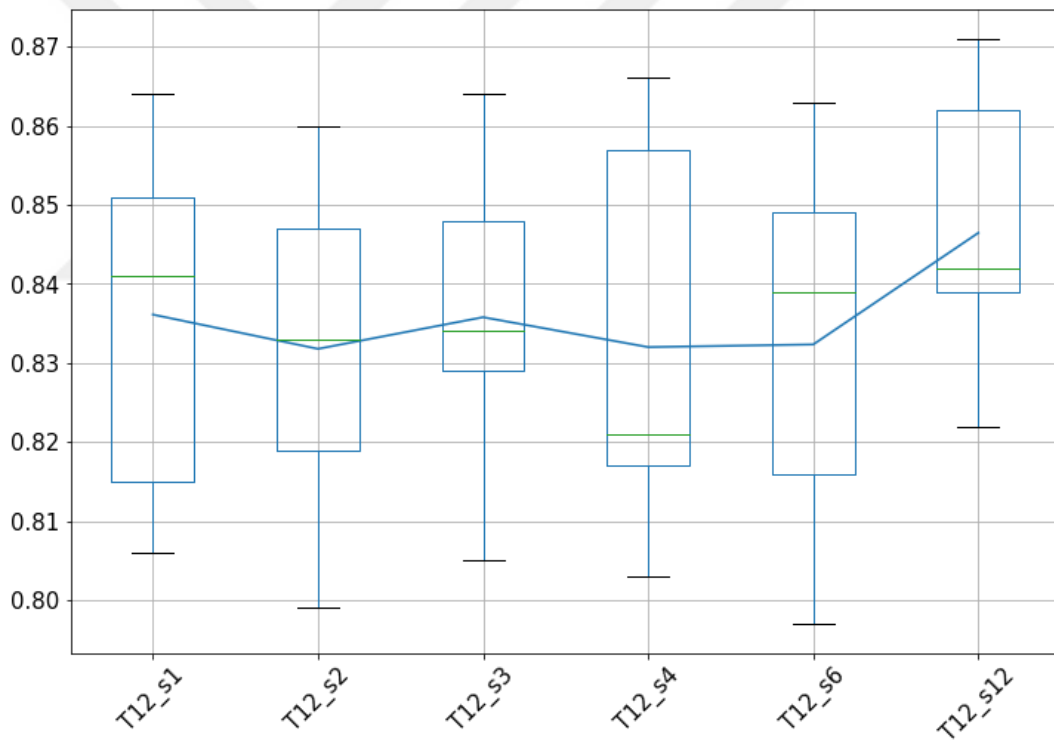


Figure 4.6 Boxplot of *OAHII1* measures of framework variants. Blue line represents mean scores.

5. EXTENSION OF THE FRAMEWORK

Multi-channel convolutional encoder used in the framework exploits the two-dimensional spatial structure of the multivariate input data, by sharing a set of weights across the input window. This results in fewer parameters than a traditional autoencoders with fully connected layers (Masci et al. 2011). Using the two-dimensional convolutional operation helps to find spatial structure between univariate signals within a multivariate dataset in each spatial region but cannot capture the spatial structures between univariate signals of different spatial regions.

The multi-channel convolutional encoder is extended by using three dimensional convolutions to capture complex spatial dependencies in each spatial neighborhood. By convolving a 3D kernel with the cube formed by stacking nearest spatial neighbors of each data point, better representative features can be extracted. This allows information across these spatially close neighbors to be connected to form feature maps, thereby capturing spatio-temporal information encoded in the neighborhood (Ji et al. 2013). In the decoding phase, the LSTM-based decoder network attempts to reconstruct the target dataset.

In most typical CNNs for image recognition, the input data is a single image with three channels for color images (R, G and B color channels) or one channel for grayscale images. In anomaly detection networks, the input data is generally a video clip consisting of multiple frames. In convolutional autoencoder based applications, T frames in the channel dimension are stuck and then fed into the autoencoder where T is the length of the sliding window. In the case of 2D convolutional autoencoders, the temporal features are rarely preserved as 2D convolution operations are performed only spatially (Hasan et al. 2016).

This idea of using 3D convolutions to extract spatio-temporal features from videos has been applied to anomaly detection problems in videos. Zhao et al. (2017) present a spatio-temporal autoencoder framework that uses 3D convolution layers for encoding and 3D deconvolution layers for decoding to extract spatio-temporal features from videos. Nogas

et al. (2018) uses a deep spatio-temporal convolutional autoencoder schema to detect falls in videos. They formulate the fall detection problem as one class classification problem. Their classification framework consists of a 3D encoder and a 3D decoder component. However, they all use semi-supervised learning approach that their models are trained only on normal data to learn normal activities of daily living without any abnormal frames. Then, they use unseen dataset to detect any abnormal frames which are considered anomaly. None of the reviewed methods is used for anomaly detection problem for non-image multivariate spatio-temporal data. In this study, 3D convolutional operations are applied on multivariate spatio-temporal data to better preserve the temporal features along with the spatial features. The input data are constructed as a hyper-cuboid by stacking spatial-temporal multivariate data frames (spatial feature maps) in the 4th dimension and 3D convolutions are performed on them.

5.1 Spatio-Temporal Encoding

The multivariate spatio-temporal input defined in Section 3.1 is encoded by a sequence of 3D convolution and 3D-max-pooling layers. Specifically, two 3D CNN blocks in the spatial encoder are used, each of which has a 3D convolutional layer, followed by a 3D max-pooling layer.

In this context, 3D convolutional operations are applied on multivariate spatio-temporal data to better preserve the temporal features along with the spatial features. The input data are re-constructed as a 3-dimensional cuboid by stacking multivariate data frames as illustrated in Figure 3.2. By applying this idea, we can accomplish dimensionality reduction both in spatial and temporal context for a given input window during the encoding phase. The main component of the extended spatio-temporal encoder is the 3D convolutional layer, which is defined as follows: the value v at position (x, y, z) of the j th feature map in the i th 3D convolutional layer, with bias b_{ij} , can be written by the following equation (Ji et al. 2013):

$$v_{ij}^{xyz} = f \left(\sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{s=0}^{S_i-1} w_{ijm}^{pqs} v_{(i-1)m}^{(x+p)(y+q)(z+s)} + b_{ij} \right) \quad (5.1)$$

where P_i , Q_i , and S_i represent the vertical (temporal depth, or window size, T), horizontal (temporal width, or number of features, d), and spatial depth (number of spatial neighbors, m) dimensions of the kernel cube w_i in the i th layer. The set of feature maps from the $(i - 1)$ th layer is indexed by m , and w_{ijm}^{pqs} is the value of the kernel cube at the position pqs connected to the m th feature map in the previous layer. The number of feature maps is defined by the number of kernel cubes at each convolution layer.

5.2 Spatio-Temporal Decoding

To improve the spatio-temporal decoding part of the framework, the convolutional LSTM (ConvLSTM) network, which is a variant of LSTM network is used. It has been introduced by Shi et al. (2015). It has been recently utilized by Chong and Tay (2017) for abnormal event detection in videos and by Patraucean (2016) for motion estimation in videos.

The major drawback of regular Long Short-Term Memory (LSTM) networks is that they are not capable of preserving the spatial information during the state transitions (Shi et al. 2015). To overcome this problem, ConvLSTM units have convolutions operations in place of matrix operations in all gates and cell outputs. As they use convolution for both input-to-hidden and hidden-to-hidden connections, they require fewer weights and yield better spatio-temporal feature encoding and decoding performance. The formulation of a ConvLSTM unit can be given by the following equations from 5.2 through 5.7:

$$f_t = \sigma(W_f * [X_t, H_{t-1}, C_{t-1}] + b_f) \quad (5.2)$$

$$i_t = \sigma(W_i * [X_t, H_{t-1}, C_{t-1}] + b_i) \quad (5.3)$$

$$\hat{C}_t = \tanh(W_c * [X_t, H_{t-1}] + b_c) \quad (5.4)$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \hat{C}_t \quad (5.5)$$

$$o_t = \sigma(W_o * [X_t, H_{t-1}, C_{t-1}] + b_o) \quad (5.6)$$

$$h_t = o_t \otimes \tanh(C_t) \quad (5.7)$$

where ' $*$ ' denotes the convolution operator and ' $\otimes$ ' denotes the Hadamard product. Equation 5.2 represents the forget gate, equations 5.3 and 5.4 are the gates where new

information (input X_t) is added, equation 5.5 combines the new and old information factored by the forget gate, whereas equations 5.6 and 5.7 give the output of the ConvLSTM unit for the next time step. The variable X_t denotes the input vector, h_t denotes the hidden state, and C_t denotes the cell state for the time step t . W 's are the trainable weight matrices and b 's are the bias vectors.



6. EXPERIMENT SET II

The world has been fighting a pandemic caused by a new type of coronavirus (SARS-CoV-2) since it was discovered in China in December 2019. Almost all countries have been affected by the novel coronavirus (COVID-19) outbreak, and Italy is one of the hardest-hit European countries. As of May 15, 2020, the total number of positive cases has reached 223,885 and the number of deaths has passed 31,000. Following the identification of the first infections on second half of February 2020 in northern Italy, authorities have put an increasing number of restrictions in place (Pepe et al. 2020). Due to the high contagiousness of the infection, this did not stop further spreading of the epidemic by asymptomatic people. The peaks of the epidemic were delayed in Central and Southern Italian regions as expected compared to Lombardy and other northern regions (Distante et al. 2020).

In the second set of experiments, the public COVID-19 data provided by the Italian Department of Civil Protection⁸ is used as a case study. The proposed hybrid deep learning framework is tested with the dataset at a resolution of the region level. One unified model with the data from northern regions is trained and with that model we track the progression of the COVID-19 epidemic in Italy and detect any anomalous trends in central and southern regions. The main assumption here is that the data generated by northern regions' experience going through the epidemic can be used to derive an anomaly detection model. As the extreme timeliness of detection is the new requirement of public health (Wagner et al. 2001), this data-driven approach may help us build an anomaly detection tool for more timely detection of COVID-19 outbreak.

As it has been shown by COVID-19 outbreak, the biggest challenge is to detect the outbreak during its early stages and mitigate its effects. The lack of an early epidemic warning system eliminated the opportunity to prohibit the epidemic spread at the initial stage. We would like to apply the proposed hybrid framework to tackle the problem of early disease outbreak detection in the midst of global health crisis. For any anomaly detection algorithm to be successful in early detection of disease outbreaks, it must

⁸ <https://github.com/pcm-dpc/COVID-19>, accessed on: May 15, 2020.

incorporate both spatial and temporal aspects of a disease (Wong et al. 2002). On the other hand, accurate monitoring of the evolution of the COVID-19 epidemic becomes extremely meaningful for the decision-making authorities to take appropriate actions against any public health crisis.

6.1 Dataset Description

The public Italian COVID-19 time series dataset provided by Italian Department of Civil Protection is used for this set of experiments. It can be downloaded from the website (COVID-19 2020) which is constructed as national response effort for coronavirus emergency. For this study, the regional dataset which shows the daily progress of new coronavirus epidemic in regions of Italy is used. The regional dataset provides detailed epidemiological figures for all 21 regions (19 regions and the 2 autonomous provinces) starting from February 24 and updated daily.

The regions dataset has 20 features as follows (translated into English): Date, country, region code, region name, latitude, longitude, hospitalized with symptoms, intensive care patients, total hospitalized patients, home isolation, total positives (current positives), change in total positive, new positives, recovered (discharged), deceased, total cases, tests performed, total number of people tested, notes in Italian, notes in English. Features "notes in Italian, notes in English, total number of people tested " were discarded for this study as they were mostly empty. As the only country in the dataset was Italy, 'country' column was also deleted.

On the website, the data format is explained as follows:

- *total positives*: Total amount of current positive cases (hospitalized patients + home confinement)
- *change in total positive*: New amount of current positive cases (total positives current day total positives previous day)
- *new positives*: New amount of current positive cases (total cases current day – total cases previous day)
- *total cases*: Total amount of positive cases

Based on those detailed descriptions of dataset, the feature "total positives" is renamed as "current positive cases" in this study to make the feature name more representative. The feature "new positives" is regarded as "daily confirmed new positive cases" and renamed as "new positive cases" for clarity. In addition to the regional epidemic data, the population data of each region is taken from the ISTAT⁹ website. By using the population information, three additional features are engineered: "total positives cases, new positive cases and deaths" on each 10,000 inhabitants. Using these engineered features, the case density information on each region has been incorporated to enrich the dataset.

Latitude and longitude are also provided for each region making this regional dataset a spatiotemporal dataset. Daily epidemic data entry for each region has two contextual attributes: a date attribute (temporal context) and latitude-longitude attribute (spatial context) which is static for each region. Beside these contextual attributes, the rest of the attributes including the engineered features are regarded as behavioral attributes and used in the model. The min-max normalization method is used to scale all behavioral attributes in the dataset into the range of [0, 1] to accelerate the learning process and to avoid large weights which cause neural networks to overfit.

6.2 Data Preparation

The regional data entries between February 24 and May 15, inclusive, are used. The model is trained with the data from northern regions which provides a complete epidemiological data in the sense that they have gone through all the peaks of COVID-19 outbreak showing a complete perspective for anomaly detection. The training dataset contains data from following northern regions: P.A. Bolzano, Emilia-Romagna, Liguria, Lombardia, Piemonte, P.A. Trento, Valle d'Aosta, Veneto, and Friuli Venezia Giulia. The data from one central region (Marche) are used as validation set; the data from one central region (Lazio), one region from southern Italy (Campania), and one island region (Sicilia) are used as test set. The total data entry for each region is 82, which means 82 days of COVID-19 epidemiological data are entered for each region.

⁹ <http://dati.istat.it/Index.aspx>, Accessed on: May 15, 2020.

The spatial attributes of all regions used in this study is given in Table 6.1. Using the latitude and longitude information, the distance matrix showing the pairwise distance of all regions used in this study is calculated. The haversine formula, which is used to measure distances on a sphere (Ingole and Nichat 2013), is used to calculate the shortest distance between regions.

Table 6.1. Information on Italian regions which are used in the experiment

Region Name	Latitude	Longitude	Population
Piemonte	45.0733	7.68069	4356406
Valle d'Aosta	45.7375	7.32015	125666
Lombardi	45.4668	9.19035	10060574
P.A. Bolzano	46.4993	11.3566	531178
P.A. Trento	46.0689	11.1212	541098
Veneto	45.4349	12.3385	4905854
Friuli Venezia Giulia	45.6494	13.7681	1215220
Liguria	44.4115	8.9327	1550640
Emilia-Romagna	44.4944	11.3417	4459477
Marche	43.6168	13.5189	1525271
Lazio	41.8928	12.4837	5879082
Campania	40.8396	14.2508	5801692
Sicilia	38.1157	13.3624	4999891

By using the spatial neighborhood of each region, a spatio-temporal multivariate data input is created for the model. The sliding window technique given in Algorithm 2 is applied to training, validation, and test datasets to build spatio-temporal multivariate subsequences. The algorithm is applied with the parameters representing the number of spatial neighbors (which is called depth in the algorithm) set to 10, the window size set to 7 representing 7-day worth of data point, and step size to 1. According to subsequence generation formula, which can be given as $\left\lceil \frac{L-T+1}{s} \right\rceil$, the maximum number of extracted subsequences can be calculated. When the input window size T is set to 7, step size s is set to 1 and the total length of the time series L is set to 82, the total of 76 multivariate subsequences are generated for each region. As the total number of behavioral attributes is 13, excluding

spatial features “region code, region name, latitude, and longitude”, and with the depth of spatial neighborhood is set to 10, we create 76x7x13x10 dimensional spatio-temporal multivariate input data from each region. By using this sliding window algorithm, data augmentation is performed by moving the start of the T-day data entry (or data window) by step size resulting in a nearly six-fold expansion of the training data.

The parameter "number of spatial neighbors" represents the number of nearest neighbors to use while building spatio-temporal multivariate subsequences. It also corresponds to the spatial dimension of the 3D CNN encoder. After data preprocessing step is completed, training, validation and test sets are created. They are 4-dimensional data matrices with the following sizes: The dimension of the training set is 684x7x13x10, the dimension of the validation set is 76x7x13x10, and the dimension of the test set is 228x7x13x10. Numbers 684, 76 and 228 represent the number of data points or observations in training, validation, and test sets, respectively. The proposed framework is trained to reconstruct the following behavioral attributes: *Hospitalized patients, intensive care patients, total hospitalized patients, home confinement, current positive cases, new positive cases, total positive cases, recovered, and deaths*. The size of the reconstruction space for the test set is 228x7x9.

We calculate the correlation coefficients for the feature "current positive cases" between every pair of districts in the training data using Pearson correlation. The correlation heatmap matrix in Figure 6.1 shows that all the neighboring regions have strong spatial correlations. Remote regions in the dataset, such as Valle d'Aosta and Marche, show weaker correlations between other regions. These results reflect that the spatial correlation of COVID-19 epidemic progression occurring in certain geographic regions at a certain spatial resolution is quite strong.

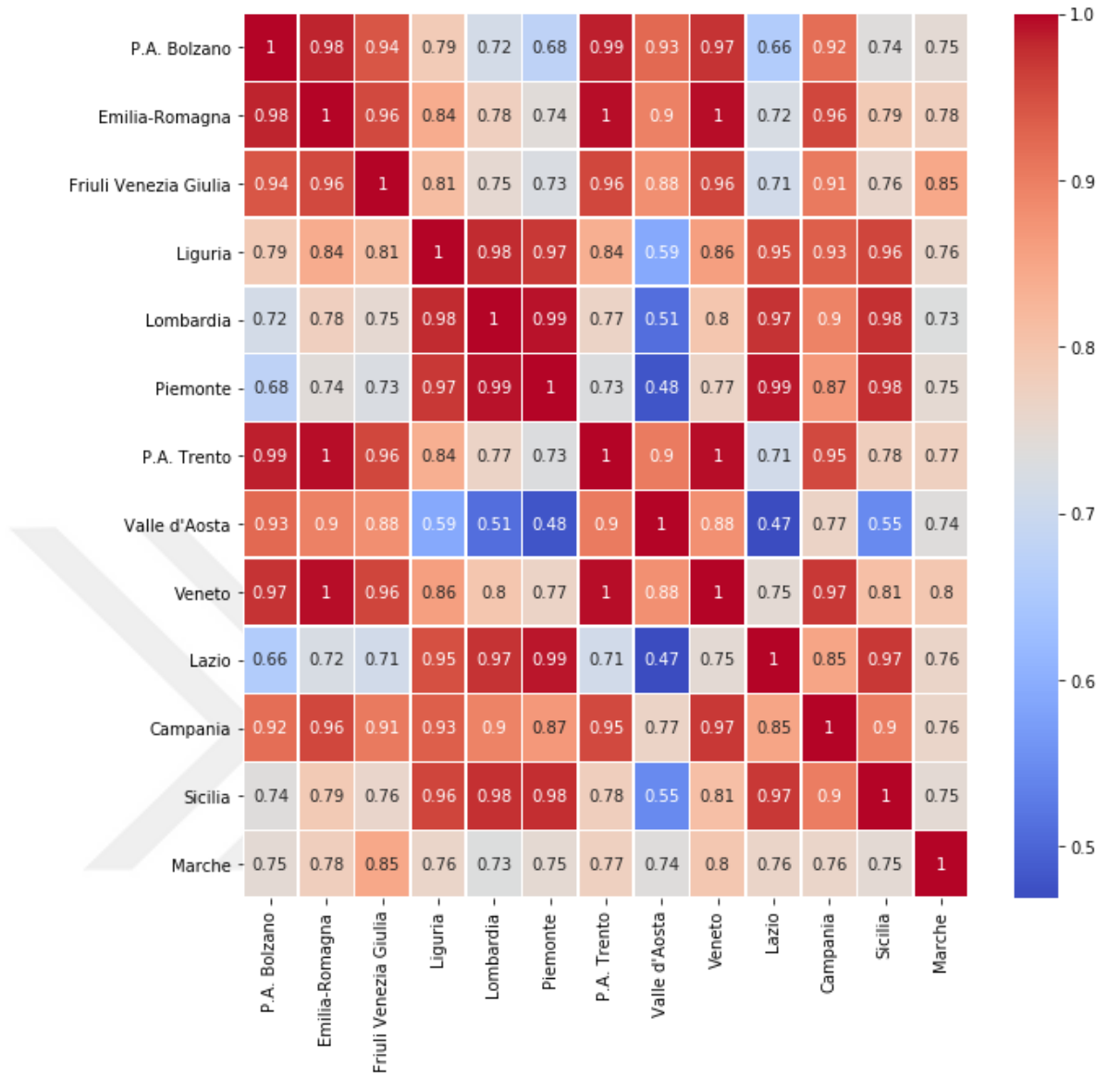


Figure 6.1 Correlation matrix visualized as heatmap. It shows the strong spatial correlation between regions for the feature “Current Positive Cases”.

6.3 Framework Tuning

Extensive experiments through grid search are executed to finalize the architecture of the framework and its hyperparameters. Specifically, we use 2 CNN blocks in the encoder component, each of which has a 3D convolutional layer, followed by a 3D max-pooling layer. Number of feature maps is set to 64 in the first block and set to 32 in the second block with padding and no striding (or with strides $1 \times 1 \times 1$). We set the kernel size to $3 \times 3 \times 5$,

where $P_i = Q_i = 3$ and $S_i = 5$ in (7), for all convolutional layers for the experiment, as these values are found to produce the best result for the dataset. The max-pooling layers have pool size of $2 \times 2 \times 2$ and strides of $1 \times 2 \times 2$ with padding. This means that the pooling operation is performed over all three dimensions: (*temporal depth* $\times$ *temporal width* $\times$ *spatial depth*). In addition, temporal width and spatial depth dimensions are reduced by a factor of 2 with every max-pooling layer. The activation function f in (7) in all hidden convolutional layers in the encoder component are set to Rectified Linear Unit (*ReLU*) non-linearity, $ReLU(x) = \max(x, 0)$, which allows the deep neural networks converge faster (Simonyan and Zisserman 2014).

The decoder component is composed of two ConvLSTM layers with the number of feature maps set to 32 and 64, respectively to preserve the symmetry of the autoencoder framework. We apply 2D convolution operation over spatial and temporal dimensions using the kernel size of 3×2 and the stride of 1×1 with padding. Batch normalization (BN) (Ioffe and Szegedy 2015) is applied to each of the ConvLSTM layers, which accelerates the training of deep neural networks. In the final layer, a fully connected neural network (FCNN) is used to reconstruct the target output. Thus, we add a layer to reshape the 4D output of final ConvLSTM layer before passing the output to the FCNN. The FCNN layer is a time distributed dense layer which applies the same fully connected operation to every time step. The number of units in the dense layer is set to $d' = 9$ and it is equal to the number of univariate time series (features) that we want to reconstruct. The number of hidden units in the FCNN layer can be adjusted according to the problem context at hand.

Activation functions of ConvLSTM units are set to hyperbolic tangent and the activation functions in the final dense layer are set to *ReLU*. Layer weights are initialized with the Glorot uniform initializer (Glorot and Bengio 2010). The deep learning framework is optimized using the Adam optimizer with learning rate set to 0.0001. It ran for 100 epochs with batch size 16. The training is regularized by weight decay (the L_2 penalty multiplier set to 1×10^{-4}) and dropout regularization for the two ConvLSTM layers (the dropout ratio set to 0.25). The model is trained to minimize the following mean absolute error (MAE) loss function:

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - \hat{x}_i| \quad (6.1)$$

where x_i and $\hat{x}_i$ represent the true value and the reconstructed value, respectively, and n is the total number of data points in each batch. The detailed architecture of the final hybrid deep learning framework is illustrated in Figure 6.2. The final framework has 299,241 trainable parameters. The data structure of each component in the trained framework is given in Table 6.2, where N represents number of data points.

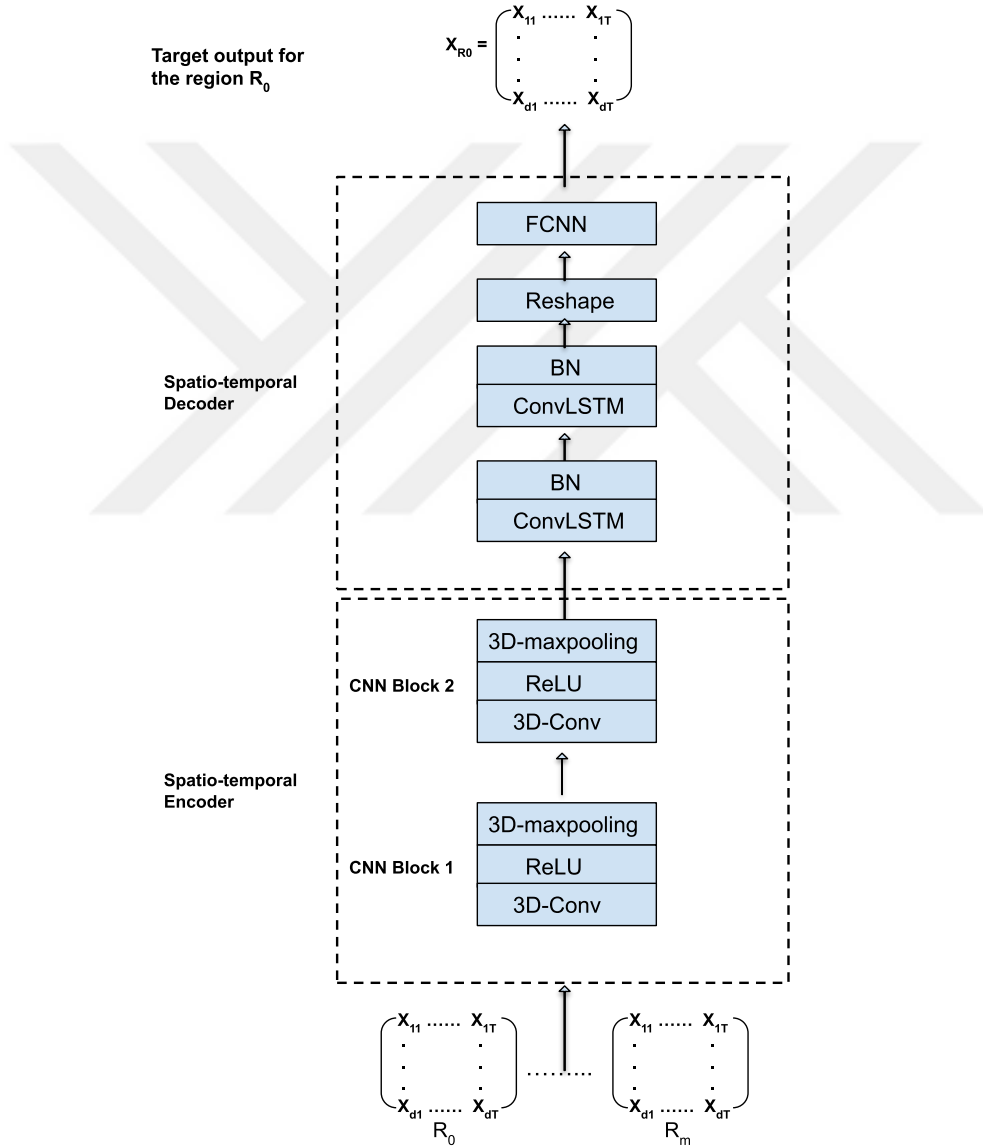


Figure 6.2 The extended framework architecture.

Table 6.2 The hybrid autoencoder framework data structure.

Layer Name	Output Shape	Param #
Input Layer (data)	$[(N, 7, 13, 10, 1)]$	0
3D CNN Encoder	$[(N, 7, 4, 3, 32)]$	95,136
ConvLSTM Decoder	$[(N, 7, 4, 3, 64)]$	197,376
Reshape Layer	$[(N, 7, 768)]$	0
FCNN Layer	$[(N, 7, 9)]$	6921

6.4 Algorithms Used for Comparison

This section summarizes the state-of-the-art methods used for comparison with the proposed approach. The ability of the proposed framework to function as an early COVID-19 outbreak detection tool is evaluated against following anomaly detection models under the following categories:

6.4.1 Statistical Models

Most detection algorithms in bio-surveillance which operate on univariate time series data have been taken from the field of quality control (Wong et al. 2005). The most common techniques include Shewhart control chart and CUMulative SUM Statistics (CUSUM) (Montgomery 2009).

Shewhart control chart comes from the quality control and originated in 1931. It uses previous data to estimate a reasonable upper limit or threshold value. If future measurements stay under the threshold value, the process is ‘under control’. New measurements which exceed the calculated threshold limit may indicate that a noteworthy change has occurred in the underlying process. In our early outbreak detection scenario, it

may indicate an anomalous daily data entry. The standard detector was trained on training dataset to obtain the mean μ and variance σ^2 . The control chart threshold value is calculated for each feature by the formula given below as defined in (Wong et al. 2005):

$$threshold = \mu + \sigma * \phi^{-1}(1 - \frac{p_value}{2}) \quad (6.2)$$

where ϕ^{-1} is the inverse to the cumulative distribution function of a standard normal, and the p -value is supplied by the user. Given a p_value of 0.5, the threshold level for the feature “**new positive cases**” can be calculated as **0.492**.

CUSUM charts are good at detecting small shift from the mean more quickly than Shewhart control charts (Neill 2009). CUSUM is calculated by taking the cumulative summation of the difference between each measured value and the estimated in-control mean value:

$$S_k = \sum_{i=1}^k (x_i - \mu) + S_{k-1} \quad (6.3)$$

where S_i is the i th cumulative sum, x_i is the i th observation and μ is the in-control mean value. It keeps a running sum of excess values over the mean each day. When this sum exceeds a threshold level, we can signal an alarm as an indication of abnormality. For a process that is under control, each measured value should be reasonably close to the mean. Thus, as long as the process remains in control, the CUSUM plot of each calculated value of S_k should be centered about zero with small fluctuations. If the process mean shifts upward, the CUSUM values for data points will eventually drift upwards.

6.4.2 Regression Based Models

Several algorithms proposed in the statistics literature have been used widely for time series prediction and anomaly detection such as autoregressive integrated moving average (ARIMA) and Exponentially Weighted Moving Average (EWMA) (Bianco et al. 2001; Yaacob et al. 2010; Yu et al. 2016; Montgomery 2009).

Standard moving average algorithm introduces lag into the original time series, which means that changes in the trend are only seen with a delay. Exponentially Weighted Moving Average (EWMA) reduces this lag effect by introducing the decay parameter and

puts more weight on more recent observations. The window (span) is chosen as 7 days.

The ARIMA model is represented by (p, d, q) model parameters which show the order of Auto-regressive (AR), the differencing component, and Moving Average (MA) respectively. The integrated part of ARIMA (the differencing component) helps in reducing the non-stationarity. The optimum parameters of this model are selected by minimizing the Akaike information criterion (AIC). The final model is built using the parameters ARIMA(4, 1, 4).

6.4.3 Distance-based Models

Distance and density-based outlier detection algorithms have also been applied to anomaly detection problem in time series and spatial data, such as Local Outlier Factor (LOF) (Breunig et al. 2000), and the Local Density-Based Spatial Clustering of Applications with Noise (LDBSCAN) (Duan et al. 2007).

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method based on local density deviation of a given dataset. It calculates the local density of a given data point with respect to its neighbors. It gives higher LOF scores to the samples that have a substantially lower density than their neighbors. For LOF model, we use 30 for number of neighbors to use for k-nearest neighbor calculations and 0.1 for contamination ratio.

LDBSCAN algorithm, is a clustering algorithm based on local density and takes the advantage of the LOF algorithm in scoring data points and identifying clusters. It is an extension to DBSCAN algorithm and is proved to be better at detecting clusters and outliers in varying density datasets. Following values are assigned to the parameters since they give the best result : $\text{MinPts}_{\text{LOF}} = 20$, $\text{MinPts}_{\text{LDBSCAN}} = 30$, $\text{LOFUB} = 5$, $pct = 0.3$. Euclidean distance is used for all proximity-based algorithms since it has generated better results compared to other distance metrics.

6.4.4 Isolation Based Model

This model detects anomalies based on the concept of isolation without employing any

distance or density measure: IsolationForest (iForest) (Liu et al. 2008; Liu et al. 2012). IsolationForest algorithm returns an anomaly score for each observation and 'isolates' anomalous points via recursive partitioning by randomly selecting a feature and then randomly selecting a split value for the selected feature. It can be represented by a tree structure and the number of splitting required to isolate a sample is used as a measure of normality. As name refers, it is an ensemble of trees doing random partitioning to detect anomalies. The number of estimators (or trees) is selected as 100 and the rate of contamination is set to 0.1.

6.4.5 Deep Learning Models

Various state of the art deep learning models which have been proven to be successful on anomaly detection problems are tested.

a) *Prediction based models*: LSTM and CNN based deep learning predictor models are used under this category. A deep stacked LSTM predictor model based on the architecture proposed by Malhotra et al. (2015) and a 1D CNN based predictor model (namely DeepAnT) proposed by Munir et al. (2019) have been employed as multivariate time series prediction based models for anomaly detection.

b) *Reconstruction based models*: Four different reconstruction based deep autoencoder architectures are tested. These architectures include a deep LSTM autoencoder architecture proposed by Malhotra et al. (2016), a deep 2D CNN based autoencoder schema proposed by Hasan et al. (2016), a deep spatio-temporal autoencoder model for anomaly detection in videos proposed by Chong and Tay (2017), and a deep 3D CNN based spatio-temporal autoencoder model proposed by Nogas et al. (2018).

To build the deep LSTM predictor model proposed by Malhotra et al. (2015), we employ a stacked LSTM network with the history window size set to 7, and the prediction window size to 1 to perform the one-step prediction. The final LSTM predictor architecture is built with 3 hidden LSTM layers (having 64, 32 and 16 units, respectively) with ReLU activation function and a final fully connected neural network (FCNN) layer for inference of target variables.

To build the CNN based predictor model, namely DeepAnT, we follow the design schema proposed by Munir et al. (2019). Each 1D convolution layer has 32 filters followed by ReLU activation function and max pooling layer. The last layer of the network is a FCNN layer in which each neuron is connected to all the neurons in the previous layer. This layer generates the final prediction of the network for the next time stamp as in LSTM based predictor.

The encoder component in the deep CNN autoencoder model, which is similar to the one proposed by Hasan *et al.* (2016), is composed of three convolutional layers: Conv1-Conv3 with 64 kernels of size 3×3 , 32 kernels of size 3×3 , and 16 kernels of size 3×3 respectively with no strides. We use a max pooling layer after the first and the second convolution layers with pool size of 2×2 and strides 1×2 with padding. The decoder component is built to maintain the symmetricity with three convolutional layers and two unpooling layers with the same number of kernels of size 3×3 .

For the deep LSTM autoencoder proposed by Malhotra et al. (2016), we use three LSTM layers in the encoder, and three LSTM layers in the decoder with a fully connected neural network as the inference layer. On the encoder side, the number of units in the LSTM layers are 64, 32, and 16. On the decoder side, the same number of units are used in reverse order to build a symmetric architecture.

The deep spatio-temporal autoencoder for abnormal event detection is built by following the architecture proposed by Chong and Tay (2017). The spatial encoder has two 2D convolutional layers with 64 and 32 kernels of size 2×2 and 3×3 , respectively. Temporal encoder-decoder component is composed of three convolutional LSTM (ConvLSTM) layers with the number of units are set to 16, 8, and 16 with the convolution kernel size of 3×3 . The spatial decoder has two deconvolutional layers with 32 and 64 kernels of size 3×3 and 2×2 , respectively. It has a final FCNN layer to generate the reconstruction of the selected test features.

To build the spatio-temporal 3D convolutional autoencoder, namely *DeepFall*, we follow the architecture proposed by Nogas et al (2018). The encoder has two layers of 3D

convolutions with stride of $1 \times 1 \times 1$ and padding. They have 16 and 8 kernels with kernel size set to $2 \times 2 \times 2$. After each convolution layer, 3D max pooling operation is applied with the stride size of $2 \times 2 \times 2$ and pool size of $2 \times 2 \times 2$ and $3 \times 3 \times 3$, respectively. The decoder has three layers of 3D deconvolutions with a stride of $2 \times 2 \times 2$ and padding. The kernel sizes are set to $3 \times 3 \times 3$, $2 \times 2 \times 2$, $2 \times 2 \times 2$, respectively.

All deep learning models including the proposed framework are trained to minimize the mean absolute error with Adam optimizer with learning rate set to 0.0001. The L_2 regularization with the penalty multiplier set to 1×10^{-4} and dropout regularization with the dropout ratio set to 0.25 are applied for training. Models are trained for 100 epochs with mini batches of size 16.

All models are implemented using Python 3.6.8 programming language. Deep learning models including the proposed framework are implemented using the TensorFlow library (Abadi et al. 2015). For LOF and IsolationForest algorithms, we use implementations available in the scikit-learn (Pedregosa et al. 2011), which is a free machine learning library for Python programming language. To build the ARIMA model, we use the *statsmodels* library (Seabold and Perktold 2010), which is a free Python module providing implementations of many different statistical models. Euclidean distance is used for all proximity-based algorithms since it has generated better results compared to other distance metrics.

6.5 Anomaly Detection

Deep learning models including the proposed framework are trained to reconstruct the following features: *Hospitalized patients*, *intensive care patients*, *total hospitalized patients*, *home confinement*, *current positive cases*, *new positive cases*, *total positive cases*, *recovered*, and *deaths*. It is a multi-step reconstruction that for each feature every step in the input window are reconstructed for the region of interest.

Each deep learning model including the proposed framework is trained using the pre-processed data from the training set which has a total of 684 spatio-temporal multivariate data points from 9 different regions. Then, the reconstruction errors are calculated for 228

data points in the test set. The mean absolute error (MAE) is used to calculate the reconstruction error for the given time period (input window) for each feature as:

$$e_{MAE}(x) = \frac{1}{T} \sum_T |x_i - \hat{x}_i| \quad (6.4)$$

where x_i is the real value and $\hat{x}_i$ is the reconstructed value at time instance t_i . Each data point in the test set represents a window of size T as the rolling window. As we set the window size T to 7 days during the data preparation process and sliding step size s to 1 day, we have calculated rolling window errors of each feature. Then, the size of the reconstruction space for the test set is 228x7x9.

In order to evaluate the performance of models, the number of days until an anomaly is detected is measured against the threshold level. In the context of this empirical study, an anomaly might mean that the COVID-19 pandemic might be moving out of control for the investigated region. According to European Centre for Disease Prevention and Control (April, 2020), the number of newly confirmed cases (or daily new positive cases) is one of the most accurate indicators of epidemic intensity. To compare the early COVID-19 outbreak detection performance, we compare the anomaly scores of models generated for the feature "new positive cases". For the prediction-based model, we use the one step prediction errors as anomaly scores. For the reconstruction-based models, we use the reconstruction errors as anomaly scores. IsolationForest, LOF and LDBSCAN models generate one anomaly score for each multivariate observation regardless of the feature monitored.

6.5 Results

6.5.1 Framework Performance

During the training process, the framework learns to reconstruct the selected features in each data point with the minimum possible error. In the case of anomalous events such as peaks of the COVID-19 pandemic, these reconstruction errors will get bigger, causing an alarm for the possible outbreak. The framework learns the normal data structure with the data from northern regions, which have gone through the pandemic earlier than other

regions. To be able to detect the COVID-19 outbreak as early as possible, the framework must learn what the normal is when it is trained with highly contaminated data. As there is no label indicating anomalous events, the framework learns the distinctive patterns of abnormal events using the data from the nearest spatial neighbors.

One unified deep learning model is trained using the training set, which has a total of 684 spatio-temporal multivariate data points from 9 different regions. Then, this model is used to calculate the reconstruction errors of each 228 data points in the test set. As we set the window size to 7 days during the data preparation process and sliding step size to 1 day, we calculate the rolling window errors of each feature.

The reconstructed values and reconstruction errors for features "hospitalized patients, current positive cases, new positive cases, recovered cases, deaths, and total positive cases" on test data are plotted against the real case numbers in Figures from 6.3 to 6.5. Test results for the region Lazio are given in Figure 6.3; test results for the region Campania are given in Figure 6.4; and test results for the region Sicilia are given in Figure 6.5. In these figures, the real data are depicted in blue; the reconstructions are depicted in orange; and the reconstruction errors are depicted in red. All values are min-max normalized into the range of $[0, 1]$ before plotting.

It can be seen in these plots that there is a subtle increase in the model error when the real case counts have peaks as each region goes through the COVID-19 pandemic. The progression of the COVID-19 pandemic in each test region is quite different: Lazio hits the peak of the epidemic earlier, and goes through a larger wave with many peaks, while Campania and Sicilia have shorter waves with a smaller number of peaks. Sicilia has the least number of peaks compared to other regions. The framework captures the overall structure of the real data and gives a good error margin to enable the early detection of the outbreak in each test region.

Real case counts and model errors for the region Lazio

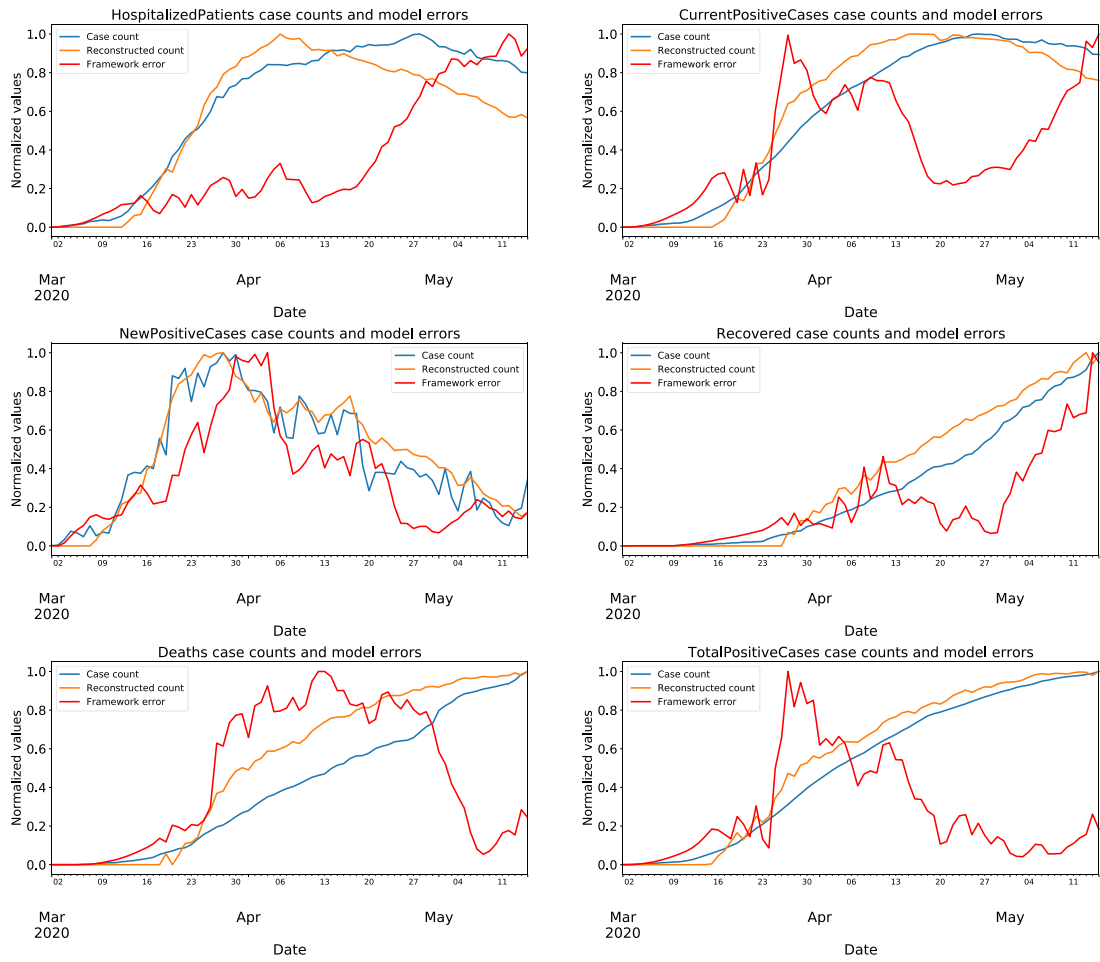


Figure 6.3 Model test results for the region Lazio are given for features HospitalizedPatients, CurrentPositiveCases, NewPositiveCases, Recovered cases, Death cases, and TotalPositiveCases. Real case counts are plotted in blue, reconstructed case counts are plotted in orange, and reconstruction errors are plotted in red.

Real case counts and model errors for the region Campania

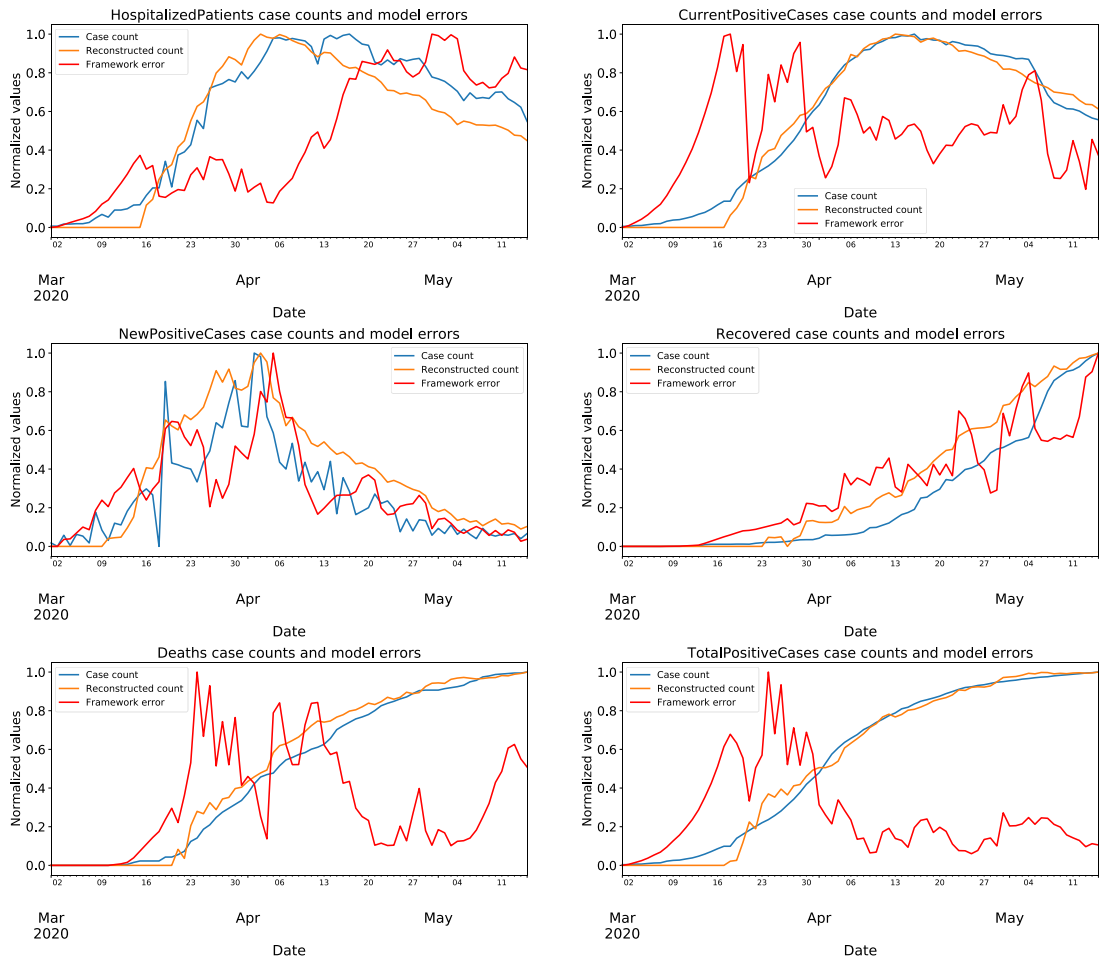


Figure 6.4 Model test results for the region Campania are given for features HospitalizedPatients, CurrentPositiveCases, NewPositiveCases, Recovered cases, Death cases, and TotalPositiveCases. Real case counts are plotted in blue, reconstructed case counts are plotted in orange, and reconstruction errors are plotted in red.

Real case counts and model errors for the region Sicilia

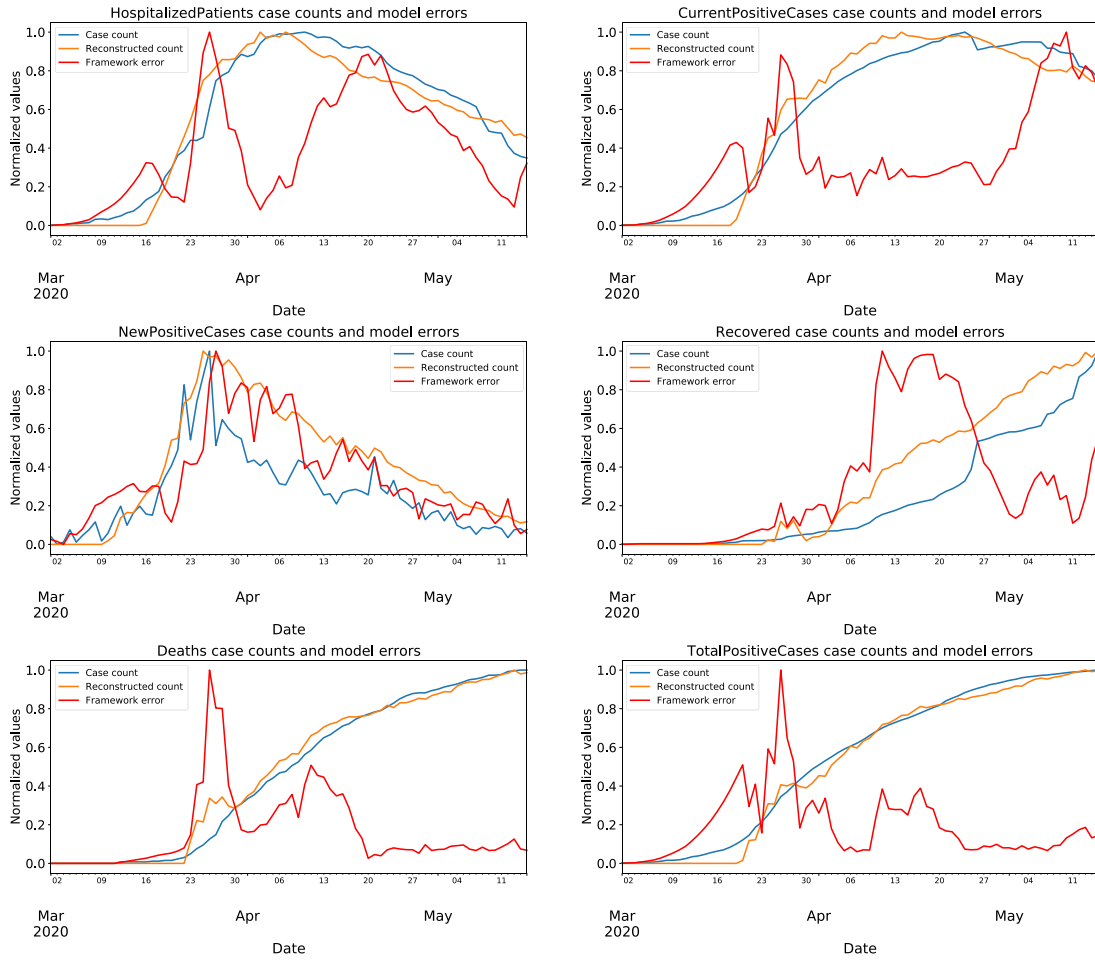


Figure 6.5 Model test results for the region Sicilia are given for features HospitalizedPatients, CurrentPositiveCases, NewPositiveCases, Recovered cases, Death cases, and TotalPositiveCases. Real case counts are plotted in blue, reconstructed case counts are plotted in orange, and reconstruction errors are plotted in red.

6.5.2 Detection Time Comparison

To compare the early outbreak detection performance of the proposed framework with other models, anomaly detection time in days is calculated. The Shewhart control chart value for the feature “new positive cases”, which is calculated as 0.492 for p_value set to 0.5, is used as the threshold level for an alarm. Outbreak detection date is calculated as the date of the first alert raised when the alarm level passes the threshold line. Table 6.3 shows

a comparison of all models based on the early outbreak detection performance. Best performing models on all three regions are highlighted. In this test dataset, the anomalies are not just the spike or point anomalies. They are contextual anomalies; and there is a trend in them showing the progression of the COVID-19 pandemic in each region. This makes this dataset difficult for an anomaly detection algorithm as contamination rate is very high in the training set.

Table 6.3. Outbreak detection date of models based on the feature “daily new positive cases”. Results for three Italian regions in the test set are given.

Model Name	Lazio	Campania	Sicilia
Shewhart Control Chart	18 March	19 March	22 March
CUSUM	12 March	8 March	12 March
ARIMA	13 March	19 March	21 March
EWMA	13 March	19 March	22 March
IsolationForest	13 March	15 March	18 March
LOF	13 March	15 March	17 March
LDBSCAN	12 March	18 March	16 March
LSTM Predictor	13 March	19 March	8 March
LSTM Autoencoder	13 March	19 March	19 March
CNN Autoencoder	13 March	19 March	22 March
Chong and Tay	16 March	19 March	21 March
DeepAnT	12 March	8 March	4 March
Nogas <i>et al.</i>	13 March	19 March	21 March
Proposed Framework	4 March	5 March	1 March

The proposed framework outperforms all compared models in early outbreak detection for all test regions. Statistical models show similar performance on the regions Campania and Sicilia. LDBSCAN algorithm shows better early outbreak detection performance on the regions Lazio and Sicilia compared to IsolationForest and LOF. CUSUM, on the other hand, performs significantly better than other statistical and distance-based models. In general, other deep learning models show similar performance on each test region except DeepAnT, which shows better performance than other compared deep learning models.

We compare our model with the best performing models, namely the CUSUM, DeepAnT, and LDBSCAN. Figure 6.6 to 6.8 illustrate the alarms generated by models, which are plotted against the standard threshold level and daily real case counts for the parameter

“new positive cases”. The case counts and model alarm values (or anomaly scores) are min-max normalized into the range of $[0, 1]$ before plotting. When an alarm value passes the threshold line, it is plotted in red to illustrate an outbreak signal.

When we look at plots for the region Lazio in Figure 6.6, the alarm level of the proposed framework passes the threshold line on March 4. It generates strong alarm signals through the wave of the COVID-19 pandemic. CUSUM, LDBSCAN and DeepAnT send first alarm signals on March 12, which is the date national lockdown was announced in Italy. Figure 6.7 shows the test results for the region Campania. The proposed model can detect the early upswing trend in the epidemic while other models fall behind in detecting the outbreak. The proposed model gives the first outbreak signal which passes the threshold level on March 5, while CUSUM and DeepAnT give on March 8, and LDBSCAN gives the signal on March 18. The region Campania is very challenging for early outbreak detection as the upswing trend starts very late and suddenly in mid-March. Figure 6.8 shows the test results for the region Sicilia. The proposed framework sends the first outbreak signal on March 1. It is followed by the DeepAnT model, which sends the first outbreak signal on March 4, followed by the CUSUM model, which sends the first outbreak signal on March 12, and followed by LDBSCAN, which sends the first outbreak signal on March 16. LDBSCAN shows good outbreak detection performance on Lazio and Campania but fails on the region Sicilia. DeepAnT is very successful in early outbreak detection in all regions, but significantly better on the region Sicilia compared to other base models. However, despite the downward trend of the pandemic in the region Sicilia after April 15, DeepAnT continues to send outbreak signals.

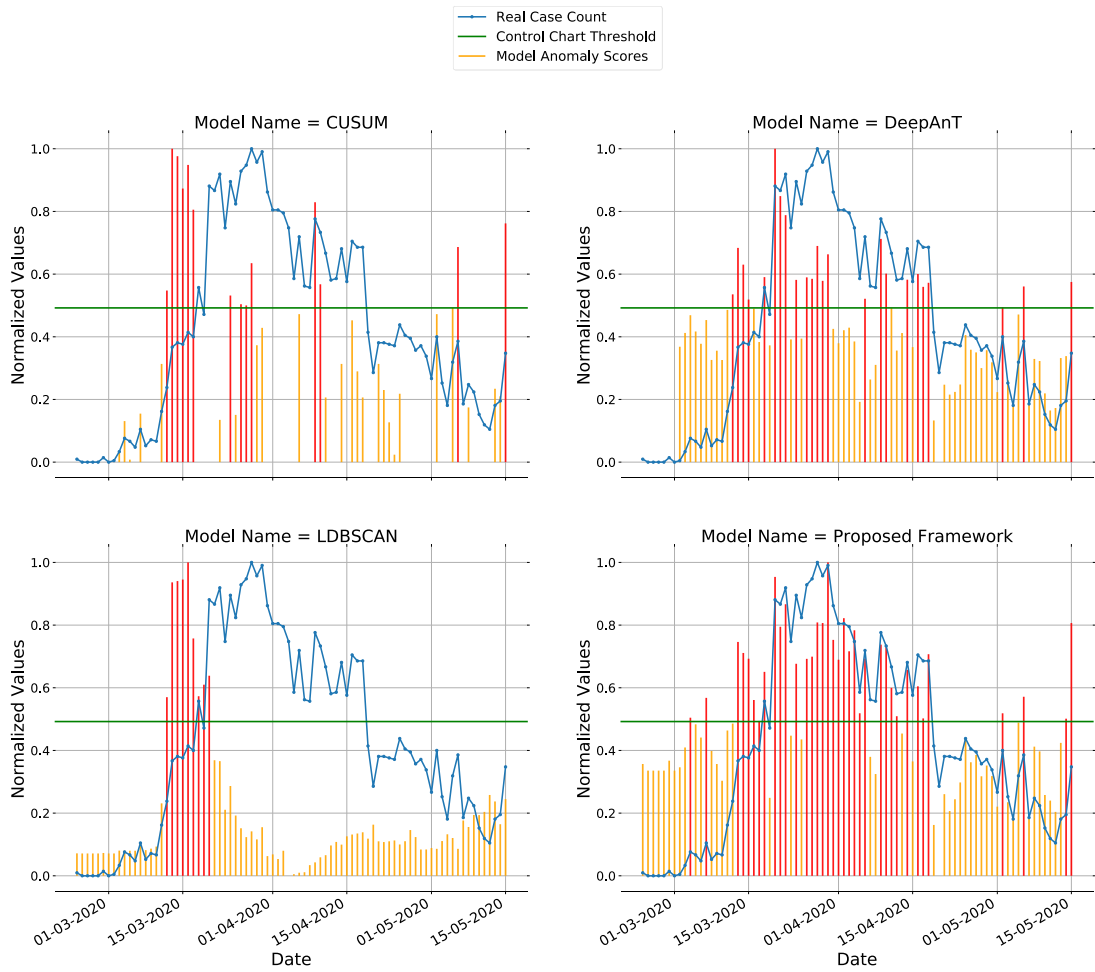


Figure 6.6 Region Lazio: Generated alarms of top performing models are plotted against daily real case counts for the parameter “new positive cases”. When a generated alarm signal passes the threshold line, it is plotted in red. In this case, we can signal an alarm for an outbreak for the region. Shewhart control chart value is plotted as the threshold level.

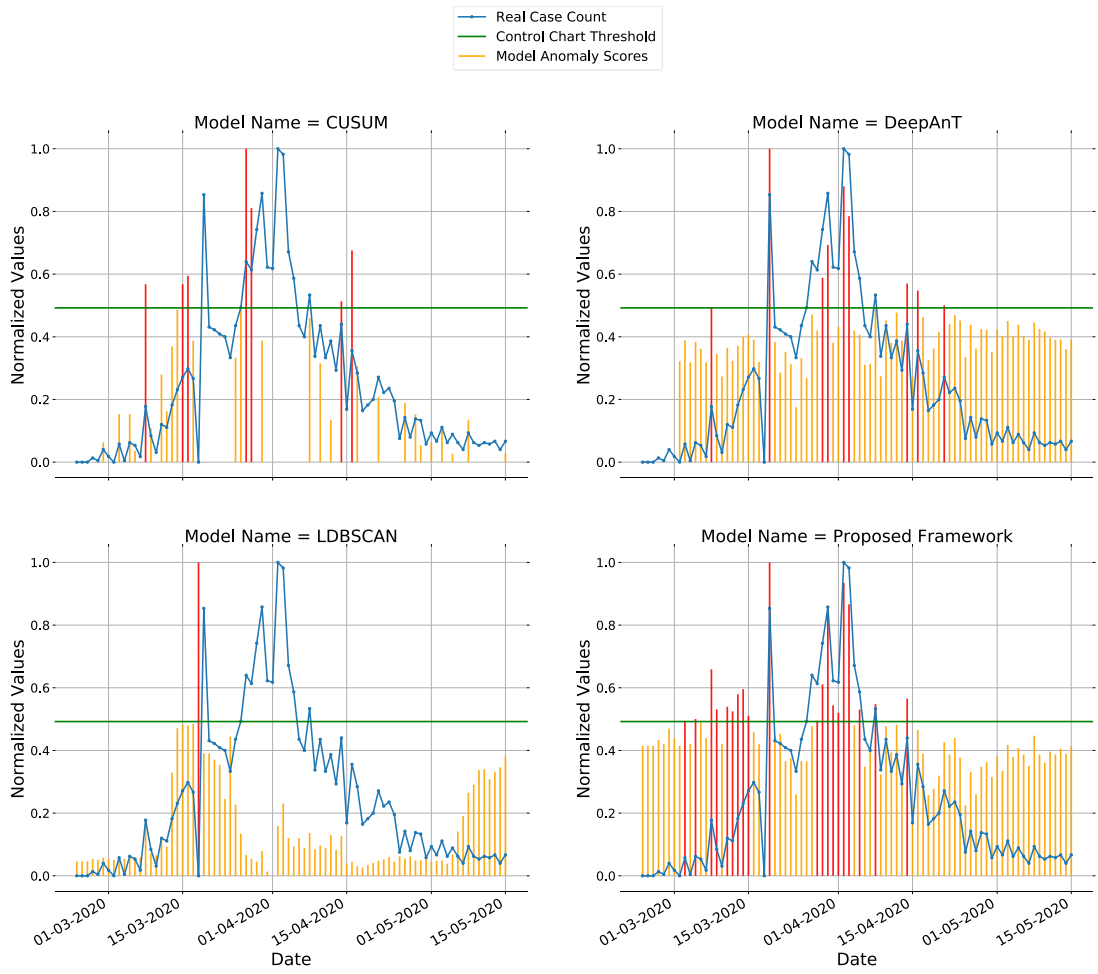


Figure 6.7 Region Campania: Generated alarms of top performing models are plotted against daily real case counts for the parameter “new positive cases”. When a generated alarm signal passes the threshold line, it is plotted in red. In this case, we can signal an alarm for an outbreak for the region. Shewhart control chart value is plotted as the threshold level.

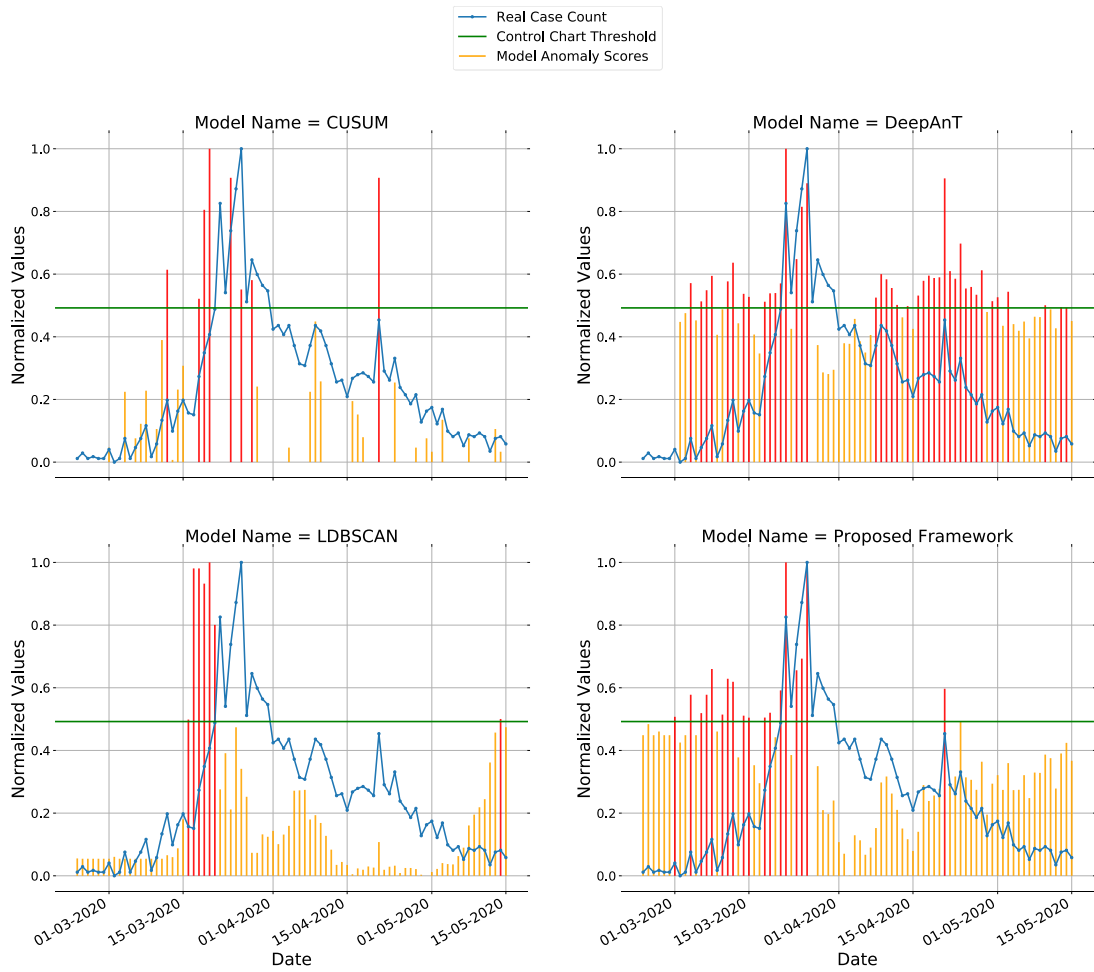


Figure 6.8 Region Sicilia: Generated alarms of top performing models are plotted against daily real case counts for the parameter “new positive cases”. When a generated alarm signal passes the threshold line, it is plotted in red. In this case, we can signal an alarm for an outbreak for the region. Shewhart control chart value is plotted as the threshold level.

It can be observed that our model has the capability of detecting abnormal upswing trends in COVID-19 pandemic waves in each test region. The main advantage of the proposed framework is that once it is trained on the training set, it does not need the history of the tested region to detect any anomalous event. It can detect point anomalies as well as contextual anomalies in test regions. It is better at tracking the abnormal events and in detecting every major peak throughout the wave of the COVID-19 pandemic in each test region. What makes the proposed framework different from other deep learning-based models is the way it uses the spatio-temporal data. The nearest neighbor's data, which is weighted based on the distance to the region, is exploited to extract the best spatio-temporal features. This makes the framework robust against noise and anomalies in the dataset.



7. CONCLUSIONS

In this thesis, a deep learning framework is presented for unsupervised detection of anomalies in multivariate spatio-temporal data. A novel way of data pre-processing technique for the non-image multivariate spatio-temporal data is also presented which uses the nearest spatial neighborhood data. The 3D cuboid data is formed by stacking the data from the nearest spatio-temporal neighbors of each multivariate data point.

For the proposed unsupervised setting, no labeled dataset is required to train the framework for anomaly detection problem. The framework is designed to be trained in a truly unsupervised fashion with the full absence of labels, for either normal or abnormal data. The proposed approach is robust enough to learn the underlying dynamics even if the training dataset is highly contaminated with anomalies (more than 5%).

In all distance/clustering-based algorithms, the biggest challenge is to combine the contextual features along the spatial and temporal dimensions in a meaningful way. In the proposed approach, we handle spatial and temporal context by different deep learning components as these contextual variables refer to different types of dependencies. The proposed framework requires no prior knowledge on anomalies such as the distribution and types of anomalies.

The proposed framework is an autoencoder based anomaly detection framework. The hybrid framework structure is based on the idea of combining various deep neural network components. It has been successfully applied to multivariate time series forecasting, face detection, and video classification. However, it has not yet been applied to unsupervised anomaly detection problem for non-image multivariate spatio-temporal data. Our proposed framework is composed of three stages: The first stage is the pre-processing of the multivariate spatio-temporal data so that the deep autoencoder network can exploit the spatial and temporal contexts jointly. The second stage is the data reconstruction stage, which is executed by a deep hybrid autoencoder network. The third stage is the anomaly detection stage, which is performed based on the reconstruction

error. It is designed to be trained in a truly unsupervised fashion for anomaly detection in non-image spatio-temporal datasets.

The framework is composed of two main components: Encoder and decoder. The first component is a multi-channel convolutional neural network encoder (MC-CNN-Encoder) to learn representative spatio-temporal features of the multivariate data. It applies three dimensional convolutional operations on the three-dimensional spatio-temporal data which are crafted by a newly proposed data generation algorithm. This new algorithm uses nearest spatial neighbors of each data point to construct the spatio-temporal multivariate data cube for the framework. It applies sliding window based on given parameters, history window size and the sliding step size. The robustness of the proposed approach comes from the fact that it uses the neighbors' data of each data point to extract the lower-dimensional spatio-temporal features. This makes the framework to learn the structure of the underlying process without memorizing the noise and anomalies which may exist in the training dataset.

The second component is Long-Short Term Memory based decoder (LSTM-Decoder) to reconstruct the part of the input sequence using the spatio-temporal latent space. MC-CNN-Encoder and LSTM-Decoder are combined into one unified framework that is trained jointly. CNN layers provide better spatial features into LSTM network by improving the framework's anomaly detection performance. LSTM Decoder component reconstructs the input sequential data and generates a reconstruction error for each input data. The point-specific reconstruction error provides the anomaly score of that point. The biggest advantage of proposed model over distance-based models is that autoencoder approach enables to analyze anomalies in detail. Computing the anomaly score for each data attribute helps in anomaly diagnosis task such as root cause identification.

To verify the performance of the framework, two sets of experiments are conducted on two different contextual datasets. For the first set of experiments, the year 2005 Gulf of Mexico (West) buoy dataset from National Data Buoy Center¹⁰ is used. The Hurricane Katrina best track path data (Knabb et al. 2006) is used as ground truth for the

¹⁰ <https://www.ndbc.noaa.gov/>

experiments. In this set of experiments, results demonstrate that the proposed hybrid deep learning framework achieves superior results by achieving at least 10% improvement in accuracy over the methods used in comparison.

In order to further improve the anomaly detection performance, the extended version of the proposed framework is introduced which is built by different neural network components. The encoder component is replaced with a 3D convolutional neural network (CNN) based spatio-temporal encoder, and the decoder component is replaced with a variant of LSTM network, a convolutional LSTM (ConvLSTM) network.

For the second set of experiments, the public COVID-19 regional data provided by the website of Italian Department of Civil Protection is used. In this set of experiments, the proposed framework is compared with various anomaly detection algorithms under five different categories: statistical models (Shewhart Control Chart, CUSUM), regression based prediction models (ARIMA, EWMA), distance based models (LOF and LDBSCAN), isolation based models (IsolationForest), and various state-of-the-art deep learning based models: Prediction based models (LSTM and CNN based deep learning predictor models) and reconstruction based models (four different reconstruction based deep autoencoder architectures are tested). These architectures include a deep stacked LSTM predictor model based on the architecture proposed by Malhotra et al. (2015), a 1D CNN based predictor model (namely DeepAnT) proposed by Munir et al. (2019), a deep LSTM autoencoder architecture proposed by Malhotra et al. (2016), a deep 2D CNN based autoencoder schema proposed by Hasan et al. (2016), a deep spatio-temporal autoencoder model for anomaly detection in videos proposed by Chong and Tay (2017), and a deep 3D CNN based spatio-temporal autoencoder model proposed by Nogas et al. (2018).

Results show that proposed framework achieves significant improvement on early outbreak (anomaly) detection performance for spatio-temporal data as compared to the state-of-the-art deep learning architectures. It outperforms other deep learning models in both early detection and tracking the COVID-19 outbreak. Experiments have shown that

the framework is capable of handling the small amount of data event if the contamination level is too high as in the case of COVID-19 Italy dataset.

The contributions can be summarized as follows:

- 1) To the best of our knowledge, the proposed framework, which is composed of a novel data crafting and a hybrid deep learning model, is the first attempt in solving unsupervised anomaly detection problem which is designed specifically for non-image multivariate spatio-temporal data.
- 2) Unlike proximity-based methods, the proposed model can be easily implemented on data with mixed-type behavioral attributes and scales well to multivariate datasets with high number of features.
- 3) The proposed model is designed to be trained in a truly unsupervised fashion without labels for either normal or abnormal data.
- 4) It achieves good generalization capabilities in scenarios where the training data are scarce and contaminated with anomalies. In the case study, only 82 daily data entries (data points) are available for each region. Even these contaminated outbreak data are sufficient to build a robust anomaly detection model due to its effective architecture to exploit spatio-temporal neighborhood data.
- 5) The biggest challenge in anomaly detection for spatio-temporal data is to combine the contextual attributes in a meaningful way. In the proposed hybrid approach, spatial and temporal contexts are handled by different deep learning components as these contextual variables refer to different types of dependencies.

The proposed approach stands out from other traditional and deep learning-based models as it is capable of handling noise and missing values better. In summary, it is simple to implement, outperforms existing competitors, and gives particularly good result even under the noisy conditions. According Machine Learning literature, this is the first deep learning-based approach to solve the unsupervised anomaly detection problem specifically designed for non-image multivariate spatio-temporal dataset.

I believe that this approach is an important improvement towards applying deep learning in unsupervised learning problems. This thesis study can be improved in several

directions. First, to enhance the anomaly detection accuracy of the framework, the attention mechanism can be added to tackle the weaknesses in analyzing long sequences. By letting the decoder component to have an attention mechanism, the framework can better detect collective anomalies in long ranging multivariate spatio-temporal data. Second, optimization-based techniques to select framework's hyper-parameters for each problem domain can be used to further enhance the performance. Finally, I would like to conduct research on more efficient data imputation techniques to improve the anomaly detection process for real life datasets.



REFERENCES

- Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Available: <https://www.tensorflow.org>
- Adam, N. R., Janeja, V. P., & Atluri, V. (2004). Neighborhood-based Detection of Anomalies in High-Dimensional Spatio-temporal Sensor Datasets. ACM SAC Conference.
- Aggarwal, C. C. (2017). Outlier analysis. Springer International Publishing.
- Aggarwal, C. C. (2018). Neural Networks and Deep Learning. Springer International Publishing.
- Aggarwal, C. C. & Reddy, C. K. Data Clustering: Algorithms and Applications, 1st ed.; CRC Press: Boca Raton, FL, USA, 2013; pp. 88-106.
- Akhilomen J. (2013) Data Mining Application for Cyber Credit-Card Fraud Detection System. In: Perner P. (eds) Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2013. Lecture Notes in Computer Science, vol 7987. Springer, Berlin, Heidelberg.
- Aleskerov, E., Freisleben, B., & Rao, B. (1997). CARDWATCH: A Neural Network based Database Mining System for Credit Card Fraud Detection. In Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), pp. 220–226.
- Bengio, Y. (2009). Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, 2(1), 1–127. Doi:10.1561/22000000006.
- Bianco, A. M., Ben, M.G., Martnez, E.J., Yohai, V. J. (2001). Outlier Detection in Regression Models with ARIMA Errors Using Robust Estimates. Journal of Forecasting, vol. 20, no. 8, pp. 565–579.
- Birant, D. & Kut, A. (2006). Spatio-temporal outlier detection in large databases. CIT. Journal of Computing and Information Technology, vol. 14, no. 4, 2006, pp. 291-297.
- Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93-104.
- Chan, P. K. & Stolfo, S. J. (1998). Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, p. 164–168.

- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, Vol. 41(3), Article 15.
- Cheng, T. & Li., Z. (2004). A Hybrid Approach to Detect Spatialtemporal Outliers. In *Proc. 12th Int. Conf. on Geoinformatics – Geospatial Information Research*.
- Cheng, T. & Li., Z. (2006). A Multiscale Approach for Spatio-temporal Outlier Detection. *Transactions in GIS*, 10(2), pp. 253–263.
- Cheng, H., Tan, P. N., Potter, C., & Klooster, S. (2009). Detection and Characterization of Anomalies in Multivariate Time Series. *SIAM Conference on Data Mining*.
- Chong, Y.S. & Tay, Y.H. (2017). Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks*, Springer, pages 189–196.
- Christiansen, P., Nielsen, L.N., Steen, K.A., Jørgensen, R.N., Karstoft, H. (2016) DeepAnomaly: Combining Background Subtraction and Deep Learning for Detecting Obstacles and Anomalies in an Agricultural Field. *Sensors*, 16(11), 1904. Doi:10.3390/s16111904.
- COVID-19 Data Italy website by Italian Department of Civil Protection, URL: <https://github.com/pcm-dpc/COVID-19>, Accessed on: May 15, 2020.
- Derrig, R. A. (2002). Insurance Fraud. *The Journal of Risk and Insurance*, vol. 69(3), p. 271-287.
- Desforges, M. J., Jacob, P. J., & Cooper, J. E. (1998). Applications of probability density estimation to the detection of abnormal conditions in engineering. *Proceedings of the Institution of Mechanical Engineers*, vol. 212(8), pp. 687–703.
- Distante, C., Pereira, I. G., Goncalves, L.M.G., Piscitelli, P., & Miani, A. (2020). Forecasting Covid-19 Outbreak Progression in Italian Regions: A model based on neural network training from Chinese data. Preprint from medRxiv; Doi:10.1101/2020.04.09.20059055.
- Duan, L., Xu, L., Guo, F., Lee, J., & Yan, B. (2007). A local-density based spatial clustering algorithm with noise. *Information Systems*, vol. 32 (7), pp. 978-986.
- Duggimpudi, M. B., Abbady, S., Chen, J., & Raghavan, V.V. (2017). Spatio-temporal outlier detection algorithms based on computing behavioral outlierness factor. *Data Knowledge Engineering*, Vol. 122, 1-24.
- Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226-231.

- European Centre for Disease Prevention and Control.(2020)."Strategies for the surveillance of COVID-19," Stockholm, 9 April 2020. [Online]. Available:<https://www.ecdc.europa.eu/sites/default/files/documents/COVID-19-surveillance-strategy-9-Apr-2020.pdf>.
- Fujimaki, R., Yairi, T., & Machida, K. (2005). An Approach to Spacecraft Anomaly Detection Problem using Kernel Feature Space. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining.
- Fujimaki, R. (2008). Anomaly Detection Support Vector Machine and Its Application to Fault Diagnosis. Eighth IEEE International Conference on Data Mining.
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proc. AISTATS, pp. 249-256.
- Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning, 1st ed.; MIT Press: Cambridge, MA, USA, pp. 493-549.
- Graves, A. & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of International Conference on Machine Learning, pp. 1764–1772.
- Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier Detection for Temporal Data: A Survey. IEEE Transactions on Knowledge and Data Engineering, 26(9), pp. 2250-2267.
- Gupta, M., Sharma, A. B., Chen, H., & Jiang, G. (2013). Context-Aware Time Series Anomaly Detection for Complex Systems. In Workshop Notes, Second Workshop on Data Mining for Service and Maintenance, SIAM International Conference on Data Mining, p. 14.
- Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K., Davis, L.S. (2016). Learning temporal regularity in video sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 733–742.
- Hazel, G. G. (2000). Multivariate Gaussian MRF for multispectral scene segmentation and anomaly detection. IEEE Transactions on Geoscience and Remote Sensing, 38(3), pp. 1199–1211.
- Hecht-Nielsen, R. (1995). Replicator Neural Networks for Universal Optimal Source Coding. Science, 269, pp. 1860–1863.
- Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, vol. 313(5786):504-7. Doi:10.1126/science.1127647.
- Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, vol. 9 (8), p. 1735–1780.

- Hodge, V. & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2), pp. 85–126.
- Ingole, P.V., & Nichat, M. K. (2013). Landmark based shortest path detection by using Dijkstra Algorithm and Haversine Formula. *International Journal of Engineering Research and Applications (IJERA)*, vol. 3, no. 3, pp.162-165.
- Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167v3*.
- Japkowicz, N., Hanson, S., Gluck, M. (2000). Nonlinear autoassociation is not equivalent to PCA. *Neural Comput.*, 12, pp. 531–545.
- Ji, S., Xu, W., Yang, M., Yu, K. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, Jan. 2013; Doi:10.1109/TPAMI.2012.59.
- Jolliffe, I.T. (2002). *Principal Component Analysis*, 2nd ed.; Springer-Verlag: New York, USA, pp. 1-76.
- Kim, T.-Y. & Cho, S.-B. (2018). Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications*, vol. 106, p. 66-76.
- Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *The 3rd International Conference for Learning Representations*, *arXiv:1412.6980v9*.
- Knabb, R. D., Rhome, J. R., & Brown, D. P. (2006). Tropical cyclone report: Hurricane Katrina, August 23-30, 2005. National Hurricane Center. https://www.nhc.noaa.gov/data/tcr/AL122005_Katrina.pdf. Accessed 16 October 2019.
- Kou, Y., Lu, C.-T., & Chen, D. (2006). Spatial Weighted Outlier Detection. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, p. 614–8.
- Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems Conference 2012*, p. 1097–1105.
- Lakhina, A., Crovella, M., & Diot, C. (2005). Mining Anomalies using Traffic Feature Distributions. *ACM SIGCOMM Computer Communication Review*, vol. 35(4), pp. 217–228.
- Lanorte A., Manzi T., Nolè G., Lasaponara R. (2015) On the Use of the Principal Component Analysis (PCA) for Evaluating Vegetation Anomalies from LANDSAT-TM NDVI Temporal Series in the Basilicata Region (Italy). In *Computational Science and Its Applications, ICCSA 2015. Lecture Notes in Computer Science*, vol. 9158. Springer, Cham.

- Leemput, K. V., Maes, F., Vandermeulen, D., Suetens, P., & Colchester, A. (2001). Automated Segmentation of Multiple Sclerosis Lesions by Model Outlier Detection. *IEEE Transactions on Medical Imaging*, vol. 20(8), p. 677–688.
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the Eighth IEEE International conference on data mining, ICDM '08*, pp. 413–422.
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Trans. Knowl. Discov.* 2012, 6(1), 31–39.
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 3431–3440.
- Lu, C.-T., Chen, D., & Kou, Y. (2003). Algorithms for Spatial Outlier Detection. In *Proceedings of 3rd International Conference on Data Mining*, pp. 597-600.
- Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In *Proc. 23rd European Symposium On Artificial Neural Networks Computational Intelligence and Machine Learning*, pp. 89-94.
- Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *ICML 2016, Anomaly Detection Workshop*, arXiv:1607.00148v2.
- Masci, J., Meier, U., Ciresan, D., Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In: *International conference on artificial neural networks*. Springer, pp 52–59.
- Montgomery, D.C. (2009). *Introduction to Statistical Quality Control*, 6th edition, USA, John Wiley and Sons, Inc.
- Munir, M., Siddiqui, S.A., Chattha, M.A., Dengel, A., & Ahmed, S. (2019). FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models. *Sensors*, 19(11), 2451. Doi:10.3390/s19112451.
- Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. In *IEEE Access*, vol. 7, pp. 1991-2005, doi: 10.1109/ACCESS.2018.2886457.
- Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

- Neill, D.B., Wong, W.-K. (2009). "Tutorial on Event Detection," presented at the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, Jul. 2009.
- Nogas, J., Khan, S.S., Mihailidis, A. (2018). DeepFall: Non-Invasive Fall Detection with Deep Spatio-Temporal Convolutional Autoencoders. arXiv preprint; arXiv:1809.00977.
- Oh, D.Y. & Yun, I.D. (2018). Residual Error Based Anomaly Detection Using Auto-Encoder in SMD Machine Sound. *Sensors*, 18(5), 1308. Doi:[10.3390/s18051308](https://doi.org/10.3390/s18051308).
- Patraucean, V., Handa, A., & Cipolla, R. (2016). Spatio-temporal video autoencoder with differentiable memory. In ICLR Workshop. [Online]. Available: <http://arxiv.org/abs/1511.06309>
- Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR* 12, pp. 2825-2830.
- Peng, Z. K. & Chu, F. L. (2004). Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography. *Mechanical Systems and Signal Processing*, 18(2), pp. 199–221.
- Pepe, E., Bajardi, P., Guavin, L., Privitera, F., & Lake, B. (2020). COVID-19 outbreak response: a first assessment of mobility changes in Italy following national lockdown. Preprint from medRxiv; Doi: 10.1101/2020.03.22.20039933.
- Prastawa, M., Bullitt, E., Ho, S., & Gerig, G. (2004). A Brain Tumor Segmentation Framework based on Outlier Detection. *Medical Image Analysis*, 8, pp. 275–283.
- Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, Bellevue, Washington, USA, pp. 833–840.
- Rogan, J., Miller, J., Stow, D. A., Franklin, J., Levien, L., & Fischer, C. (2003) Land-Cover Change Monitoring with Classification Trees Using Landsat TM and Ancillary Data. *Photogrammetric Engineering and Remote Sensing*, 69(7), pp. 793–804.
- Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).
- Sakurada, M. & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. Doi:10.1145/2689746.2689747.

- Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In Proceedings 20th International Conference on Artificial Neural Networks, Part III, p. 92-101, Springer.
- Seabold, S. & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. Proceedings of the 9th Python in Science Conference.
- Shekhar, S., Lu, C. T., & Zhang, P. (2001). Detecting Graph-based Spatial Outliers: Algorithms and Applications. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 371–376.
- Shekhar, S., Lu, C. T., & Zhang, P. (2003). A Unified Approach to Detecting Spatial Outliers. *Geoinformatica*, 7(2), pp. 139–166.
- Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Proc. Adv. Neural Inf. Process. Syst.*, pp. 802-810.
- Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. Published in ICLR 2014, arXiv:1409.1556v6.
- Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised Learning of Video Representations using LSTMs. arXiv:1502.04681v3.
- Sun, P. & Chawla, S. (2004). On local spatial outliers. In Proceedings of 4th IEEE International Conference on Data Mining, p. 209-216.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*.
- Thottan, M. & Ji, C. (2003). Anomaly Detection in IP Networks. *IEEE Transactions on Signal Processing*, vol. 51(8), pp. 2191–2204.
- Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11, pp. 3371–3408.
- Wagner, M.M., Tsui, F.C., Espino, J.U., Dato, V.M., Sittig, D.F., Caruana, R.A., McGinnis, L.F., Deerfield, D.W., Druzdzal, M.J., & Fridsma, D.B. (2001). The Emerging Science of Very Early Detection of Disease Outbreaks. *J. Public Health Manag Pract.*, vol. 7, no. 6, pp. 51-9; Doi:10.1097/00124784-200107060-00006.
- Wang, Z., Yan, W., & Oates, T. (2016). Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578–1585.

- Wong, W., Moore, A., Cooper, G., & Wagner, M. (2002). Rule-based anomaly pattern detection for detecting disease outbreaks. In Proc. AAAI, Edmonton, Alberta, Canada, pp. 217-223.
- Wong, W., Moore, A., Cooper, G., & Wagner, M. (2005). What's strange about recent events (WSARE): An algorithm for the early detection of disease outbreaks. *Journal of Machine Learning Research*, vol. 6, pp.1961–1998.
- Wu, E., Liu, W., Chawla, S. (2010) Spatio-temporal Outlier Detection in Precipitation Data. *Knowledge Discovery from Sensor Data. Sensor-KDD 2008. Lecture Notes in Computer Science*, vol. 5840. Springer, Berlin, Heidelberg.
- Wu, Z., Wang, X., Jiang, Y.-G., Ye, H., & Xue, X. (2015). Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification. The 23rd ACM International Conference, arXiv:1504.01561v1.
- Yaacob, A. H., Tan, I. K., Chien, S. F. & Tan, H. K. (2010). ARIMA based network anomaly detection. In Proc. 2nd Int. Conf. Commun. Softw. Netw. (ICCSN), pp. 205-209.
- Yaminshi, K. & Takeuchi, J. (2002). A Unifying Framework for Detecting Outliers and Change Points from Time Series Non-Stationary Data. *ACM KDD Conference*.
- Yu, Q., Jibin, L., Jiang, L. (2016). An improved ARIMA-based traffic anomaly detection algorithm for wireless sensor networks. *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, p. 9653230.
- Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W. et al. (2019). A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. *The Thirty-Third AAAI Conference on Artificial Intelligence, Technical Track: Computational Sustainability*, vol. 33(01), p. 1409-1416.
- Zhao, J., Lu, C.-T., & Kou, Y. (2003) Detecting Region Outliers in Meteorological Data. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, p. 49–55.
- Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., Hua, X.S. (2017). Spatio-Temporal AutoEncoder for Video Anomaly Detection. In: *Proceedings of the ACM on multimedia conference, MM '17*. ACM, New York.
- Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014) Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. In: *Web-Age Information Management, WAIM 2014. Lecture Notes in Computer Science*, vol. 8485. Springer, Cham.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2016). Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Frontiers of Computer Science*, vol. 10(1): 96–112.



CURRICULUM VITAE

Personal Information

Name Surname : YILDIZ KARADAYI
Place and Date of Birth : OSMANIYE 18-08-1976

Education

Undergraduate Education : Control and Computer Engineering, Istanbul Technical University, Istanbul, Turkey
Graduate Education : M.A.Sc in Computer Science, Simon Fraser University, Burnaby, B.C, Canada
Foreign Language Skills : English (Advanced)

Work Experience

I have been working with private sector since I graduated from Istanbul Technical University in 1998. I have worked with many companies from different sectors such as Telecommunications and Finance. I have led various big data and machine learning projects to solve business problems such as fraud detection, anomaly detection, sales prediction, and predictive maintenance. My research interests include machine learning, deep learning, unsupervised learning, anomaly detection, prediction, multivariate and spatio-temporal data analysis.

Contact:

E-mail Address : yildiz.karadayi@gmail.com