A Sequence to Image Transformation Technique for Anomaly Detection in Drifting Data Streams

Detection, Segmentation and Generative Models on Multiple Objects

Sid Ryan

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Ph.D. degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In many real-world applications, the characteristics of data change over time. This behavior is known as concept drift. Maintaining optimal algorithms and their hyperparameters in such applications becomes cumbersome, as models become outdated very quickly. Although the data often consists of one-dimensional streams (e.g. collected by activity logs, sensors and mobile devices), in a higher level the aggregated sources produce multiple streams. Machine learning, therefore, requires univariate and multivariate analysis of long term dependencies to create valuable insights. In this thesis, we assess hundreds of combinations of data characteristics and methods in sequential data. Particularly we use real-life anomalous instances in the network traffic domain and to increase complexity we combine it with synthesized drifting data. From our preliminary evaluation of conventional machine learning, meta-learning and deep learning methods and comparing their generalization performance in the presence of concept drift, the results show that deep learning outperforms all other tested methods. Although, one-dimensional Convolutional Neural Networks (1D-CNN) produced the highest performance in image classification, similar to other models, they are able to label if sliding windows are anomalous or not. However, in majority of real-life applications, it is crucial to find individual instances that resulted in an anomalous pattern. Therefore, we introduce a method to transform the representation of the data to tensors of two dimensional images, enabling modern deep learning methods to become directly applicable to sequential data. We propose Sequential Mask Convolutional Neural Network (SMCNN) pinpoints the location of anomalous patterns. SMCNN model transforms sequential data by means of a specialized filter that produces flexible shape forms and detects multiple types of outliers simultaneously. In addition, to solve the issue of high ratio of False Positive in the unsupervised Generative Adversarial Networks (GAN) in concept drifts, we introduce a method for finding optimal sliding windows that automatically removes normal repetitive patterns. We introduce DriftGAN architecture that discriminates between normal and anomalous patterns. Our SMCNN and DriftGAN methods significantly outperform prior endeavours and provide high generalization capabilities on a wide array of one-dimensional data characteristics with repetitive nature.

List of Publications

Patents:

1. Utility Patent Application
   Title: Pattern detection in time-series data
   Inventor(s): Sid Ryan et al.
   Application No.: 62/683,889
   Filing Date: June 12, 2018

2. Utility Patent Application
   Title: Unsupervised outlier detection in time-series data
   Inventor(s): Sid Ryan et al.
   Application No.: 16/540,414
   Filing Date: August 13, 2019

Publications:

1. Ryan, Sid, et al., "Applied Data Science in Dynamic Systems", The 36th Annual Quality and Productivity Research Conference (QPRC), Washington DC, 2019. (accepted)

2. Ryan, Sid, et al., "Pattern and Anomaly Localization in Complex and Dynamic Data", 18th IEEE International Conference on Machine Learning and Applications (ICMLA), Special Issue on Machine and Deep Learning in Cyber Security and Privacy Issues, Florida, USA, 2019. (accepted)

3. Ryan, Sid, et al. "Deep Learning versus Conventional Learning in Data Streams with Concept Drifts", 18th IEEE International Conference on Machine Learning and Applications (ICMLA), Special Issue on Deep Learning, Florida, USA, 2019. (accepted)

4. Generative Adversarial Networks for Pattern and Anomaly Detection in Concept Drift. (Being prepared for submission)

Acknowledgments

First and foremost, I would like to express my profound gratitude to my advisors Dr. Nathalie Japkowicz and Dr. Iluju Kiringa, for their encouragements and great support throughout my PhD.

I had the greatest chance to learn from one of the most well-known professors in the field of machine learning, Dr. Nathalie Japkowicz. Her enthusiasm, inspiration, and great efforts to explain things clearly and simply laid the foundation of a thinking mechanism that helped me with transforming my raw ideas to applicable real-world outcomes. Another tremendous chance was the access to the wisdom and guidance of Dr. Iluju Kiringa. His insights were always generously offered without hesitation.

I had the opportunity to learn and work with a great number of researchers outside of the University of Ottawa. I would like to thank Dr. Roberto Corizzo, Dr. Todd Morris and Dr. Petar Djukic for their guidances and support. I am endlessly thankful for all that they have contributed.

Much of my research would undoubtedly have been impossible without the contributions of Miatcs and Ciena Corporation. I also thank Dr. Doina Precup at the McGill University, where I had the opportunity to attend as a Graduate Research Trainee in the Reasoning and Learning (RL) lab, and the Montreal Institute of Learning Algorithms (MILA).

I would like to thank the University of Ottawa, for the financial support that facilitated my doctoral studies. I would like to thank Dr. Tet Yeap and Dr. Karin Hinzer. I thank my friends and colleagues in the University of Ottawa IoT Lab for their help whenever I needed it. And in the end I would like to express my thanks to my lovely wife, Dr. Fari Ryan, who is the inspiration of my life and her support was one of absolute importance.

This research is partially supported by the Mitacs Accelerate PhD Fellowship and NSERC Engage Grant.

Contents

List of Tables

List of Figures

Glossary of Terms

NN Neural Networks

CNN Convolutional Neural Networks

NFL No-Free-Lunch (Theorem)

Autoencoder A Neural Network model whose goal is to predict the input somewhere in the network to learn a lower-dimensional representation of the input.

Activation Function Learns the complex decision boundaries by applying a nonlinear activation function to some of the layers (sigmoid, tanh, ReLU (Rectified Linear Unit) and variants of these).

Attention Mechanism Inspired by human visual attention to focus on specific parts of an image.

Average-Pooling A technique used in Convolutional Neural Networks by sliding a window over patches of features, such as pixels, and taking the average of all values within the window and compressing the input representation into a lower-dimensional representation.

Backpropagation An algorithm to calculate the gradients by applying the chain rule of differentiation starting from the network output and propagating the gradients backward.

Batch Normalization A technique that normalizes layer inputs per mini-batch to speed up training with higher learner rates that act as a regularizer.

Convolutional Neural Network (CNN, ConvNet) Uses convolutions to connected extract features from local regions of an input, by combinations of convolutional, pooling and affine layers.

Restricted Boltzmann Machine (RBN) A type of probabilistic graphical model that can be interpreted as a stochastic artificial neural network in an unsupervised manner. An RBN consists of visible and hidden layer, and connections between binary neurons in each of these layers.

Deep Belief Network A type of probabilistic graphical model that learn a hierarchical representation of the data in an unsupervised manner consisting multiple hidden layers

built by stacking multiple RBNs on top of each other and training them one by one.

Embedding An embedding maps an input representation into a vector. For example, mapping images and their textual descriptions into a common embedding space and minimizing the distance between them.

Exploding Gradient Problem The Problem is the opposite of the Vanishing Gradient Problem. In Deep Neural Networks gradients may explode during backpropagation, resulting number overflows.

Vanishing Gradient Problem The activation functions whose gradients tend to be small (in the range of 0 from 1) are multiplied during backpropagation, and tend to vanish throughout the layers, preventing the network from learning long-range dependencies. Common ways to counter this problem is to use activation functions like ReLUs that do not suffer from small gradients, or use architectures like LSTMs that explicitly combat vanishing gradients.

Fine-Tuning The technique of initializing a network with parameters from another task (such as an unsupervised training task), and then updating these parameters based on the task at hand. For example, NLP architecture often use pre-trained word embeddings like word2vec, and these word embeddings are then updated during training based for a specific task like Sentiment Analysis.

Keras A Python-based Deep Learning library that includes many high-level building blocks for deep Neural Networks. It can run on top of either TensorFlow, Theano, or CNTK.

LSTM Long Short-Term Memory networks were invented to prevent the vanishing gradient problem in Recurrent Neural Networks by using a memory gating mechanism. Using LSTM units to calculate the hidden state in an RNN we help to the network to efficiently propagate gradients and learn long-range dependencies.

Max-Pooling A pooling operations typically used in Convolutional Neural Networks. A max-pooling layer selects the maximum value from a patch of features. Just like a convolutional layer, pooling layers are parameterized by a window (patch) size and stride size. Pooling layers help to reduce the dimensionality of a representation by keeping only the most salient information, and in the case of image inputs, they provide basic invariance to translation (the same maximum values will be selected even if the image is shifted by a few pixels). Pooling layers are typically inserted between successive convolutional layers.

Momentum is an extension to the Gradient Descent Algorithm that accelerates or damps the parameter updates. In practice, including a momentum term in the gradient descent updates leads to better convergence rates in Deep Networks.

Multilayer Perceptron (MLP) A Multilayer Perceptron has multiple layers that use activation functions to deal with data which is not linearly separable. An MLP is the most

basic form of a multilayer Neural Network, or a deep Neural Networks if it has more than 2 layers.

Recursive Neural Network A generalization of Recurrent Neural Networks to a tree-like structure and are often applied to problem that already have a predefined structure, like a parse tree in Natural Language Processing.

Recurrent Neural Network (RNN) Models sequential interactions at each time step, based on the current input and the previous hidden state.

ReLU Rectified Linear Unit(s) are defined by f(x) = max(0, x) and are the most commonly used activation function in Convolutional Neural Networks (Leaky ReLUs, Parametric ReLU (PReLU) or a smoother softplus approximation).

Softmax This function is typically used to convert a vector of raw scores into class probabilities at the output layer of a Neural Network used for classification. It normalizes the scores by exponentiating and dividing by a normalization constant (Hierarchical Softmax or sampling-based loss such as NCE).

TensorFlow Is an open source C++/Python software library for numerical computation using data flow graphs, particularly Deep Neural Networks.

Chapter 1

Introduction

1.1 Motivation and Importance

Anomaly detection in evolving data is of broad scientific importance. It is vital to detect, localize and classify various types of anomalous patterns in their early stages, because they can warn of future failures. The accurate detection of anomalies also helps to protect assets by proactively changing the current path of progress to avoid future catastrophic events. In many real-world applications, such as in studies of cybersecurity [71], credit card fraud detection [31], in smart home applications [6], and medical informatics [105], to name a few, the data characteristics change over time. In addition, many connected systems worldwide produce data, often in the form of steams (e.g. phones, cars, airplanes, Internet of Things (IoT), and traffic sensors) with unique data characteristics while commonly show cyclic patterns. Detection of anomalous patterns among normal cyclic patterns can be hard for conventional machine learning models.

In many real-life applications the data shows repetitive nature as it correlates to cyclical human behaviour. An anomalous pattern, therefore, can be defined as an unusual deviation from the normal patterns, while the normal patterns can evolve over time. It is common that normal data characteristics vary among domains in the real-world, while the anomalous patterns remain relatively the same. Creating machine learning models with stable performance over time is essential for optimal operations. However, the majority of current machine learning models for anomaly detection are designed to perform well in a limited range of characteristics changes from the training-set.

Complex data in many real-life applications have two main properties:

- higher order cyclic behaviours resulted from interaction between components and the environment, in which an effective anomaly detection requires learning several one-dimensional data streams in the form of multivariate analysis, and
- dynamic patterns produced by variations in elements produce changes in data distribution over time, known as concept drift.

Real-world systems often exhibit nonlinear dynamics, translating to the possibility of sudden changes in the collected data and moving to a novel regime, which means that the future of data is fundamentally uncertain. Dynamic and complex data might contain stable repetitive patterns but can change gradually or suddenly. In addition over time

the data distribution can manifest in different forms. The concept drift may happen by observing a new pattern that was not seen before, or previously seen concepts may reoccur after a long period of absence. The drift can happen constantly and incrementally with intermediate concepts in between (e.g. a sensor slowly becomes less accurate). It can abruptly switch (e.g. replacing a sensor with different calibration). It also can gradually move from one regime to another (e.g. network traffic ratio changes from business to home usage on holidays. The traffic patterns might switch to the previous observation in the same period of time). Figure 2.1 illustrates various types of concept drift while X-axis is Time, and Y-axis is P(Y|X). One big challenge is to design an pattern detection algorithm that can differentiate between legitimate concept drift from outlier or noise which refers to a once-off random deviation or anomaly.

In dynamic data any new point can have a Singularity, and the learned models from previous trend and behaviour become invalid. Singularities are determined by their characteristics such as: Complexity (events in accordance to the environment), Interaction (arise when unexpected interactions occur between events), Relatedness (represent a peculiarity based on a domain), Uniqueness (stand out by qualitative features), Randomness (causes/effects are not well known), Irreversibility (changes are largely irreversible), Subjectivity (The awareness is dependent on the perception and experience), and Instability (small causes produce great effects). Further, interconnectivity and accumulation of individual data sources combined with concept drift can result in high levels of False Positive and False Negatives in anomaly detection. In such scenarios, a system close to a critical point can have no significant alarm until a total system failure. In this condition abnormal events happen more frequently than expected, based on the known normal distribution of previous events. Low-probability anomalies occur more often and a single trigger can erupt suddenly and forcefully, leading to the total failure of large parts of the system. Efficient and optimal intelligent mechanisms are therefore necessary to detect and isolate patterns and anomalies at all levels of a system. The monitoring, alerting and protection methods are essential for forecasting the future states and detecting anomalous activities to maintain the quality of services and protect assets. Hence, the majority of machine learning algorithms are not applicable to complex and dynamic data. To the best of our knowledge, no method currently exists that can be generalized to all types of novel data characteristics in great detail without the need for explicit re-configurations of models.

1.2 Problem

The majority of conventional machine learning algorithms are not applicable to multi-sensor anomaly detection in concept drift. These models require explicit and extensive configuration adjustments, which often quickly become outdated. Although, deep learning methods provide automation of optimization of parameters, they are often limited to the image, text and speech domains. A big challenge, resulting from the significant concept drift, is how to achieve optimal rates of detection performance with higher localization precision than regular section labeling. As it is impractical to assign high quality labels to the training data for many real-world problems, we investigate the possibility of achieving relatively high performance, similar to that of detection and segmentation tasks, using

generative models that minimize the requirements for labels.

1.3 Limitations of Current Solutions

The characteristics of complex environments result in concept drift in data. The majority of mainstream machine learning solutions perform poorly with sequential data and even worse with multidimensional streams with concept drift. Deep learning methods provide abstraction layers that overcome the limitations of conventional models. The wide knowledge acquired in the context of deep learning cannot be directly applied to sequential data. Methods such as convolutional neural network (CNN) and GAN models are mainly developed for image domain, and the limitation in generalization to non-image-based data is a drawback for other dimensions of data. In addition, deep learning requires huge amounts of labeled data for training, which in many real-world applications is not possible. We also show that GANs have limitations in differentiating between anomalies and legitimate concept drift and propose a method to overcome these limitations.

1.4 Proposed Solution

In order to achieve a model that generalizes to new data characteristics, we investigate the performance of deep learning models (e.g., LSTM, CNN) and compare it with an upper-bound of combining conventional machine learning models in a meta-learning set up. We expect deep learning to have better performance in detecting patterns and anomalies in data and generalize to drifted characteristics. To extend the framework we introduce a segmentation method in which in the first step transforms the streaming data to a unified tensor of two-dimensional (2D) representation and then using Sequential Masking CNNs we find location and boundaries of various types of patterns and anomalies in sequences and in parallel. In the final part of this thesis we stretch the solution to unlabeled data, we introduce an automatic sliding window finder and an unsupervised method in which by benefiting from GAN architecture overcomes the limitations of supervised models in real-world applications.

1.5 Method

In order to achieve high degrees of accuracy we develop various methods for transforming sequential data to image domain and to automatically preprocess the data enabling advanced deep learning become directly applicable to sequential data. Advantages of that would be:

- harnessing the power of Deep Learning for one-dimensional data,
- dealing with changes in data characteristics with no reconfiguration requirement,
- localization of multiple patterns and anomalies in large slices of time, and
- detecting anomalies and patterns in data without labels.

We propose detection, segmentation and generative frameworks that extend the precision of current state-of-the-art anomaly detection. We evaluate numerous combinations

of tests by changing the characteristics of the data, step by step. We also evaluate various preprocessing configurations and machine learning models to derive reliable results for concluding the generalization performance of methods. Further, beyond the limitations of classification, we explicitly determine the boundaries of patterns and anomalies by introducing an approach towards segmenting streaming data with minimum labels. We also investigate the feasibility of transforming the problem to be executable in unsupervised algorithms. We introduce three approaches toward data science in sequential and streams data by studying the following approaches:

- abstract the raw data to a 2D format with visualized representation,
- generate images by finding optimal scanning window size parameters that match the patterns of the data,
- remove short-term and long-term normal repetitive normal pattern in the data,
- compare the performance of deep learning models, and investigate the improvement over conventional machine learning models in concept drift,
- explicitly localize the boundaries of multiple patterns and anomalies in images,
- generalize the generative adversarial network capacities to detect novel pattern and anomalies while normal data characteristics change.

1.6 Contribution

Our main contributions consist in enabling computer vision algorithms to be applicable to one dimensional non-image data and extending detection, segmentation and generative models to find patterns and anomalies in cyclic data with concept drift. The following lists the contributions of this thesis in the order they appear in the thesis.

- **A feature extraction approach for unifying multidimensional stream data.** We propose a guideline for transforming various data into one unified abstracted representation that facilitates learning complex and evolving data. Our novel method works via visualizing the data to a sequence of 2D tensor representation. The approach generalizes to various types of data via pixel-wise learning and made the segmentation for detecting and localizing patterns and anomalies possible in various domains with no significant re-configuration requirements.

- **Deep learning architectures.** Autoregressive-based models (e.g. ARIMA for trend and SARIMA for seasonal component) are unable to learn changes in characteristics over time. Deep learning methods deal with variations in data and optimally detect targets, but they are often designed for image processing problems. We compare the performance between deep learning-based models and conventional machine learning models, including long short-term memory (LSTM), 1D CNNs, and 2D CNNs, on sequential data with concept drifts and showed the performance of each model.

- **An in-depth comparison of conventional machine learning algorithms.** We compare various data characteristics anomaly detection performances using seven mainstream and commonly used machine learning algorithms, altering one aspect of the data each time. We examine if there is a necessity of using a meta-learning

approach or if there are models that consistently offer higher performances. We examine the effectiveness of the deep learning methods using similar data on sequential data with concept drifts.

- **Sequential masking CNNs (SMCNN).** In a supervised context we modify a state-of-the-art method from image segmentation domain to sequential data. We detect and localize the exact boundaries of multiple targets simultaneously with minimum labels. Our feature masking technique can be applied to temporal data in various dimension data generated. We apply the framework to heat-map and Fourier transformed representations of sequential data with concept drifts and compare the performance.

- **An unsupervised analytical framework to learn drifting paradigms.** In an unsupervised context our contribution suggests a GAN-based framework for unsupervised learning of sequential data with concept drifts. Our proposed DriftGAN is a framework for detecting and localizing patterns and anomalies of sequential data without assigning labels. In addition, DriftGAN enables deep learning models to be applicable in sequential data with drifting paradigms.

The remainder of this chapter provides further context for the problem and outlines our approach. We first describe our synthetically generated concept drift combined with real-world anomalous patterns. Then we evaluate the generic conventional algorithm and meta-learning approaches, and deep learning technologies that can be used for the mentioned problems. We introduce an approach towards temporal data studies and adopted a version of a state-of-the-art feature masking [30], [59] method for object segmentation to detect and localize patterns and anomalies in the data. Finally, we propose an unsupervised architecture based on Generative Adversarial Networks (GAN) [52], [129].

1.7 Thesis Outline

Chapter 1 provided an introductory overview of the various paradigms of learning complex and dynamic data and describes our experimental setup for generating data. The remainder of this thesis is organized as follows.

Chapter 2 surveys the literature and provides an overview of anomaly detection method in the presence of concept drift, particularly meta-learning and deep learning methods. Chapter 3 describes the methodology used for the experiment and in the second half of the chapter compares the performance of several conventional machine learning with deep learning models for pattern and anomaly detection and introduces our innovative methods for the transformation of raw data to higher level abstraction and illustrates the architectures we designed to overcome the challenges in working with concept drift both for supervised and unsupervised learning and our technique for automatically finding the optimal rolling window size and configuration. In addition, Chapter 3 presents details of a real-world implementation of the system in networking applications. Chapter 4 covers results obtained from various data characteristics and changes in concept drift intensity while testing different types of preprocessing methods and machine learning models. Chapter

5 introduces our novel segmentation method that can pinpoint the location of anomalies within the time slice. Chapter 6 proposes a novel approach to automatically remove concept drift and works towards segmentation of multiple targets in sequential data in an unsupervised way using Generative Adversarial Networks. Chapter 6 provides the discussion and Chapter 8 summarizes the conclusions of this thesis.

NOTE: The Chapter 5 and 6 of this thesis are manuscript-based chapters and organized to be self-contained and include materials that might seem slightly redundant but enable the reader to understand the chapters without necessarily requiring to read and have the knowledge of other chapters of the document.

Chapter 2

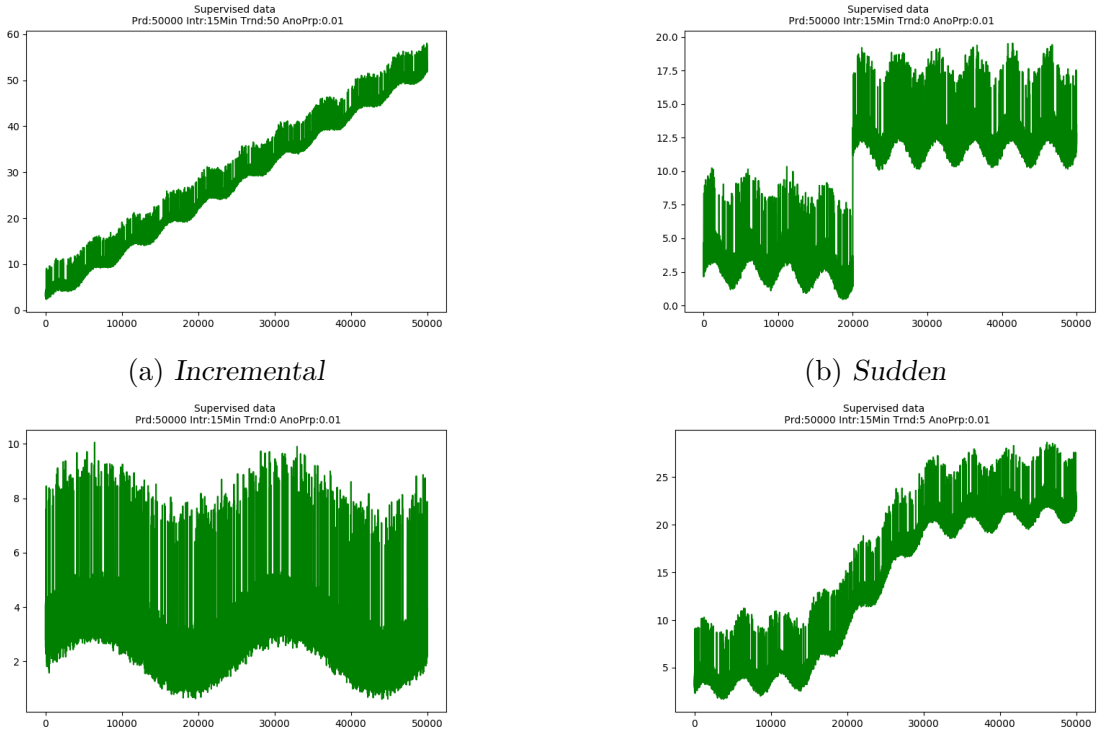
Previous Work

In this chapter, we review important works in the fields of supervised and unsupervised pattern and anomaly detection, segmentation and generative-discriminative models. We review a very few previous attempts in transforming one dimensional data to image in order to deal with limitations of current methods. We also describe the specifics of our problem space and deficiencies of previous research.

2.1 Learning in Concept Drift

Concept drift refers to the change of relation between input data and target variables over time. The core assumption, when dealing with the concept drift problem, is uncertainty about the future. We assume that the source of the target instance $X_t + 1$ is not known with certainty. The future can be assumed, estimated or predicted but there is no certainty. The concept drift may happen by observing a new pattern that was not seen before, or previously seen concepts may reoccur after a long period of absence. The drift also can happen constantly and incrementally with intermediate concepts in between (e.g. a sensor slowly becomes less accurate). In other scenarios, a concept can abruptly switch (e.g. replacing a sensor with different calibration). Also, the concept drift can gradually move from one regime to another (e.g. network traffic ratio change from business to home usage on holidays). A big challenge is to design a detection algorithm that can differentiate between legitimate concept drift from noise and outlier, which refers to a once-off random deviation or anomaly [136].

Concept drifts can be categorized to two main groups, based on where the distribution changes happen. A *Real concept drift* refers to changes in the conditional distribution of the output, while the distribution of the input remains unchanged. *Virtual drift* happens if the distribution of the incoming data changes (i.e., P(X) changes) without affecting P(Y|X) [118]. This, usually, occurs due to incomplete data representation rather than changes in concepts in reality [43].



(a) *Incremental* (b) *Sudden*

(c) *Recurrent* (d) *Gradual*

Figure 2.1: Overtime the data distribution can manifest in different forms. The concept drift may happen by observing a new pattern that was not seen before, or previously seen concepts may reoccur after a long period of absence. The drift can happen constantly and incrementally (a) with intermediate concepts in between (e.g. a sensor slowly becomes less accurate). The concept can abruptly (b) switch (e.g. replacing a sensor with different calibration). The concept drift can gradually (d) move from one regime to another (e.g. network traffic ratio changes from business to home usage on holidays), and recurrently (c) such as in the network traffic in which patterns might switch to the previous observation in the same period of time. One big challenge is to design a detection algorithm that can differentiate between legitimate concept drift from outlier or noise which refers to a once-off random deviation or anomaly. Note that X-axis is Time, and Y-axis is P(Y|X).

Single example such in Online learning algorithms can be seen as naturally adaptive to evolving distributions, mainly due a mechanism that continuously updates the model with the most recent examples. However, online learning systems do not have explicit forgetting mechanisms. *Multiple examples* often use memory window (which we use in this thesis). At each time step, the learning algorithm builds a new model using the examples from the training window. The model updates based on the new data and a forgets by discarding data that is moving out of the window. Sliding windows of a fixed size store in memory a fixed number of the most recent examples, while in sliding windows of variable size the number of examples in a window varies over time, typically depending on the indications of a change detector. However, the key challenge is how to select an appropriate window size. A short window reflects the current distribution more accurately, thus it can assure

fast adaptation in periods with concept changes, but in stable periods too short window worsens the performance of the system. A large window gives a better performance in stable periods, but it reacts to concept changes slower. The assumption behind relying on windowing is that the recency of the data is associated with relevance and importance. Unfortunately, this assumption may not be true in every circumstance. For instance, when the data is noisy or concepts reoccur, recency of data does not mean relevance. Moreover, if slow change patterns lasts longer than the window size, windowing may fail as well [43].

In dynamic environments [29] it is unreasonable to expect a well performing optimization algorithm to work in in all situations. The performance of algorithms depends on how close the assumptions and biases of algorithms are to the features and characteristics of the data [13]. Generally, induction algorithms are either explicitly or implicitly biased results in unequal performance in various domains [85]. Generalizing performance of algorithms to all types of novel data is essential for many real-world applications to coup with system changes. However, as defined in the Conservation law [98], a positive performance must be balanced by a negative performance in other situations. Moreover, the NFL theorem [124] implies that no learner can beat random guessing over all domain functions (e.g. flipping a coin). If one inducer is better than another in some domains, then there are other domains in which this relationship is reversed.

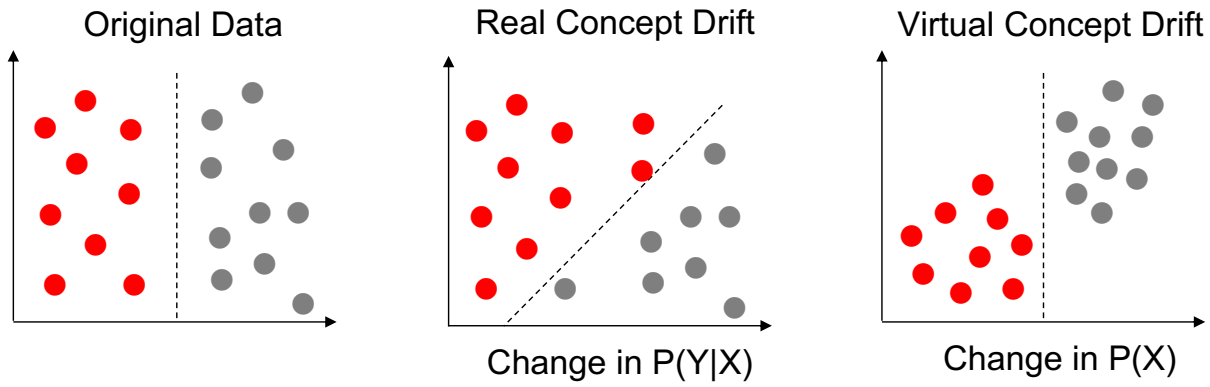


Figure 2.2: There are two main categories of concept drift, real concept drift and virtual concept drift. Real concept drift happens when the P(Y|X) changes, while virtual concept drift is when only P(X) changes and the P(Y|X) remains the same [43].

The *change detection* refers to the mechanisms for explicit drift detection using characterizing concept drift by indicating change-points [7]. The advantage of explicit change detection is the information about the dynamics of the process generating data [43], while it will require more precisely labeled data. Further, online learning systems can adapt to evolving data without any explicit change detection mechanism. Furthermore, the changes in data distribution over time may manifest in different forms. In one-dimensional data the data mean changes may happen suddenly by switching from one concept to another, incrementally consisting of many intermediate concepts in between, suddenly or gradually.

In machine Intelligence in dynamic data, the *Learning* refers to the mechanisms for generalizing from examples and updating models from evolving data that can have two modes of Retraining and Incremental adaptation. The *Retraining* method discards the

current model and builds a new model from scratch using buffered data. At the beginning a model is trained with all the available data. Next, whenever new data arrives, in equal intervals or based on the new data characteristics, the previous model is discarded, the new data is merged with the previous data, and a new model is learned on this data. The *Incremental* adaptation that updates the current base model using the most recent data. Incremental algorithms process input examples one-by-one and update the model after receiving each example and might have access to previous examples or summaries of examples. Further, the Online learning mode updates the current model with the most recent example. They are error-driven updating the current model depending on whether it misclassified the current example. Streaming algorithms are online algorithms for processing high-speed continuous flow of data sequentially and can be examined in typically one pass using limited memory. Furthermore, the *Adaptation* methods fall into two major types of blind and informed strategies. The *blind adaptation* updates models without any explicit detection of changes in the data. It typically uses a fixed size sliding window to periodically retrain models with the latest samples (i.e., assigning a higher weight to recent samples). The *informed* strategy, however, is reactive and depends on whether a change of data characteristic is detected [43].

Adaptive Learning algorithms are incremental learning methods that are able to adapt to evolution of the data generating process over time [43]. While *Incremental algorithms* process input examples one-by-one (or batch-by-batch) and update the decision model after receiving each example while may have random access to previous representative examples [43]. Most adaptive learning techniques implicitly or explicitly assume and specialize in some subset of concept drifts. Many of them assume sudden non-recurring drifts. But in reality often mixtures of many types can be observed [43]. Further, *Abrupt drift*, or sudden drift occurs when a stream with concept a suddenly changes to concept such as in a market crash [116], or impacts of a virus pandemic in a stock market stream, almost instantly, stock values will change and follow a pattern different to previously. In contrast, a real world example of *extended drift* could be a recession. Unlike a market crash, stock values at the beginning of a recession will slowly change over an extended period of time. Eventually, the changes in all of the stock values will follow a different pattern to before the recession started [116].

2.2 Learning Sequential and Streaming Data

In the context of data with hidden patterns (e.g. time-series, temporal measurements, IoT and streaming data), variables are endogenous and dependent upon one another. The observations are only meaningful when they are analyzed together because the instance a depends upon the observation at $a-1$, $a-2$, ..., $a-m$ and likely influences the $a+1$, $a+2$, ..., $a+n$ instance. Further, these types of data often exhibit repetitive discernible time-dependent structures correlating with periodic behaviour and dynamics. The characteristics of sequential data can be classified into three main categories: the trend or long-term direction of data growth, the seasonal or systematic periodic movements, the concept drift, and the irregular unsystematic short-term fluctuations. More explicitly in this context, the temporal data exhibit cyclic and seasonal patterns often with a trend component

that frequently combines with noise. Further, anomaly detection can be applied to single or multiple sources of data in univariate or multivariate analysis tasks. *Univariate* is a sequence of data points, measured at successive uniform interval points in time $T = t1, t2, ..., tn$, and n is the length of T. While *Multivariate* is a set of sequences with the same timestamps. In a multivariate M, each element m_i is a univariate that at any timestamp t, $m_t = m1t, m2t, ..., mlt$, where l is the number of univariate sequences.

Traditional data mining considers an offline approach such as using historical data to train models for predicting the output for new unseen input data. However, streaming data can not be processed similarly because data comes continuously over time and theoretically is never ending. To accommodate such data in the machine's main memory an online processing method can be used to either train models incrementally by continuous update or by retraining using the most recent batches of data. In *offline* learning the whole training data must be available at the training phase. While in contrast, *online* algorithms process data sequentially. The model is continuously updated during operation as more data arrives [43]. In addition, learning under concept drift is based on single or multiple examples. Further, *Streaming algorithms* are online algorithms for processing high-speed continuous flows of data, using limited memory and processing time for each instance [43]. A time-series is a data set in which the data elements have time stamps. In a standard application, a system only has access to data with time stamps prior to a specific point of time, but the models to be developed must be applied to data elements with subsequent time stamps [116]. In this thesis the generated data is in the form of sequences with long-term patterns, none of our assumptions exclude the generic one-dimensional data characteristics. The only limitation we dictate is keeping the order of inputs because we do not shuffle the instances. In this sense, we might consider that the conclusions are generic and apply to all types of data.

The majority of classical machine learning methods that have been designed for sequential data are not applicable when trends (long term direction), seasonal (systematic periodic movements) and irregular (unsystematic short-term fluctuations) components exist. Further, real-world complex systems exhibit non-linear dynamics, meaning the possibility of sudden changes in the data or moving to another regime, as in concept drift, is higher than observing a single source of data, which means that the future of data is uncertain. For the reasons mentioned, generalizing models obtained from the past data to the unseen future data is often invalid. Furthermore, concept drift is often handled by time windows or weighted examples according to their age or utility [42]. However, anomalies are by nature unexpected, so efficient detection systems should be able to determine anomalous events without relying on hard-coded thresholds. For efficient anomaly detection, continuous learning and detecting anomalies in the early stages are the two important factors to quickly detect subtle changes in patterns that are not apparent. While the SARIMA and HWES methods can handle data with trends and seasonality, they have limited generalization capacities, and they only perform well when the test-sets contain characteristics similar to the training set. Therefore, in the case of a concept drift, their performance is limited. Recently, concept drift has been handled by time windows or examples weighted according to their age or utility [42]. However, these methods potentially remove important aspects of the data and might delete abrupt and scarce anomalies.

When measuring streaming data produced by a dynamic systems, we can only observe

a limited portion of an infinitely large sequence. The observable section of all possible states of the data is often not enough to drive conclusions that are applicable to other parts of a system or changes over time. Methods that scramble the data and randomly select instances are not applicable in data with long-term patterns, and thus generalization to unseen data requires different approaches.

2.3 Anomaly Detection in Sequential Data with Concept Drifting

Anomaly detection refers to finding observations that do not conform to the expected behaviour (normal) of a system. These exceptional observations defined as anomalies, outliers, novelty and events in different application domains. Anomaly detection is an important problem in various areas from industrial damage, medical and public health and sensor networks domains to detecting anomalous regions in images and texts, as well as to intrusion or fraud detection (e.g., public security and military surveillance, system breakdowns, malicious activity, credit-card fraud, intrusions). Detecting anomalies in data often translate to significant critical actionable alerts that might avoid high costs. An anomalous network traffic pattern could identify a hacked computer that is sending out sensitive or private data to an unauthorized destination [69], anomalous readings from a space craft sensor could signify a fault in some component to avoid an explosion [41], a malignant tumour can be detected from anomalous MRI images [104] or in bank transactions anomalies could indicate credit card or identity theft [2], however, false alarms might result in high costs (e.g., shutting-down a data-centre, performing unnecessary surgeries, or cancelling flights). Anomaly detection is often tackled as one class classification task, where the properties of a normal behavior are well defined, while the properties of abnormal behovior may be changing. While in the data a concept drift can happen due to changes in behavior, characteristics of legitimate users, or new creative adversary actions. Anomaly detection is very relevant for computer security domain, in particular, network intrusion detection, In telecommunications fraud prevention [137].

A straightforward anomaly detector can find normal behavior and declare any observation that does not belong to this normal region as an anomaly,but there are several factors that make it challenging. Defining a normal region which encompasses every possible normal behavior is cumbersome, since the normal defined by history that might not cover all possible scenarios. In addition, the boundary between normal and anomalous behavior often cannot be defined precisely. For instance, the malicious adversaries often adapt their patterns to transform from obvious anomalous instances to normal appearing patterns. Further, in many domains the normal behavior keeps evolving over time or change its representation in various conditions. Therefore, the historical notion of normal behavior becomes insufficient for maintaining performance in the future. Furthermore, in applications such as medical domain a small deviation from normal might be an anomaly (e.g., body temperature), while similar deviation in values in the stock market domain can be considered normal. Thus, applying a technique developed in one application to another is not straightforward and often impossible. Main categories of pattern and anomaly

detection problems [107] break down as follows [21]:

- Outlier detection: Finding the most different sequence, given a corpus of unlabeled data, while the targets exist in the training dataset.
- Novelty detection: Finding the most different sequence, given a corpus of unlabeled data, while the targets do not exist in the training dataset.
- Sequence-based: Entire sequences are compared to one another (e.g. comparing aircraft landing control sequences to each other to identify pilot errors [16]).
- Subsequence-based: Test for anomalous subsequences within a longer sequence.
- Pattern frequency: Identify n-grams occurring with unusual frequencies in single or multiple sequences.
- Online: Sequences are tested one symbol at a time to identify anomalies as soon as they are apparent. This method is applicable to semi-supervised and unsupervised cases.
- Feature-based: Instead of evaluating sequences or subsequences directly, a different approach is to derive features from the sequence (i.e. symbol rates or other statistical measure of the sequence).
- Discord detection: Given a sequence, finding the subsequences most different from the rest of the sequence [67].

Supervised Anomaly Detection: A setup where the data is labeled in training and test data sets, and classification methods can be applied. This case is like traditional pattern recognition with the exception of classes which are in most cases strongly unbalanced. Not all classification approaches suit for this task. For example, some types of decision trees cannot deal well with unbalanced data. Although in many cases anomalies are not known in advance or may occur as novelties during the test phase. There are two main types of labeling in supervised anomaly detection; a scoring or a labeling method. The difference between scoring and labeling is in its flexibility. Using scoring techniques analyst can choose values which are more suitable for the problem area and use a threshold value to select anomalies or just choose the top ones. Labeling is a classification approach. Often none of the two approaches can be used in a new domains with the same success without knowledge about the domain and feature extraction. The formulation is induced by various factors such as nature of the data, availability of labeled data, type of anomalies to be detected, etc. Often, these factors are determined by the application domain in which the anomalies need to be detected.

- Scores; assigns an anomaly score to each instance in the testset depending on the degree to which that instance is considered anomalous as a ranked list. The analyst can choose top few anomalies or use a cut-off threshold.
- Labels; assigns a label to each test instance and the analyst to uses a domain-specific threshold to select the most relevant anomalies.

One-class classification: This setup also uses training and test datasets, where only training data consists of normal data without any anomalies. The idea is, that a model of the normal class is already taught, and anomalies can be detected by their deviation from learned model. In the beginning, when we do not have any knowledge, we gather

it from training results. Well-known approaches are One-class SVMs and Autoencoders [83], [65], [95]. In general, any density estimation approach can be applied to model the probability density function of the normal classes, such as Gaussian Mixture approaches or Kernel Density Estimation. Novelty detector identifies test data that does not fit the normal distribution of training data, which can be complex and high-dimensional.

Unsupervised Anomaly Detection: A setup, when we do not know, what is normal in the data and what is not. It is the most flexible configuration which does not require any labels. There is also no difference between a training and a test dataset. The concept is that an unsupervised anomaly detection approaches score the data solely based on natural features of the dataset. Typically, distances or densities are used to give an evaluation what is normal and what is an outlier.

Anomaly detection has been extensively studied and surveyed [20], and several techniques have been developed [116], [99]. The most popular approaches include clustering approaches, nearest neighbor methods and one-class classifications [135], [22]. More recent works use deep neural networks (DNNs), which do not require explicit feature construction for anomaly detection such as energy-based models [130], autoencoders [3], and deep autoencoding [138]. Carrera et al. [17] utilized convolutional sparse models to learn a dictionary to detect anomalous regions in texture images. However, these approaches mainly remain in the realm of images obtained from a camera, or they often perform poor when the characteristics of the data drift. In generally, anomalies can be classified into three categories:

- Point Anomalies: when one object can be observed against other objects as anomaly, and
- Collective Anomalies: in which a collection of linked objects (not individual object) might be anomalous, and
- Contextual Anomalies: when the object is anomalous in some defined context (conditional anomaly).

However, the above anomalies can be connected in many ways. For instance, point anomalies can become contextual and collective ones, if in a context, multiple point anomalies are joined together. The complexity of patterns and anomalies will make the task of accurate detection very challenging. Some of the challenges to mention include,

- Finding representative or real-world data with normal drifts and labeled anomalies for training and validation of model is a significant problem,
- defining normal regions is a very hard task especially in many cases that boundaries between anomalies and normal data are not precise. In this case, normal observations could be considered as anomalies and vice-versa,
- in actions that are malicious, such as in frauds, very often attackers adapt their actions to the normal behaviour, which makes the task of identifying patterns and even harder,
- what is considered normal today can be not normal in the future, or the opposite. Also many of the systems and businesses change in time under the influence of the various factors that makes trained models and their relative data outdated. This notion can be described as concept drift in the data both for normal patterns and

anomalous instances, and

- more often approaches for anomaly detection in one field cannot be generalized automatically in the other domains or applications and will be ineffective in most cases.

In this thesis we do not consider periodic seasonality as a concept drift problem, however, if seasonality is not known with certainty, we consider it as a concept drift problem. For instance, a peak in home usage network traffic is associated with holidays at the same time can be affected by a popular video or vacations that can start at different times every year. In the literature concept drift defines as change in the mean of the data [43]. However characteristic changes might not reflect on the mean of the data, such as changes in the magnitude of the repetitive patterns and seasonality amplitude. In this thesis we are considering an offline mode for the training phase, while the test setting is designed with an image transformation method in which the test-data arrives online and in real time, forming a stream which is potentially infinite and is transformed to images. Also, we focus on handling the real concept drift which is not visible from the input data distribution. In many cases the techniques that deal with real concept drift often perform well in virtual concept drift, but not vice versa [43]. In the first scenario, the $P(Y|X)$ changes from the training-set to the test-set without any change in the $P(X)$. In the second scenario, the data characteristics affects both $P(X)$ and $P(Y|X)$. It is common that normal data characteristics vary among domains in real-world, while the anomalous patterns remain relatively the same. Our model provides gradual forgetting of the past by increasing the number of data points in older images which results in slowly fading the older patterns.

2.3.1 Anomaly detection in network security domain

Network Intrusion Detection Systems deal with detecting intrusions in network data. The intrusions typically occur as anomalous patterns (point anomalies) in a sequential fashion and anomalous subsequences (collective anomalies) [56]. The primary reason for these anomalies is due to the attacks launched by outside hackers who want to gain unauthorized access to the network for information theft or to disrupt the network. A typical target setting is a large network of computers which is connected to the rest of the world via the Internet. The data available for intrusion detection systems can be at different levels of granularity. The data often has a temporal index associated with it but most of the techniques typically do not handle the sequential aspect explicitly. Another type of data is high dimensional typically with a mix of categorical as well as continuous attributes. A challenge faced by anomaly detection techniques in this domain is that the nature of anomalies keeps changing over time as the intruders adapt their network attacks to evade the existing intrusion detection solutions [20].

Machine learning methods have been used for a long time to tackle pattern analysis and anomaly detection tasks [99]. However, the majority of classical machine learning methods that have been designed for sequential data are not applicable when trend (i.e., long-term direction, seasonal or systematic periodic movements) and irregular or unsystematic short-term fluctuations components exist. For instance, the Seasonal Auto-Regressive Integrated Moving Average (SARIMA) [121] and the Holt Winter's Exponential Smoothing (HWES) [63] methods can handle data with trend and seasonality, but they have very limited generalization capabilities, i.e., they only perform well when the test set has characteristics

similar to the training set. Specifically, in time series forecasting applications, their performance starts to decrease as soon as the data distribution changes over time, due to concept drift [19]. In the literature, one way used to handle this phenomenon is represented by the adoption of time windows or examples weighted according to their age or utility [42]. These methods, however, potentially remove important aspects of the data and discard the consideration of abrupt and rare anomalies. Traditionally, dominant methods for sequential data types are linear models such as ARIMA and SARIMA. These models are easy to implement and explainable, but they suffer from severe limitations, such as an inability to generate nonlinear relationship functions and a lack of data imputation support. Another big issue with these models is their focus on fixed temporal dependences, which result in their impossibility to handle discontiguous data with nonuniform time stamps and fixed lagged observations. Other limitations are that they concentrate on univariate data, which is inadequate for the majority of real-world problems with multiple input variables, and one-step forecasts, which result in a short time horizon. To simplify the modelling process, classical methods identify and remove obvious structures systematically. However, this process might remove long-term patterns and anomalies reflecting seasonality or trends. In addition existing conventional machine learning techniques often depend on handcrafted features that are expensive to create and require injection of human-expert knowledge of the field.

Comparing the performance of anomaly detection algorithms is not as straight forward as in the classical cases due to highly imbalanced data. In contrast to simply comparing an accuracy value or precision-recall, the number of the anomalies should be taken into account. In classification, a wrongly classified instance is for sure a mistake. In anomaly detection however, a large dataset that contains few anomalies that are ranked among the top-n outliers. Therefore, detecting a few is still considered good result, even if it is not perfect. To this end, a common evaluation strategy for anomaly detection algorithms is to rank the results according to the anomaly score and then iteratively apply a threshold from the first to the last rank. This results in N tuple values (true positive rate and false positive rate), which form a single receiver operator characteristic (ROC). The area under the curve (AUC), the integral of the ROC, can be used as a detection performance measure. The AUC is the probability that an anomaly detection algorithm will assign a randomly chosen normal instance a lower score than a randomly chosen anomalous instance. Nevertheless, AUC based evaluation has been evolved to be the de facto standard in anomaly detection, most likely due to its practical interpretability [48], and thus also serves as the measure of choice in our evaluation. We use a mix of synthetic and real-world data (similar to Numenta Anomaly Benchmark (NAB) [74]). However, this thesis focuses on robust models that are able to withstand increases in concept drift intensity over time.

2.4 Supervised Learning in Dynamic Data

An environment can be considered dynamic if the behaviour of observable objects in a defined environment changes in time. A dynamic environment (communication network, production process, highway traffic, etc.) may contain huge amounts of information that can change with time. Understanding the general behaviour of the environment is necessary

in order to discover the regularities and anomalies for controlling, managing and intelligent modelling or managing of the environment. For instance, traffic and transaction volume change can be anomalous if suddenly shifted, but it also can be a normal increase. In another example the classification of customer behaviours can shift because of the change in their culture, perspectives, age or other factors in time or suddenly by a new user or a hacker. A systems can evolve with hardware upgrade, software updates or as user behaviour is changed [20], [103], [66], [48] and [74]. Majority of complex and dynamic data collected from various sensor and streaming data forming sequences of data with time dependency that often contain repetitive patterns.

Given the labels of in training dataset, a supervised deep learning model can find the patterns of how, for example, the pixel values translate to what is inside the image. For example, we can use a CNN for pattern recognition from the video data. Without the labels, however, we can often find patterns and structure within the data without recognizing whether there is a certain type of objects in the image. The existence of labels in training often results in higher accuracy of supervised methods as the models are able to evaluate whether the prediction or classification is correct or not.

2.4.1 Supervised conventional models

Box-Jenkins models, reviewed in [10], are perhaps the most commonly used method for temporal data. Autoregression (AR) methods observe current steps and produce only linear functions that predict the next step in the sequence. Similar linear function limitations also apply to moving average (MA) models which use the residual errors of the mean process at prior time steps. Autoregressive-MA (ARMA) [117] methods, which combine AR and MA, produce a linear function of the observations and residual errors of prior step. The Vector Autoregression (VAR) model creates generalization of AR to multiple parallel data for multivariate problems without trends and seasonal components. However, Autoregressive Integrated MA (ARIMA) methods produce a linear function of the difference of observations and residual errors at prior steps to make the sequence stationary, which makes the models applicable to data with trends, but they are still incapable of dealing with seasonality. The Vector Autoregression Moving-Average (VARMA) generalizes the ARMA models to multivariate data but without trend and seasonal components. Similar limitations apply to VARMA with Exogenous Regressors (VARMAX) and Simple Exponential Smoothing (SES) models. To tackle these limitations, the Seasonal ARIMA (SARIMA) models generate a linear function of the observations, seasonal observations and their errors. The SARIMA with Exogenous Regressors (SARIMAX) can handle univariate data with trends and seasonal components with exogenous variables as covariates of parallel input sequences with similar steps to the original endogenous data series. Additionally, the Holt Winter's Exponential Smoothing (HWES) uses an exponentially weighted linear function of observations of previous steps to deal with trends and seasonal components. However, the limitations of these methods are their need for training with explicit data and, in the case of concept drift, the performance becomes very limited over time [57], [18], [88].

Another research venue in supervised anomaly detection can be considered as performing generative modeling. These approaches attempt to build a model over the normal data

and check how well the new data fits to the model. An approach for modeling normal sequences using look ahead pairs and contiguous sequences is presented in [62]. A statistical method for ranking each sequence by comparing how often the sequence is known to occur in normal traces and how often it is expected to occur in intrusions is presented in [60]. One approach uses a prediction model obtained by using neural networks to obtain the model [45] to detect novel attacks.

2.4.2 Meta-learning

For tackling anomaly detection in concept drift, one choice is to select between various models based on their previous score in same data characteristics. It is well known that every individual category of learning algorithm has a unique strategy for creating a mapping function from the same input data. This ramification of learning paradigms results in a variation of model performances in consonance with the characteristics of the data. Generally, one model can derive more information than other models when applied to the same data [123], which suggests that a combination of algorithms can generalize better to several domains. Since the late 1970s, ensemble methodology has been used to combine several opinions of models before making a crucial decision. The term "ensemble methods" is usually reserved for a collection of same family models that are minor variants of a basic algorithm. The core principle is to train several individual classifiers and decide labels based on the weights or votes of all models. Theoretically, in a multi-strategy, the performance is equal to the best of the learning algorithm members. For large-scale data this performance can be achieved by alleviating the need for individually employing every individual algorithm to all data and simplifying the knowledge acquisition. More ambitiously, combining different paradigms may produce synergistic effects (e.g. constructing various frontiers types between class regions), leading to a level of accuracy that no individual atomic approach can achieve [92]. However, the selective superiority problem, an empirical comparison of different ensemble approaches and their variants, has shown that each approach performs best only in very limited domains [15]. Andrychowicz et al. [4] suggested that specialization to a subclass of problems is the only way that improved performance can be achieved in general. In the first part of this thesis, we cover broader families of algorithms, which fit into the definition of meta-learning. Unlike in ensemble methods, such as bagging [14] and boosting [40], where the idea is to improve algorithms by producing variations of the data or algorithms, the core concept of meta-learning is that it exploits knowledge about learning.

An approach to deal with sub-optimality in data diversity is to change models on the fly, albeit with a computationally expensive overload. A meta-model could be used to constantly look at the data and quickly select and configure the optimal model based on the previously recorded performances. For generalizing machine intelligence to real-world applications, a self-regulating model with the ability to automatically find optimal features and perform abstraction and learning methods is extremely indispensable. Learning about learning, which is recognized as a core component of human intelligence [97], originates from deutero-learning. It describes progressive changes in the rate of learning curves (i.e., proto learning). Based on the assumption that learning is stochastic and contains errors, the author in [8] suggested a hierarchical classification of learning levels and an error

correction mechanism. Error correction provides the basis for a classification of learning levels [9].

The term meta-learning first appeared in [122] to address "learning how to learn" and later the phrase was adopted for a similar concept in machine intelligence as one in human cognition. The classical definition of meta-learning [12] describes a system that includes adaptable learning subsystems with the ability of exploiting knowledge from previous learning episodes on a single dataset or from different tasks and domains. Meta-learning methods reach better classifications and are more robust than separately obtained decisions and in complex hypothesis spaces reduces manual intervention [131]. As real-time computing and the ensuing problem of complex environments have become more pervasive, the need for meta-learning models has become more acute. The level of abstracted information in the meta data is the main indicator of meta-learner performance. Meta-data provides information about the characteristics of the data and evaluation metrics of machine learning algorithms.

A challenge in designing machine learning in dynamic data is finding optimal configurations. Hyperparameter tuning can be seen as an optimization of a nonconvex and nonsmooth function in a high dimensional space. In meta-learning for hyperparameter selection the input is metadata characterizing the data and hypothesis evaluation metrics, and the output is an optimal hyperparameter configuration. The purpose of meta-learning algorithms in this context is to produce meta models that link configurations of base machine learning models with a table of performance and data characteristics. A meta-learner generates a set of experience rules that map the training data onto a certain bias. If the hypothesis exceeds the acceptable threshold, the meta model switches the base model configurations to a suitable higher performing set. The models mechanisms of learning representations from the data, resulting in variations in their performance when characteristics of the data differ. The model switching approach promises to always use the optimal model. In this study, we evaluate a best-case scenario of an assumptive meta-learning approach which can identify the optimal models and find the upper bound AUC.

2.4.3 Supervised deep learning

Deep learning methods deal with variations in data and optimally detect targets. Meta-learning, however, requires precise and constant configuration of algorithms. Meta-learning other issue is requiring an extra mechanism of extracting metadata and learning on top of the regular learning process. Low metadata precision results in potentially missing important events (e.g. sudden anomalous bursts in the data) that potentially introduces noise and exponential increase in computational complexity which results in an impractical solution for deployment. It is challenging to produce efficient metadata, transferring knowledge between meta-models presents is an even bigger challenge. Deep learning methods provide abstraction mechanisms that lower the dimensionality of the data without losing any information that could potentially decrease the error.

The time-delayed networks (TDNN), introduced in 1988, is simple ways to represent mappings between past and present values. The delays in the TDNN remain constant throughout the training procedure and are estimated before training by using trial and error along with some heuristics require injection of human expertise. In TDNNs, however,

may be the case that these fixed delays do not capture the actual temporal locations of the time dependencies. A TDNN has delay taps (e.g., t, t-1, t-2, ...) that connect past time steps to the current output with fixed weights. Deep learning algorithms have been able to provide better performance than conventional machine learning algorithms in a variety of domains and tasks [73], [77] [27].

In sequential data analysis, in particular, Recurrent Neural Networks (RNNs) are more intuitive paradigms for time series data, speech recognition and NLP, since the structure of RNNs is sequential [26], [55]. The memory feature of the RNN structures can capture time information by learning dependencies.However, although they successfully model temporal dependencies, they are vulnerable to changes in the order in which observations are processed. A recurrent neural network (RNN) exhibits temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. RNNs input current and previous states resulting in a response to any arbitrary lengths of time. The feedback structure makes RNNs a natural architecture for streams with variable length sequence inputs. RNNs can be used for learning fairly complex relationships [128] with symbolic (e.g. words, sentences), real values (e.g. digitized sounds and sensor data), and multidimensional (e.g. image, video) structures. RNNs can thus be applied to speech recognition [54], anomaly detection [108], NLP [24], [106], and handwriting recognition [53] problems. The output can be in the form of fixed length vector representation or signals, class labels (e.g. binary, softmax) or real-valued sequence. In theory, RNNs maintain a flexible length of necessary memory for the application. However, there are practical limits to the history they can retain, which limits their usage with large-scale streams. RNNs have a reputation for being difficult to train and have historically not performed well. Another problem of RNNs is that they are impractical to use when trained with traditional techniques (e.g., Backpropagation through time) for learning long term dependencies. This problem arises from the so-called exploding/vanishing gradient which basically means that as the model propagates the error signals backwards through the networks structure they tend to vanish or explode. Traditionally, RNNs are used for time-series data, speech recognition and natural language processing (NLP) because the structure of RNNs is sequential [26], [55].

A method used for time-series data is Long-Short Term Memory (LSTM) that has gained popularity because of its possibility to remember long running dependencies. LSTM is a deep learning architecture used in the field of sequential data (such as speech or video). For example, LSTM is applicable to tasks such as connected handwriting recognition, speech recognition and anomaly detection in network traffic or intrusion detection systems. LSTMs remember values over arbitrary time intervals and the gates regulate the flow of information into and out of the cells. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs, and are well-suited for classifying, processing and making predictions based on time series data. Although, LSTM units [61] provide performance improvements [25], they are highly biased to the order of the input resulting in limitations in detecting variation in patterns. In addition, LSTM consists of layers that require large amounts of memory bandwidth to

be computed. Hierarchical Temporal Memory (HTM) [44] also has the ability to learn and remember long running dependencies but in an unsupervised to semi-supervised way. It is an online machine learning method that is designed to work in dynamic environments as in sensory-motor data. Sensor input can change depending on different obstacles as metrics from a server. Instead, the input data may change, because a sensor is moving the same as human being eyes are watching at the static picture. When there is any change in input data, the memory of HTM system is updated. At each point in time, HTM makes a prediction of what it expects will happen next. In other words, HTM builds a predictive model of the world. Then, the prediction is compared to what happened and this result is a basis for learning. As a classical machine learning methods, it tries to minimise the error of the predictions. HTM learns continuously by pretending to be a universal learning method for any task, which has temporal behaviour. However, HTM method has large limitation when data contains concept drift [82].

A convolutional neural network (CNN or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery due to its shift and space invariance. Because of CNNs shared-weights architecture and translation invariance characteristics, they have been widely used in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing. One dimensional Convolutional Neural Network architectures can deal with section of time series data as the input in its raw form. In [133] and [134] the methods were tested using synthetic data. However, only a few basic convolutional neural networks algorithms exist that can deal with time-series values in numerical format as in general a generic CNNs are not designed for temporal data. Even in multivariate time series data [134], the raw form of data limits the model to detect only a few correlations between sources. 1D-CNNs also cannot deal with long-term patterns or variation in characteristics in the data over time. The following CNN architecture, for example, can handle time-series data in its raw format [133], while with our proposed transformation method almost all types of CNNs can be used for one dimensional data. Generally, deep learning models provide well performing strategies compared to conventional machine learning [73], [77]. Applying CNNs to sequential data has resulted in a few approaches that have performed better than the classical models. Studies have shown that CNNs perform much better on these problems due to their location invariance [1]. To date, however, only 1D CNN has been used for image, speech and time-series data pattern recognition [76], [112], [125]. While CNNs allow for larger input sizes by reducing the number of parameters in the learners, they often require extensive preprocessing steps to reduce the dimensionality of the raw data. Methods such as PCA have been used for this purpose [115], but they remove patterns that have light expressions and scarce anomalies.

Neurons are biologically-inspired building blocks of a method that is enable to learn representations of data. Each neuron receives a set of x_1 to x_n values as input and compute the $\hat{y}$. The input vector essentially contains the values of the features in one of m examples from the training set. Neurons have parameters, a column vector of weights, referred to as w_i and b for bias which changes during the learning process. In each iteration, the neuron calculates a weighted average of the values of the vector x passed through a non-linear activation function f. The basic source of information on the progress of the learning process is the value of the loss function showing the distance to the target. The goal is that in each iteration the value of the loss function decreases and learning is complete when the loss is

minimized using gradient descent. In each iteration the values partial derivatives of the loss function with respect to each of the parameters being calculated. With manipulating variables we can move downhill the slope with successive epoch heading towards the minimum. Backpropagation allows calculating complicated gradients to update parameters of the neural network. With a learning rate to control the speed of each step forward and backward propagation we optimize the loss function. Deep learning extends the category of neural network algorithms to architectures consisting of multiple transformation layers that create high-level abstraction inputs [73]. Deep learning algorithms have drawn researchers' attention to reconsider legacy NN approaches. In particular, the advent of CNNs has led to the development of a superior approach over traditional algorithms in various utilizations. The abstraction and learning power plus the location invariance of CNNs are persuasive aspects of the method applicability. Notably, CNNs produce high-level features by automatically learning the values of abstraction layers. This property of the deep learning models has enabled conventional NNs to generalize to a broad range of problems with well performing strategies [73], [77], [76], [112], [125].

The advent of CNNs led to several performance enhancements [73], [77], [76], [125]. Compared to conventional machine learning algorithms, CNNs [76], [112], [125], improve the generalization and allow larger input sizes by reducing the number of parameters of the learner. The abstracted input with disentangled details escapes overfitting, and the NN analogue architecture creates a spectrum function with less rigid decision boundaries [49], in the presence of concept drift.

$$y(x_1, \ldots, x_n) = f(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \tag{2.1}$$

w_i are parameters, f is the activation function

In our experimental setup data can be transformed prior to the learning.

Fourier series:

$$F(x) = \sum_{i \geq 0} (a_i \cos(ix) + b_i \sin(ix)) \tag{2.2}$$

Taylor series:

$$F(x) = \sum_{i \geq 0} a_i (x - x_0)^i \tag{2.3}$$

n inputs, one output:

$$y(x_1, \ldots, x_n) = f(w_1 x_1 + \ldots + w_n x_n) \tag{2.4}$$

Simplest activation function (McCulloch/Pitts):

$$f(v) = 1_{x \geq 0}(v) \tag{2.5}$$

However the simple neural network model could only distinguish between two linearly-separable sets (XOR gate).

Add a bias input:

$$y(x_1, \ldots, x_n) = f(w_0 + w_1 x_1 + \ldots + w_n x_n) \tag{2.6}$$

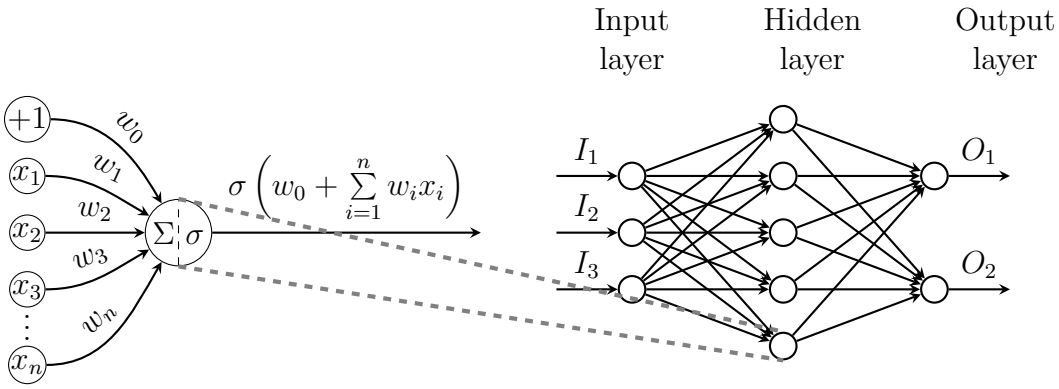


Figure 2.3: A multilayer perceptron (MLP) is a class of feedforward neural network that consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Each node is a neuron that uses a nonlinear activation function and utilizes back-propagation algorithm for training and can distinguish data that is not linearly separable [52].

Same as an input connected to the constant 1

A vector notation:

$$y(\mathbf{x}) = f(\mathbf{w}\mathbf{x}) = f_{\mathbf{w}}(x) \tag{2.7}$$

Assume we need to separate sets of points A and B.

$$E(\mathbf{w}) = \sum_{\mathbf{x}\in A}(1 - f_{\mathbf{w}}(\mathbf{x})) + \sum_{\mathbf{x}\in B} f_{\mathbf{w}}(\mathbf{x}) \tag{2.8}$$

Goal: $E(\mathbf{w}) = 0$ Start from a random $\mathbf{w}$ and improve it

1. Start with random $\mathbf{w}$, set $t = 0$
2. Select a vector $\mathbf{x} \in A \cup B$
3. If $\mathbf{x} \in A$ and $\mathbf{w}\mathbf{x} \leq 0$, then $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$
4. Else if $\mathbf{x} \in B$ and $\mathbf{w}\mathbf{x} \geq 0$, then $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{x}$
5. Conditionally go to step 2

Guaranteed to converge *iff* A and B are linearly separable.

Connect the output of a perceptron to the input of another consisting Input, Hidden and Output layers. The hidden layer maps inputs into a second space: "feature space," or "classification space". Each hidden unit computes a linear separation of the input space and several hidden units can carve a polytope in the input space and the output units can distinguish polytope membership.

An input layer with $N \times k$ neurons, where k is the variate number of input time series and N is the length of each univariate series. A convolutional layer that performs

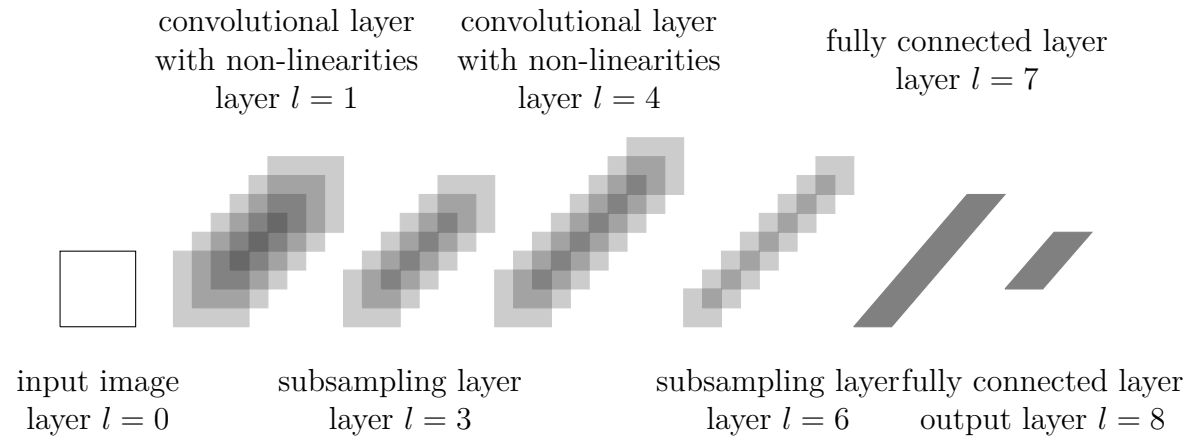


Figure 2.4: The architecture of the original convolutional neural network, as introduced by LeCun et al. (1989), alternates between convolutional layers including hyperbolic tangent non-linearities and subsampling layers. In this illustration, the convolutional layers already include non-linearities and, thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an arbitrary number of fully connected layers. The output layer usually uses softmax activation functions.

convolution operations on the time series of preceding layer with convolution filters such as, filter numbers m, stride s and the size of filter $k \times l$, where l is the length of filter. A nonlinear transformation function f also needs to be determined in this layer. A Pooling layer with a feature map that is divided into N equal-length segments each represented by its average or maximum value. Feature layer. A feature maps generates a new long time series as the final representation of the original input in the feature layer. Finally, fully connected output layer with n neurons that corresponds to n classes of time series. The most popular method is taking the maximum output neuron as the class label of the input time series in classification task.

The step function for gradient descent techniques is replace with a smooth step function:

$$f(v) = \frac{1}{1 + e^{-v}} \tag{2.9}$$

and
$f'(v) = f(v)(1 - f(v))$

And the output Activation,

$$f(v) = \frac{1 - e^{-v}}{1 + e^{-v}} \tag{2.10}$$

with Softmax probability distribution:

$$f(v_i) = \frac{e^{v_i}}{\sum_j e^{v_j}} \tag{2.11}$$

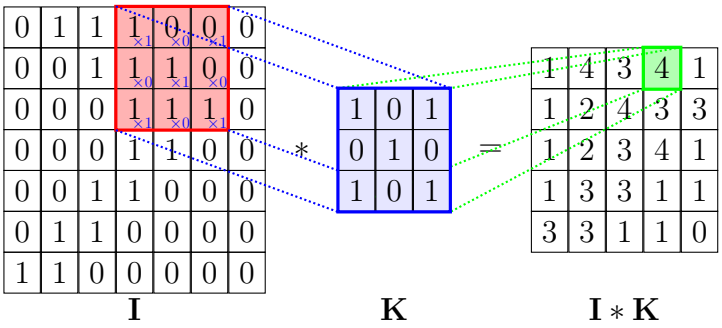


Figure 2.5: CNNs exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers as each neuron is connected to only a small region of the input volume. The CNN parameters consist of a set of learnable filters (or kernels) that each filter is convolved across the input data during the forward pass. The process computes the dot product between the inputs of the filter and produces a 2-dimensional activation map. As a result, the network learns to activate when it detects some specific type of feature at some spatial position in the input.

The Backpropagation Algorithm works with any differentiable activation function with a gradient descent in weight space which stops in different equilibrium so we can add a slight pull term:

$$f(v) = \frac{1 - e^{-v}}{1 + e^{-v}} + cv \tag{2.12}$$

Minimizing the error function,

$$E = \frac{1}{2} \sum_{i=1}^{p} \|o_i - t_i\|^2 \tag{2.13}$$

where o_i is the actual outputs, t_i is the desired outputs and p is the number of patterns.

Compute error, $\nabla E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_l}\right)$ and update weights,

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \quad i = 1, \ldots, l \tag{2.14}$$

to find a point that $\nabla E = 0$

As weights can grow uncontrollably and to avoid overspecialization, we add a regularization term that opposes weight growth and produces smoother outputs.

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} - \alpha w_i \tag{2.15}$$

CNNs exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers as each neuron is connected to only a small region of the input volume. The CNN parameters consist of a set of learnable filters (or kernels) that each filter is convolved across the input data during the forward pass. The process

computes the dot product between the inputs of the filter and produces a 2-dimensional activation map. As a result, the network learns to activate when it detects some specific type of feature at some spatial position in the input. By stacking the activation maps for all filters, each neuron corresponds to a certain region in the input and shares parameters with neurons in the same activation map. Therefore, a CNN architecture is invariance to the exact spatial structure of the data. The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space, but always extend along the entire depth of the input data. A convolutional layer that performs convolution operations on the time series of preceding layer with convolution filters such as, filter numbers m, stride s and the size of filter $k \times l$, where l is the length of filter. A nonlinear transformation function f also needs to be determined in this layer. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.

The gradient surf can stop in a local minimum and the convergence not guaranteed. Many NLP applications foster discrete features but neural networks accept real numbers and discrete inputs treating as integral numbers undemocratic. One discrete feature with n values $\rightarrow$ n real inputs, and the i^{th} feature value sets the i^{th} input to 1 and others to 0, now the Hamming distance between any two distinct inputs is constant, but results a much larger input vector. Hidden unit has all inputs zero except the i^{th} one that is just multiplied by 1. Regrouping weights by discrete input, and not by hidden unit and producing matrix $\mathbf{w}$ of size $n \times l$ results in that input i just copies row i to the output. The row $\mathbf{w}_i$ is a continuous representation of discrete feature i

In classification, n real outputs summing to 1 with softmax normalization

$$f(v_i) = \frac{e^{v_i}}{\sum_j e^{v_j}} = \frac{e^{v_i - v_{max}}}{\sum_j e^{v_j - v_{max}}} \tag{2.16}$$

Train with $1 - \epsilon$, and $\frac{\epsilon}{n-1}$ for all to avoids saturation.

Maybe the targets are known probability distribution or to reduce the number of training cycles, for feature vector $\mathbf{x}$, labels l_1, l_2, l_3 are possible with equal probability, train with $\frac{1-\epsilon}{3}$ for the three, and $\frac{\epsilon}{n-3}$ for all others. In language modeling, the input is n-gram context and outputs the probability distribution of the next word. In Lexicon learning, the Input is Word-level features (root, stem, morph) for instance the most frequent next word and outputs the probability distribution of the word's possible POSs. In Word Sense Disambiguation the input is the bag of words in context and local collocations and outputs the probability distribution over senses.

Benefiting from the local affine transformations through layers of deep learning has made CNNs completely location and representation invariant. CNNs produce high-level features by automatically learning the values of the filters. The architecture consists of several layers of convolutions often with non-linear activation functions (e.g. tanh, ReLU) and a finishing classifier layer. To achieve the properties and behaviour of RNNs, we can use large strides. The automatic abstraction, learning power and location invariance are powerful aspects of CNNs. We compare the performance between deep learning-based models and conventional machine learning models, including baseline long short-term memory (LSTM), 1D CNNs, and 2D CNNs deep models. CNNs take a different approach towards

regularization. they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Recent studies showed that CNNs can potentially perform much better on several problems [1]. To date, however, only 1D-CNNs have been used forspeech and time series data pattern recognition and 2d-CNN generally for image domain, [76], [112], [125]. While convolutions allow for larger input sizes by reducing the number of parameters in the learners, they often require extensive preprocessing steps to reduce the dimensionality of the raw data. Methods such as Principal Component Analysis (PCA) have been used for this purpose [115], but they remove patterns that have light expressions, as well as rare anomalies. Moreover, 1D-CNNs are only able to detect whether the target slice considered contains an anomaly or not, and cannot determine its specific location. Using smaller window sizes in the attempt to mitigate this problem has the disadvantage to remove possibly existing long-term patterns. Although 2D-CNNs overcome this limitation and are able to localize the anomaly, they provide only a coarse rectangle boundary and exhibit poor performance with sequential data, since the anomalies are usually dispersed in distant segments of the time series.

Following a series of deep learning breakthroughs in the area of image segmentation [47], [46], [30], [59], we can finely sub-categorize multiple objects in an input. The method efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. An airplane might crash if no warning was issued prior to the point of no return. However, some unusual signals that happened to differ somewhat from other known faulty system errors might have remained undetected. Perhaps the distance, order, shape or intensity of these signals were slightly different on this occasion. In such conditions, it seems that almost no machine learning method is able to detect anomalies, except a purely invariant method, such as CNNs. To enable the 2D CNNs to deal with sequential data and localize and classify various shapes of patterns and anomalies, we adapted a technique from the area of video segmentation [59]. More specifically, our approach is based on Region-CNN (R-CNN) architectures [47], [46], [30] and goes beyond a conventional 2D CNN to segment the target and pinpoint the exact boundaries of multiple objects on an image representation. A simple reversing function then maps the detected pattern and anomalies back from image representation to their positions on the sequence. One of the main challenges in deep learning models is the fact that they require high number of data to optimize parameters using stochastic gradient descent to minimize loss function. Supervised learning provides the desired value for the optimization algorithm to move toward minimizing the distance between output and the label value.

2.5 Unsupervised and One-class Learning

Explosive growth in streaming and sequential data is occurring across industries. This sudden increase is largely driven by the rise of the Internet of Things (IoT) and the proliferation of connected real-time data sources and applications. Sensors around the world are constantly producing waves of data. Voluminous amounts of this data are being stored but without any explicit labels that describe the contents and annotate the events. Unsupervised learning can mitigate the issue with low quality and low quantity labaled instances.

An example of Unsupervised Learning is clustering, where the model finds clusters within the data set based on the underlying patterns of the data. There are huge amounts of unlabeled data in many databases that can create value if the patterns and anomalies can be unlocked.

In One-class classification and Unsupervised Anomaly Detection, algorithms typically use an explicit representation of the distribution of normal data in a feature space and determine novel instances or outliers based on the local density at the position of the observation in the feature space. Noise, that often exist in raw data, is a phenomenon in data that is not of our interest and acts as a hindrance to analysis. The data that can be obtained is often contains noise which tends to be similar to the actual anomalies and hard to distinguish and remove. The anomaly detection problem, in its most general form, is not easy to solve. In fact, most of the existing anomaly detection techniques solve a specific formulation of the problem. In the Network Security Domain anomalies are often log patterns that deviate from regular patterns of data profiles. Unexpected bursts in the data indicate a single point of anomaly that can be associated with a popular posted video and should not be filtered out, while a defined pattern of bursts can identify an cyber-attack. Although a developed method for this data can be applied to other domain with complex pattern characteristics such as for instance, engine failure, a heart-attack in ECG data or a record-breaking temperature global pattern in the winter. The real-time anomaly detection in large-scale streams constructed by complex systems is an open research question. Sequential structure adds the complexity of order or temporal dependence to the features and requires specialized data preparation before training and evaluating models.

2.5.1 Unsupervised conventional models

Applying unsupervised anomaly detection has been studied in the academic community specially in network intrusion detection [132]. Oldmeadow, et al. [87] carried out research based on the clustering method and showed improvements in accuracy when the clusters are adaptive to changing traffic patterns. In [127], a novel two-tier IDS is proposed. The first tier uses unsupervised clustering to classify the packets and compresses the information within the payload, and the second tier used an anomaly detection algorithm and the information from the first tier for intrusion detection. Lane and Brodley [72] evaluated unlabeled data by looking at user profiles and comparing the activity during an intrusion to the activity during normal use.

One-class anomaly detection [65] uses purely normal instances as training data, to model normal data (traffic) to find novel deviations in the testset. Using self-organising maps is presented in [50], while another one uses principal component classifiers to obtain the model [102]. One approach uses graphs for modelling the normal data and detect the irregularities in the graph for anomalies [86]. Another approach uses the normal data to generate abnormal data and uses it as input for a classification algorithm [51]. Lane and Brodley [72] performed anomaly detection on unlabeled data by looking at user profiles and comparing the activity during an intrusion to the activity during normal use.

2.5.2 Unsupervised deep learning

Achieving models able to capture markers relevant to latent patterns and anomalies in data is challenging. Millions of labeled examples are required for training models. Majority of models require large amount of data with annotated instances of known labels which is impractical in many real-world problems. The generalization becomes more sophisticated when the characteristics of the data varies in different sensors or for each stream over time.

A generative adversarial network (GAN) consists of two neural networks contest with each other in a game. Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. GANs are designed to learn generative models that generate detailed realistic images [34], [33]. Radford et al. [89] showed that GANs are capable of capturing semantic image content enabling vector arithmetic for visual concepts. Zenati et al [129] leveraged simultaneous encoder learning to develop an efficient anomaly detection using the BiGAN-based [34] method. Schlegl at al. [100] proposed AnoGAN, a deep convolutional GAN, to learn a manifold of normal anatomical variability that accompanied a novel anomaly scoring scheme based on mapping from an image space to a latent space. A GAN model trained to fit the distribution of normal samples is able to reconstruct normal samples from a certain latent representation and discriminate the sample as coming from the true data distribution. However, as GANs only implicitly model the data distribution, using them for anomaly detection necessitates a costly optimization procedure to recover the latent representation of a given input example, making this an impractical approach for large datasets, real-time applications and when the characteristics of the anomalous data changes over time.

There is limited number of training data in many real-life applications. While early anomaly detection has practical and significant applications across many industries (e.g., monitoring critical IT infrastructure, detecting potential fraudulent financial transactions, understanding energy consumption, geo-tracking of vehicles in logistics networks, there is not enough labeled data in many domains. Autoencoders construct smaller sets of features to represent the information in the data that has many features. The goal is to condense the representation of the data without losing any information from the data, so called dimensionality reduction. We designed a on-dimensional to image transformation method that enables GAN-discriminator to work as an autoencoder of raw data and detect novelties. To test whether our data representation is well-represented, or whether some information has been lost along the way, we reconstruct the encoding representation of encoder back into its original features using a decoder. This is an unsupervised dimensionality reduction method as there is no label for the input features to encode. As we only have labels for the decoder but not for the encoder, we train the encoder and decoder as one large neural network rather than separately. The input features for our encoder is the input features and the label for the decoder is for the large neural network and the encoding will produced somewhere in the middle of the network in a bottleneck layer. The example of autoencoder in our context would be to encode the sequential data images into some compressed set of features and using a decoder to reconstruct the original anomalous pattern representation in the original data which called the ground-truth label. The

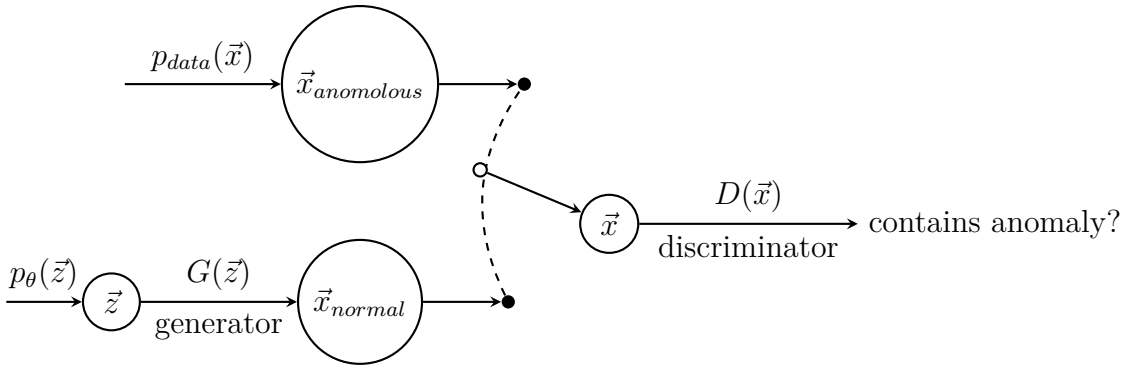


Figure 2.6: The architecture of a generic GAN. The architecture consists of two neural networks in which a model learns from the other model's loss. Given a training set, GAN technique theGenerator model learns how to generate new data with the same statistics as the training set with many similar characteristics. The Discriminator model learns how to differentiate between real-world data characteristics and the data generated by the Generator model [52].

auto-encoder reduce the feature dimensions to an encoding that can be used for detecting outliers. We suggest a GAN-based framework for unsupervised learning of sequential data. The DriftGAN is a framework for transforming sequential data to images using automatically sliding window methods and a GAN architecture for detecting and localizing patterns and anomalies in sequential data without assigning labels. This method combined with our optimization method for finding flexible sliding window parameters enables the DriftGAN to learn complex and shifting paradigms. DriftGAN provides an architecture enabling unsupervised learning to identify novel patterns and anomalies in imaging data. Our sequential to image transformation technique, enables a deep convolutional GAN model to learn the distribution of normal data and perform segmentation on anomalies pixel-wise. Our other contributions is to provide an automatic way for slicing time-series data and enabling computer vision algorithms to be applicable to one dimensional non-image data and extending the solution from classification to detection, segmentation and generative models to find patterns and anomalies in streaming data.

We extend the pattern and anomaly detection solution to an unsupervised approach by adopting GAN architecture to our data representation of sequential data and designed DriftGAN.

- **Generative**: generative models create a joint probability distribution.
- **Discriminative**: a discriminative model defines the conditional probability of the target, given an observation.

2.6 Transforming Sequences to Images

One-dimensional CNNs have long been used for image, speech and time-series data pattern recognition [76], [112], [125]. Although 1D CNN models are generally used for temporal data learning, they often require extensive preprocessing steps to reduce the dimensionality

of the raw data. Two-dimensional CNNs have previously been used in the context of time-series data but in a limited way. The proposed methods attempted to map the entire data from their Cartesian coordinates to their polar coordinates [114], [80]. These approaches use CNNs for time-series analysis, but they look at the data as a whole assuming that all the data (past and present) fits into the computational memory simultaneously and one single image. These suggested methods thus seem impractical in real-world temporal data analysis because they are not scalable, especially when considering the growth rate of certain types of data. Furthermore, these methods of data transformation create images with scarcer representations and higher amounts of noise, and they remove important aspects of the data, resulting in a poor performance in large-scale data [113].

A study with conversion of real-world road traffic data to image compared CNNs to methods such as least squares, k-nearest neighbors, neural networks and random forest, as well as deep learning architectures including stacked autoencoder, recurrent neural network, and long-short-term memory [80]. In their study the CNN outperformed other algorithms on testing data with an average accuracy improvement of 42.91 percent and performs best in long-term predictions compared with other algorithms. Traditional traffic prediction models often treat traffic data as sequential data, ineptness to deal with outliers, noisy or missing data, and inability to the entire road network rather than on a single road. this approach helps to provide solutions to make better route choices and to support traffic managers to allocate resources systematically. The transformation of transportation traffic data to image domain creates a representational abstraction of spatial correlation of vehicles into a two-dimensional plane that enables longer-term congestion prediction. However, the study states that the selection of hyperparameters only relies on experts' experience and no general rules can be applied directly. This manual process can be cumbersome, error prone, suboptimal and expensive. Also one image for the whole data could potentially increase underfitting and is impractical for large-scale and streaming data.

Wang et al encode time series to a single image for classification using Tiled-CNN [115], [113]. In their study inspired by recent successes of deep learning in computer vision and speech recognition, a framework suggested to encode time series data to Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) image representation to build the Markov matrix of quantile bins after discretization and encode the dynamic transition probability in a quasi-Gramian matrix. The image represent time series in a polar coordinate system and each element is the cosine of the summation of angles. However, if the data is large a single image method could produce extremely low performance. One frame for the whole data in a polar coordinate system and using GAF or MTF images that are unnatural images and remove concepts such as "edges" and "angles" that are important features for detecting patterns and anomalies. The study, however, does not cover large-scale data since it could not be fit in one single image, as well any anomaly detection task or concept drift data. The study demonstrates that the integration of data into one compound image decreases the risk of overfitting but might result in high degrees of underfitting [109]. We propose an algorithm that automatically find optimal sliding window parameters on the fly and a tensor representation of 2D images that handles stream and large-scale data.

2.7 Hypothesis

1-D CNNs perform well in detecting labels for a large slice of data but cannot pinpoint the instances that produce a label (e.g. the exact sequence of anomalous instances that resulted in a plane crash). 2-D CNNs cannot localize either.

What if, instead, of doing what others have done, we did a transformation that make data compatible for instance segmentation method, and in addition create an stream of images. What if we randomly create images and subtract them to find the optimal size that removes seasonality and repetitive long-term concept drift to improve performance.

We expect that to solve problems: detecting long patterns, generalizing to concept drift and pinpointing the location of individual instances that result in the change of a label

Chapter 3

Methodology

The main goal of this thesis is to provide a machine intelligence technique for anomalous pattern detection in sequential and time-series cyclical data with dynamic characteristics and when the concept drifts. Our assumption is that data contains *real concept drift* (see Fig 2.2) with repetitive normal patterns and we use an *offline* learning approach for training and to evaluate models in offline batch processing setups. We also evaluate the trained models in an online mode with drifting data forming a stream which is potentially infinite. To illustrate the complexity of real-world applications we use a combination of four types of concept drift. In each one-dimensional data source we generate mean changes that may happen *suddenly* by switching from one concept to another, *incrementally* consisting of many intermediate concepts in between, and *recurrently* or *gradually*. In addition we generate other types of data characteristic changes such as the amplitude and duration of seasonality and noise and other aspects of the data that might affect the P(X) and P(Y|X) simultaneously. In the meta-learning part we use reactive strategy and depending on whether a change or specific data descriptors is detected the meta-model proposes the optimal model and its hyperparameters (i.e., *Informed Adaptation*) [43]. In the deep learning studies we propose a method that does not require any explicit detection of changes. Instead of using a typical approach of fixed size sliding windows and an updating method such as in Blind Adaptatiom, in which the model is periodically retrained using the latest samples, we propose a data transformation technique. We introduce novel algorithms that perform supervised and unsupervised anomaly detection in sequential data with concept drift [43]. We also use a combination of real-world anomalies collected from traffic data of fibre-optic switches in a cyber-security domain and synthetic concept drift to evaluate our models.

The descriptions herein aim to define method for detecting and localizing anomalies in a data with drifting concept, such as in computer network performance monitoring. In this chapter, we explain the methodology and discuss the main challenges when designing our experimental setup. We explain the training and evaluating methods using data with various sizes, trends, seasonality amplitudes and probability of anomalies, and how by progressively altering the characteristics in the test set, we evaluate the models' generalization performance to novel data. Overall, we generated more than 14000 combination of data characteristics, preprocessing methods, model types and hyperparameter configurations to test the highest and average achievable performance of each model. In this section, also

a general guidance is described for an experimental setup and how synthetic time-series data was generated. Further, the properties of temporal data are demonstrated in a simple composition of synthetically generated data and the effects of altering data characteristics on the performance of anomaly detection. Furthermore, it is explained how meta-learning and deep learning are compared for the task of anomaly detection in time-series data.

To perform the experiment, a combination of four types of concept drifts were synthesized by generating mean change, as visualised in Fig 2.1. We evaluated the performance of the models when data characteristics changes from training set to the test set (e.g., amplitude of anomalous instances, duration of seasonality, level of noise). We generated several base normal data by altering each data characteristic (e.g., trend, amplitude of seasonality), while step by step increased the intensity of changes. Then, real anomalous patterns, which were collected from cyber-security domain, were added to the base data. For every combination of baselines and anomalous data, we trained a new model and tested against all other generated data and we plotted AUC of models while the data characteristics change.

In cybersecurity domain attacks occur in various patterns. We are concerned precisely with those attacks that have patterns. For example, a denial-of-service (DoS) attack often starts with short bursts in the traffic after an initial ping before flooding with an excessive amount of traffic, while brute force attacks use short messages of attempting various keys to gain access to a particular system. Attackers often take the control of large number of host machines (in a distributed attack) to perform the attack and hide their footprints. An effective anomaly detection system, therefore, has be able to learn various types of relatively short and less obvious to long and complex patterns.

3.1 Data Characteristics

In the data generating phase, our assumption is that it contains repetitive normal cycles. The data also includes instances that shape anomalous patterns. An anomalous pattern can be determined in the form of a known order or position of spikes in the sequence or as an unusual pattern of spikes in a sequence. The anomalies have a significantly lower probability to occur compared to normal patterns. Our data contains real concept drifts, and we used an offline learning approach for training and evaluating the models in a batch processing setup. However, we also use an online method to evaluate the trained models. Since in many real-life applications, the data comes in the form of a stream, which is potentially infinite.

Since deep learning models require large number of data samples for training models, the training-set was prepared based on a known procedure for generating large scale network traffic. However, the characteristics of the network traffic data is similar to many other domains' data with sequential and cyclic nature. We investigate a data simulating accumulation of several data streams in the form of network traffic, for example, in a large data centre.

We collected over one thousand real-world cyber-attack anomalous patterns as a benchmark. In addition to evaluate robustness of trained models, we added artificial concept drift and variation in data characteristics. Temporal and sequential attributes require dif-

ferent treatment compared to the individual time independent instances. Our approach to demonstrate the performance of anomaly detection in large size time-series data is to first create a controllable abstraction of normal data and then add labeled (real-life) anomalies. Thus, the properties of the data are specified including the number of sensors, time stamps and interval durations, maximum and minimum range of signals. Next, trend, seasonality, and noise are added to original data and subsequently the probability and amplitude of spikes as anomalies are combined as well as gradual and abrupt linear or exponential trend transition as concept drift. As illustrated in 3.21 for the preprocessing, a lagging step is employed that slides a window with certain size over the time-series. We generated 20 datasets with different rates of concept shift, and different frequencies and amplitudes of anomalies as defined in [116], [21]. In order to capture the seasonality correlations in conventional models, sizes of slides are chosen equal to human behavior for various activities. For instance, the window sizes could include one day worth of samples, one week worth of samples, one month worth of samples, or samples over any other suitable time period corresponding to the cycles of the signal. Another aspect of defining windows is the decision of how many steps should be taken for each sliding slice which describes the overlap or strides of the window. Additionally, one may want to execute a first difference estimator to eliminate trend and seasonality which might not be an ideal action in case of searching for anomalies that correlate to long-term changes in the time-series of sudden shifts. Further, in the result chapter we describe the effects of altering data characteristics, preprocessing configuration and models hyper-parameters on the performance of anomaly detection in several machine learning algorithms and various Convolutional Neural Networks to find the optimal set of models and adjustments for every characteristic. Furthermore, a comprehensive comparison was performed between the performance of meta-learning of machine learning algorithms and deep learning models in detecting anomalies. The data model used the following approximation for generating every sequence. Then by altering each parameter, we created various data for adding various types of concept drift and testing the generalization of the model.

$$\begin{aligned} x(t) &= m(t) + \sqrt{am(t)}z(t) \\ where: m(t) &= l(t)[K + s(t)] \end{aligned} \tag{3.1}$$

where $x(t)$ is the data sequence, $m(t)$ is the mean of the data, which changes over time, and $z(t)$ is the random component of the sequence. $l(t)$ is the long-term trend and $s(t)$ is the seasonal component of the data (the diurnal portion). Generally, the long-term trend $l(t)$ is an increasing function of time.

In Eq. 3.2, a,b and c are the components correlated with periods of time (e.g., day, week, year).

$$s(t) = M\Big[\sum \frac{a}{b} sin\Big(\frac{2\pi(t - 1/4)}{cT_d}\Big)\Big] \tag{3.2}$$

and,

$$l(t) = 1 + \frac{\alpha}{\Delta}t \tag{3.3}$$

where Δ is the period over which the data volume increases by α. Assuming that the data properties are estimated from a sampled sequence of the observed measurements,

$$x[n] = x(t_0 + nT_s), \qquad 0 \leq n \leq N - 1 \tag{3.4}$$

where t_0 is the time that sampling collection started and T_s is the time taken between samples (see Eq. 3.4).

The process used to generate streams can be considered to be a random variable X from which the objects O_i are drawn at random to present various instantiations of a random variable. Further, the variation of the characteristics was added with the probability distribution $P(X)$ to denote the prior probability distribution over covariates. Furthermore, we created a class variable or label y, where Y denotes a random variable over class labels, and the covariates x, where X denotes a random variable over vectors of attribute values. Therefore, $P(Y)$ denotes the prior over the class labels and $P(Y|X)$ denotes the likelihood distribution over class labels given objects O_i and $P(X|Y)$ to denote the posterior probability distribution over covariates given class labels.

To perform the experiment, a combination of four types of concept drift are synthesized. In each one-dimensional data source we generate mean changes that may happen suddenly by switching from one concept to another, incrementally consisting of many intermediate concepts in between, recurrently or gradually as visualised in Fig 1.1. In addition to concept drift, in real life other data characteristic changes might also happen that simultaneously affects both $P(X)$ and $P(Y|X)$. In such cases the data characteristic changes might not be directly categorized as real or virtual concept drift. These changes are fundamentally affecting the nature of the data. We can observe such changes when a trained model from a domain A is being tested on another domain B, while the targeted anomalous patterns is relatively the same in both A and B domains. For example in computer network traffic the definition of high or low traffic changes if a new datacentre moves to a new location and connects to the network and at the same time new generation technology standards for broadband cellular networks increase the definition of high traffic (e.g., 5G networks). In this scenario both data characteristics and the label probability change, but the anomalous patterns remain relatively the same. In this thesis our goal is to train models that can detect these anomalous patterns but at the same time the model is not sensitive to concept drift or characteristics changes. To demonstrate characteristic changes in addition to concept drifts, we evaluate the performance of models when data characteristics changes from the training set to the test set (e.g., the amplitude of anomalous instances, duration of seasonality, level of noise). Once the base data generated we add real anomalous patterns collected from the cyber-security domain to evaluate models' performances. In order to simulate complex and dynamic data characteristics, we present some preliminary approaches that are frequently adopted in sequential data. Subsequently, we present our proposed method in detail, along with the experimental results obtained with our datasets. Our data generator was designed in collaboration with security experts in network traffic analysis. Specifically, the data generation approach followed complies with the standards described in [70], [78] and [93], and the data extracted resembles the data distribution of real network logs collected in the company. We consider several parameters such as data amplitude, time series length, number of sensors, minimum and maximum amplitude for

each sensor, trend percentage in the short and in the long-term, seasonality shape and intensity, noise duration and intensity, amplitude and shape of anomalies. By altering each parameter, we created several time series to test the generalization capabilities of models. Specifically, we generate random numbers between 0 and 1 to simulate sensor measurements, and add linearly or exponentially increasing numbers, to obtain a defined trend in data. After adding seasonality and noise, we add larger random numbers as anomalies, with a very low probability. Subsequently, we defined the probability and the amplitude of abrupt bursts in the data for burst anomalies. We also added the sudden linear or exponential trend transition or concept drift to the data. The motivation for adopting a custom data generation approach instead of simply using benchmark datasets, stands in the possibility to control each single characteristic for multi-dimensional data. More specifically, our method is tailored to generate a range of datasets with the same characteristics, which progressively differed in a single aspect. The main advantage of this approach is that we are able to assess the performance of each method on a specific type of drift.

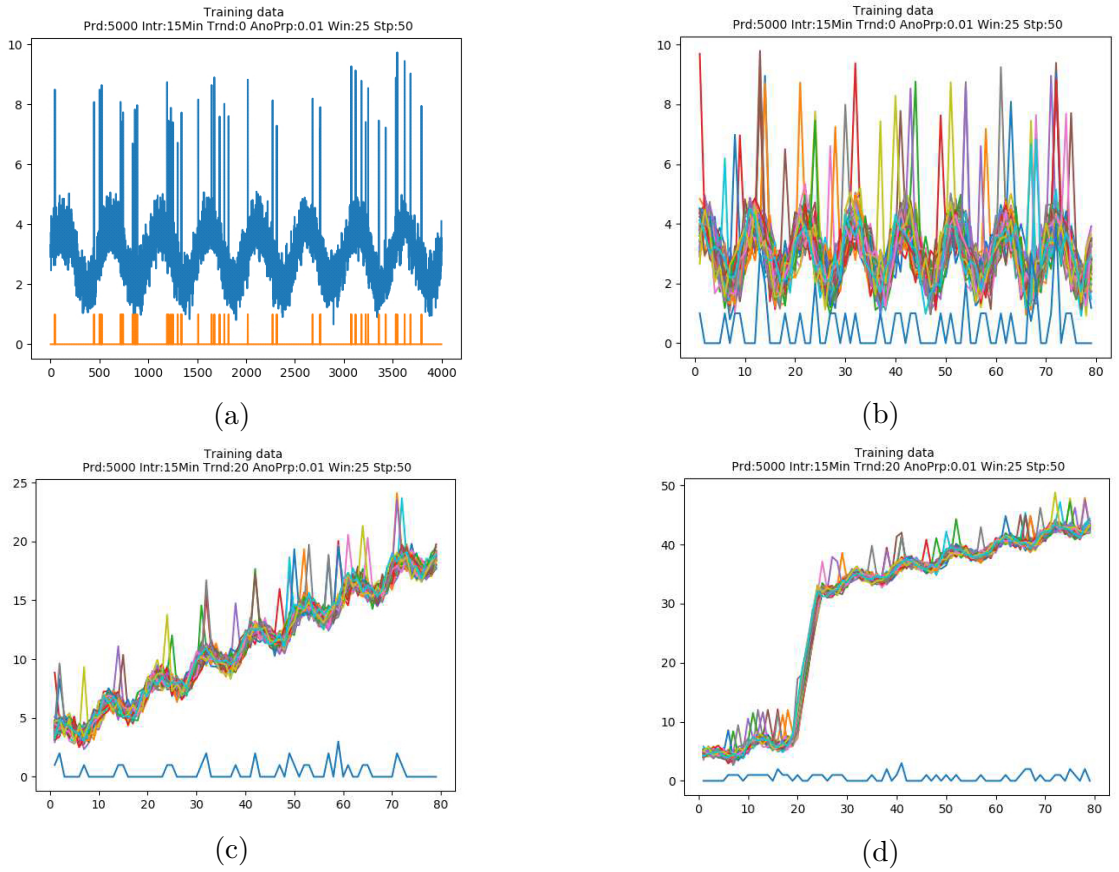
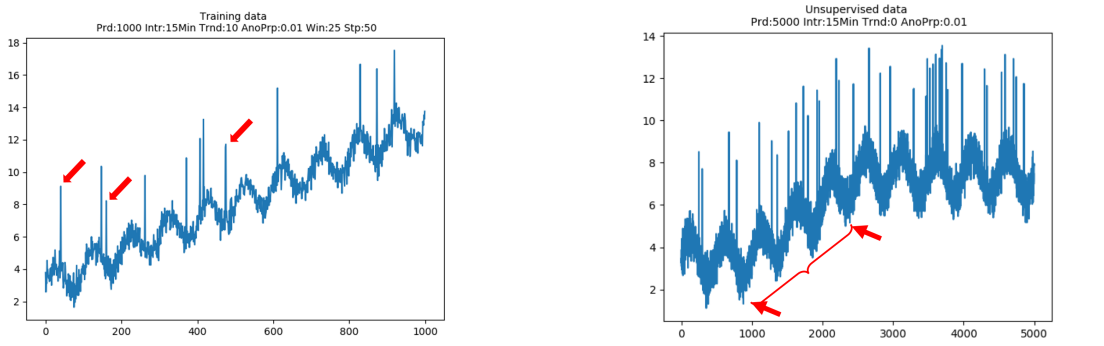


Figure 3.1: Example of data in which sudden bursts are considered as anomalies, and the labels indicate normal and anomalous data (top left). Sliding windows displayed with stacked color lines (top right). Normal trends added to the data (bottom left). Data before and after abrupt trend changes, known as concept drift (bottom right).

In the initial phase we perform an offline approach and do not stream data. However, once the model has been trained, we evaluate its performance on various concept drifts.

In particular, we divide the raw data into two parts. The first 80% data are used as the training set, and the remaining 20% for the test set. The test set was not used in the training phase, but was a holdout subset. Once the model was trained, we used the holdout dataset to evaluate the model performance. We applied various concept drift and characteristics changes to the test set and re-evaluated the same method while slowly increasing the intensity of concept drift using parameters. Using this method and calculating the maximum and average achievable AUC of a model, we tested the performance of models when the severity of concept drift and data characteristics change over time. The trained model was then tested on streaming data by transforming the sequence sliding window to images and evaluating the performance for every new image.

The software applications of the present systems and methods use relevant Performance Monitoring (PM) data along with other data to describe the behavior of a telecommunications network. The network can include an optical layer (e.g., Dense Wavelength Division Multiplexing (DWDM), etc.), a Time Division Multiplexing (TDM) layer (e.g., Optical Transport Network (OTN), Synchronous Optical Network (SONET), Flexible Ethernet (FlexE), etc.), a packet layer (e.g., Ethernet, Multiprotocol Label Switching (MPLS), Internet Protocol (IP), etc.), and the like. Those skilled in the art will recognize actual network implementations can span multiple layers. The present software applications can operate at a single layer or concurrently at multiple layers. Each of these layers can include associated PM data which describes the operational status over time at the layer.



(a) (b)

Figure 3.2: A raw sequential data example. The red arrows indicating some of spikes to be detected as abrupt anomalies. 1-D CNNs are limited in detecting multiple types of anomalies and localizing their exact positions (left). Longer term anomalies, the notion is very similar to trend and distinction requires long term analysis (right).

Examples of PM data include, without limitation, optical layer data, packet layer data, service and traffic layer data, alarms, hardware operating metrics, etc. The optical layer data can include pre-Forward Error Correction (FEC) Bit Error Rate (BER), post-FEC BER (estimate), number of corrected errors, chromatic dispersion, Polarization Dependent Loss (PDL), Estimated Optical Signal to Noise Ratio (OSNR), latency, TX power, RX power (total, individual channels), power loss, Q factor, fiber type and length, etc. The packet layer data can include port level information such as bandwidth, throughput, la-

tency, jitter, error rate, RX bytes/packets, TX bytes/packets, dropped packet bytes, etc. The service and traffic layer data can be Time Division Multiplexing (TDM) Layer 1 (L1) PM data such as Optical Transport Network (OTN). The packet layer data can be associated with a device port while the service and traffic layer data can be associated with a particular L1 connection/service. The alarm data can be various types of alarms supported by a network element (e.g., chassis, MPLS, SECURITY, USER, SYSTEM, PORT, SNMP, BGP-MINOR/WARNING/MAJOR/CRITICAL, etc.). The hardware operating metrics can include temperature, memory usage, in-service time, etc. Video quality metrics includes the number of re-buffering events, number of video codec rate changes, or data submitted by users such as thumbs up or thumbs down after a video conference call.

Throughout, the term "network element" (NE) can interchangeably refer to any of a variety of network devices, such as nodes, shelves, cards, ports, or even groups of such NEs. Regardless of the identity of the elements, however, the technique described herein for determining the normalcy of their behavior remains similar and remains valid as long as the relevant data for each element is accessible to the anomaly detection software application.

The systems and methods of the present implementation include building a single trend from multiple PM data time-series and using the single trend to predict network anomalies for proactive actions. Both these techniques can be implemented in a machine learning engine that can use arbitrary PM data from any device type, any vendor, etc.

To investigate the effect of the pre-processing steps, we performed a side study on the effect of variation of one characteristic in the training set, and various variations of test sets. For this purpose, we applied a first-difference estimator [23] to eliminate trends and seasonality, which negatively impact the ability to identify anomalies that correlate to long-term changes. To demonstrate the performance on the anomaly detection task with large-scale sequential data, we first created a controllable abstraction of normal data, and subsequently added anomalous instances to test the algorithms. Figure 3.2 shows one example of generated data. We generated 20 base datasets with different rates of concept shift, different frequencies and amplitude of anomalies, etc. The sequential data obtained were sliced into several bins using sliding windows. The labels indicate normal and anomalous data.

In non-sequential data, the notion of attributes is usually defined prior to the data science (e.g. color, weight, structure). However, in data with ordered instances (e.g. time-series data), the features have similar natures but are entangled. For this reason, the patterns in the data should remain untouched to produce trustable insights, by localizing the target objects and their relative locations in the sequence. The temporal nature of these types of problems results in gradual or sudden changes in the data characteristics. Additionally, the data patterns might repeat or evolve over time, which thus need to be separated from a specific pattern or anomaly. The two main properties of this type of data, concept drift and high complexity, make the design of machine learning solutions that generalize to unseen data an immense challenge. Motivated by the necessity to assess the anomaly detection performance of different algorithms on large-scale sequential data, we designed a data generator and added real anomalous patterns to the generated data. Our data generator was designed in collaboration with security experts in network traffic analysis. Specifically, the data generation approach followed complies with the standards described in [70] and [93], and the data extracted resembles the data distribution of real

Table 3.1: Generating variation in data characteristics and concept drift.

Parameter	**Default Value**
Start point	*2020-01-01 12:00:00*
Periods (time instances)	*50000*
Intervals	*15Min*
trend (percentage of growth)	*0*
Sensor-1 minimum	*1*
Sensor-1 maximum	*1*
Sensor-2 minimum	*1*
Sensor-2 maximum	*1*
Sensor-3 minimum	*1*
Sensor-3 maximum	*1*
Probability of anomaly	*0.01*
Spikes amplitude	*5*
Seasonality	*10*
Window size	*48*
Overlap size	*0*
Multiplication of anomaly amplitude	*1*
With differentiate	*yes*
Drift beginning in percentage	*0*
Drift end in percentage	*1*
Drift Angle in percentage	*1*
Drift type (linear or exponential)	*linear*

network logs collected in the company. Figure 3.2 shows an example of the data addressed. We generated several datasets with different rates of concept shift, and different frequencies and amplitudes of anomalies.

After specifying the basic properties of the data including the number of simulated sensors, timestamps, interval duration and the maximum and minimum range of signals, we added trend, seasonality and noise to the original data. We then defined the probability and amplitude of abrupt bursts in the data as burst anomalies. We also added the sudden linear or exponential trend transition or concept drift to the data. The motivation for adopting a custom data generation approach stands in the possibility to control each single characteristic for multi-dimensional data. More specifically, our method is tailored to generate a range of datasets with the same characteristics, which increasingly differed in a single aspect. The main advantage of this approach is that we are able to assess the performance of each method on a specific type of drift. During running our source code, the method prompts to input value for generating the training set characteristics, as shows in Table 3.2.

3.2 Concept drifts characteristics

For classification learning, y dom(Y), where Y denotes a random variable over class labels, and the covariates x $\in$ dom(X), where X denotes a random variable over vectors of attribute values. A concept is the prior class probabilities P(Y) and class conditional probabilities P(X|Y). As P(Y) and P(X|Y) uniquely determines the joint distribution P(X,Y) and vice versa, a concept as the joint distribution P(X,Y), as proposed by Gama et al. [43] that we adopt in this thesis. The main goal of this thesis is to provide a machine intelligence technique for pattern and anomaly detection in sequential and time-series data that can be generalized to data with dynamic characteristics and when the concept of the data drifts. Our assumption is that data contains *real concept drift* with repetitive normal patterns and we use an *offline* learning approach for training and evaluate the models in offline batch processing setups as well as online setups with real time drifting data, forming a stream which is potentially infinite. To illustrate the complexity of real-world applications we use a combination of four types of concept drift. In each one-dimensional data source we generate mean changes that may happen *suddenly* by switching from one concept to another, *incrementally* consisting of many intermediate concepts in between, *recurrently* or *gradually*. In addition to sudden concept drifts we evaluate the performance of models when data characteristics changes from training set to the test set (e.g., amplitude of anomalous instances, duration of seasonality, level of noise). We also use real anomalies from cyber-security domain to evaluate our models performance. In meta-learning part we use reactive strategy and depending on whether a change or specific data descriptors is detected we propose the optimal model and hyperparameters *Informed Adaptation* [43]. In the deep learning studies we propose a method that similar to blind adaptation technique does not require any explicit detection of changes. Instead of using a typical approach of using fixed size sliding windows that periodically retrain the model with the latest samples and example weighting, we propose a data transformation technique and algorithms that perform supervised and unsupervised anomaly detection in sequential data with concept

drift [43].

To illustrate the performance of models in several types of concept drift, we used gradual, incremental, recurrent and sudden concept drifts, individually and as combinations. Gradual and Incremental concept drifts are common types of drift in many real-world applications. For example, they occur when network traffic increases with population growth. Recurrent concept drift can happen, for instance, in the reads of a temperature sensor with changes of seasons. Sudden concept drift can take place, for example, with changes of regulations, such as abrupt changes in air travels due to the recent global pandemic. However, the definition of concept drifts and anomalies can vary in different domains. As seen in computer network traffic, an unanticipated change of patterns often corresponds to an anomaly.

For concept drift in each characteristic, a (10×10) study for each model is performed. We investigate the effects of data and preprocessing on the performance by testing various combinations of data types. We consider several parameters such as data amplitude, time series length, number of sensors, minimum and maximum amplitude for each sensor, trend percentage in the short and in the long-term, seasonality shape and intensity, noise duration and intensity, amplitude and shape of anomalies. By altering each parameter, we created several time series to test the generalization capability of models. Specifically, we generate random numbers between 0 and 1 to simulate sensor measurements, and add linearly or exponentially increasing numbers, to obtain a defined trend in data. After adding seasonality and noise, we add larger random numbers as anomalies, with a very low probability. Subsequently, we defined the probability and the amplitude of abrupt bursts in the data for burst anomalies. We also added the sudden linear or exponential trend transition or concept drift to the data. Figure 4.13 illustrates an example of the plots obtained by the models trained with base data characteristics and tested against its variations. We remove the trend and seasonality components from the data to show the effects of the anomaly amplitude alteration. In this example when the altitude of the anomaly compared to the base normal data is very low in the training set, Naive Bayes (NB) and multi-layer perceptron (MLP) outperform the other methods, where the x-axes are various data characteristics and the y-axes are the AUC of the algorithms.

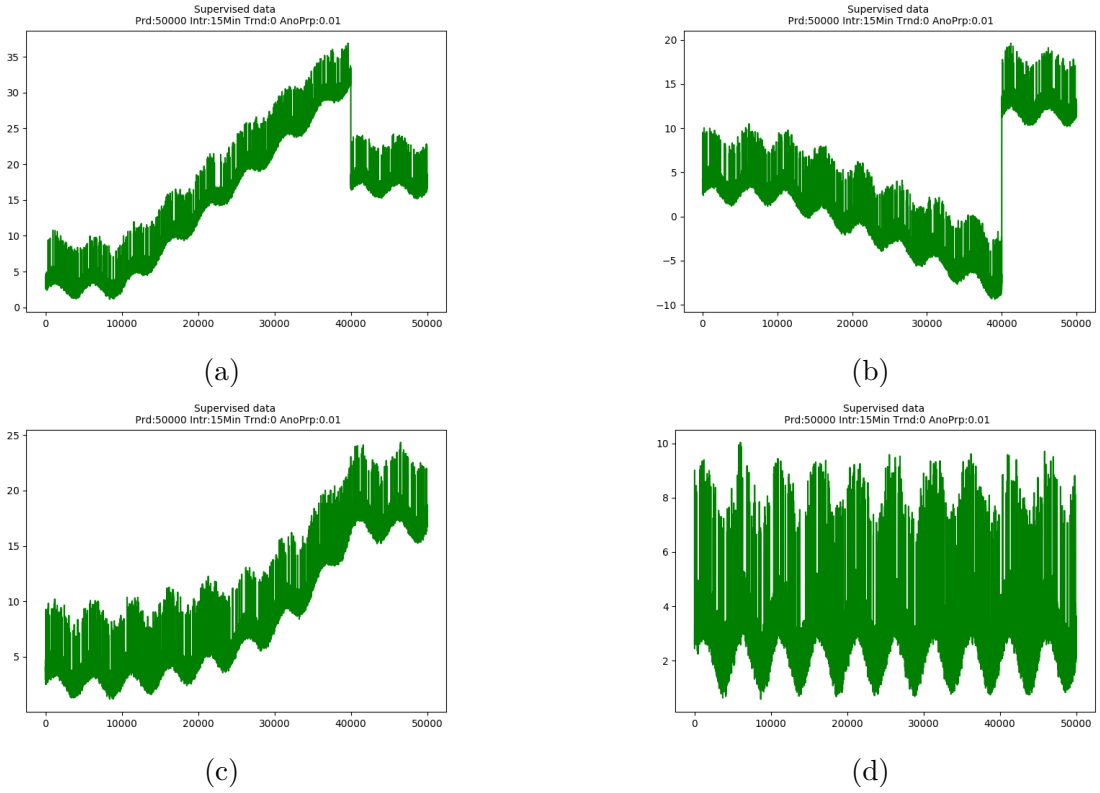


Figure 3.3: Examples of anomalies in network security domain. Concept drift and anomalies could be very similar in some domains. Depending the domain a type of concept drift might considered as an anomaly or vice versa. In some of the domain a type of concept drift might never happen, and could be an anomaly that has similar characteristics to a concept drift at its initial stages.

3.3 Selecting models and hyperparameters

In the first part of our study, we focus our attention on seven conventional machine learning algorithms. Specifically, we consider Random-Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), AdaBoost (AB), Naive Bayes (NB), K-Nearest Neighbour (KNN) and Multi-layer Perceptron (MLP). To eliminate the effects of the hyperparameter configuration on the performance of the models, we perform an extensive grid search for each parameter involved in each algorithm, in order to find the optimal hyperparameters over all datasets. The ranges of values considered for all the algorithms are reported in Table 3.2. We demonstrate the properties of conventional models and their performance while the compositions and characteristics of the synthetically generated data drifts. We also discuss the preprocessing and evaluation methods used to obtain results. Our data model exploits the approximation described in Eq. 3.1 to generate every sequence. Then by altering each parameter, we created various datasets to test the generalization performance of the model. It is well known that each category of learning algorithms follows a unique strategy to create a mapping function from the same input data. This leads to

Table 3.2: Ranges of values for the hyperparameter optimization via grid search.

Algorithm	**Parameter**	**Minimum Value**	**Maximum Value**	**Step Size**
Random Forest	*n_estimators*	1	2000	3
Random Forest	*max_features*	*auto, sqrt, log2*		None
SVM	*c*	0.0001	10	3
SVM	*kernel*	*linear, rbf, poly, sigmoid*		None
DT-C4.5	*max_depth*	1	100	3
Adaboost	*n_estimators*	5	300	3
kNN	*k*	1	30	3
MLP	*hidden layer size*	1	20	3

a variety of performance distribution models which exhibit different strengths and weaknesses in each type of data characteristics. In general, algorithms are successful when their biases match the features of the application domain. Therefore, when working in complex or dynamic environments in which the characteristics of the data change over time, models become sub-optimal and they need to be replaced by new ones. In the first part of this study, we focus on widely adopted conventional machine learning algorithms. We test their performance on sequential data characterized by concept drift and evaluate whether some algorithms are superior to others. In order to optimize their performance, we adopt an idealized meta-learning approach that selects the best a-posteriori conventional method and configuration in each data setting.

In our conventional model experiments, we perform a pre-processing step adopting a sliding window approach commonly used in time series analysis. To capture the seasonality correlations, we chose a fixed sliding window size equal to a specific time frame that represent human behaviour activities. For instance, the windows sizes can contain 24 hours, or a week/month worth of samples, corresponding to a cycle of the signal. In this case patterns repeat close to each other and become easier to detect. We also defined how many steps should be considered for each sliding slice, i.e. the overlap or stride between two adjacent windows. To investigate the effect of the pre-processing steps, we performed a side study on the effect of variation of one characteristic in the training set, and various variations of test sets. For this purpose, we applied a first-difference estimator [23] to eliminate trends and seasonality, which negatively impact the ability to identify anomalies that correlate to long-term changes. For preprocessing, we performed a lagging step that slides a window with a certain size over the time-series data. To capture long term patterns and seasonality correlations, we chose sizes of slides equal to human behaviour activities whenever the data was produced by human behaviour. For instance, the windows sizes could contain 24 hours, one week or a month worth of samples that often correspond to the cycles of the signal. Using this approach patterns repeat close to the previous location of instance and helps the conventional machine learning models produce higher degrees of detection. We also defined how many steps should be taken for each sliding slice, which describes the overlap or strides of the window. To investigate the effect of preprocessing, we performed a side study on the effect of variation of one

characteristic in the training dataset and various variations of test sets. To execute this, we first applied a first-difference estimator to eliminate trends and seasonality, which is not ideal when searching for anomalies that correlate to long-term changes; however, here we only detect sudden bursts of data during concept drifts. The details of hyperparameters of models are provided at the end of this chapter. We also perform a pre-processing step adopting a sliding window approach commonly used in time series analysis. To capture the seasonality correlations, we chose a fixed sliding window size equal to a specific time frame that represent human behaviour activities. For instance, the windows sizes can contain 24 hours, or a week/month worth of samples, corresponding to a cycle of the signal. In this case, patterns repeat close to each other and become easier to detect. We also defined how many steps should be considered for each sliding slice, i.e. the overlap or stride between two adjacent windows.

We used Informed Adaptation to form a reactive strategy for evaluating conventional machine learning models. The meta-model suggests the optimal model and its hyperparameters. For deep learning methods, we propose the APMS and FAPMS algorithms that are similar to Blind Adaptation. Instead of detecting changes as in Informed adaptation, the algorithms perform the optimization in equal time intervals and require no injection of human expertise. To extract statistically valid results on a broad variety of data, we perform extensive experiments considering 1400 different configurations for each parameter of the data characteristics (7 $[Model] \times 20$ $[Training\ Set] \times 10$ $[Test\ Set]$). Once models are trained with data of a specified size, trend, seasonality amplitude and probability of anomalies, we increasingly altered data characteristics in the test set, in order to evaluate the generalization ability of models with data in the presence of concept drift. Overall, considering concept drift in each characteristic, we perform a study of (10×10) configurations for each model. We investigate the effects of data pre-processing steps on the anomaly detection performance by testing various combinations of data types.

3.4 Proposed evaluation method

Comparing the performance of anomaly detection algorithms is not as straight forward as in the classical cases due to highly imbalanced data. In contrast to simply compare an accuracy value or precision-recall, the order of the anomalies should be taken into account. In classification, a wrongly classified instance is for sure a mistake. In anomaly detection however, a large dataset that contains few anomalies that are ranked among the top-n outliers. Therefore, detecting a few is still considered good result, even if it is not perfect. To this end, a common evaluation strategy for anomaly detection algorithms is to rank the results according to the anomaly score and then iteratively apply a threshold from the first to the last rank. This results in N tuple values (true positive rate and false positive rate), which form a single receiver operator characteristic (ROC).

The area under the curve (AUC), the integral of the ROC, can be used as a detection performance measure. The AUC is the probability that an anomaly detection algorithm will assign a randomly chosen normal instance a lower score than a randomly chosen anomalous instance. Nevertheless, AUC based evaluation has been evolved to be the de facto standard in anomaly detection, most likely due to its practical interpretability [48],

***k-Nearest Neighbors algorithm* (*k*-NN)**					
-K	KNN	Number of nearest neighbours	Low-number: underfitting. Every new instance being classified to its closest one not the majority of neighbors High-number: overfitting. Every new instance being classifies as the majority of all instances which might include several subclasses	1 weka.classifiers.lazy.IBk	a positive integer, typically small
-W	Window Size	Maximum number of training instances	The addition of new instances above this value will result in old instances being removed Smaller batch size might be effective on concept drift or to speedup	0 (No limit window)	0-any
-A	search algorithm	The nearest neighbour search algorithm to use		"weka.core.neighboursearch.LinearNNSearch \"weka.core.EuclideanDistance -R first-last\""	
-R	Distance function	Euclidean Distance			
-I	Weight neighbours	Weight by the inverse of their distance		No distance weighted	use when k > 1
-F	Weight neighbours	Weight by 1 - their distance		No distance weighted	use when k > 1
-E		Minimize mean squared error rather than mean absolute error		when using -X option with numeric prediction	
-X		Select the number of nearest neighbours between 1 and the k value specified using hold-one-out evaluation on the training data			use when k > 1

(a)

***Random Forest*(RF)**					
Random Forest		Description	Effects	Regular setting	Range
-P	Bag size	Size of each bag, as a percentage of the training set size		100 (%)	1-100 (%)
-I		Number of iterations		100	<num>
-num-slots		Number of execution slots		1 (no parallelism)	<num>
-K	Attribute numbers	Number of attributes to randomly investigate		0	<1 = int(log_2(#predictors)+1
-M	minimum number of instances	Set minimum number of instances per leaf		1	
-V	minimum variance for split	Set minimum numeric class variance proportion of train variance for split		1e-3	
-S		Seed for random number generator		1	
-O		Calculate the out of bag error			
-store-out-of-bag-predictions		Whether to store out of bag predictions in internal evaluation object			
-output-out-of-bag-complexity-statistics		Whether to output complexity-based statistics when out-of-bag evaluation is performed			
-print		Print the individual classifiers in the output			
-depth		The maximum depth of the tree	0 for unlimited	0	
-N		Number of folds for backfitting		0 (no backfitting)	
-U		Allow unclassified instances			
-B		Break ties randomly when several attributes look equally good			
-output-debug-info		If set, classifier is run in debug mode and may output additional info to the console			
-do-not-check-capabilities		If set, classifier capabilities are not checked before classifier is built	use with caution		
-num-decimal-places		The number of decimal places for the output of numbers in the model		2	

(b)

Figure 3.4: Hyper-parameter configuration of kNN and Random Forest methods

and thus also serves as the measure of choice in our evaluation.

To evaluate the performance of models, we create a simplified area under the curve (AUC) scoring metric to reduce the complexity that enables us to produce an upper-bound against which to compare the deep-learning methods. Otherwise, the comparison of all optimal thresholds for every individual model and between models is NP-hard. In imbalanced data, such as anomaly detection, the accuracy is not the desired metric. Additionally, (true positive rate) TPR or (false positive rate) FPR are not informative independently because they have a reverse relation and pushing the threshold toward one will have a negative effect on the other metric. The solution is to find the optimal point that TRP and FPR produce; the highest point on the ROC curve, which in our setup produces the highest AUC. The ROC curve is used for finding the best threshold for a certain model. However,

Boosting				
ADABoostM1	Description	Effects	Regular setting	Range
-P	Percentage of weight mass to base training on	reduce to around 90 will speedup	100	
-S	Random number seed		1	
-I	Number of iterations		10	
-W	Base classifier	Options specific to classifier should be adjusted	weka.classifiers.trees.DecisionStump	
-Q	Use resampling for boosting			
-D	If set, classifier is run in debug mode and may output additional info to the console		No distance weighted	use when k > 1

(a)

Bagging (Bootstrap aggregating)				
Bagging	Description	Effects	Regular setting	Range
-P	Size of each bag, as a percentage of the training set size		100 (%)	
-S	Random number seed		1	
-num-slots	Number of execution slots		1 (no parallelism)	
-I	Number of iterations		10	
-W	base classifier		weka.classifiers.trees.REPTree	
-O	Calculate the out of bag error			
-print	Print the individual classifiers in the output			
-store-out-of-bag-predictions	Whether to store out of bag predictions in internal evaluation object			
-output-out-of-bag-complexity-statistics	Whether to output complexity-based statistics when out-of-bag evaluation is performed			
-represent-copies-using-weights	Represent copies of instances using weights rather than explicitly			

(b)

Stacking					
Stacking		Description	Effects	Regular setting	Range
-X	number of folds	Sets the number of cross-validation folds		10	
-M	scheme specification	name of meta classifier, followed by options	weka.classifiers.rules.Zero		
-S		Random number seed		1	
-B	classifier specification	Full class name of classifier to include, followed by scheme options. May be specified multiple times	weka.classifiers.rules.ZeroR		
-D	Weight neighbours	If set, classifier is run in debug mode and may output additional info to the console			

(c)

Naïve Bayes					
NaiveBayes		Description	Effects	Regular setting	Range
-K	Use kernel density estimator rather than normal distribution for numeric attributes			---	---
-D	Use supervised discretization to process numeric attributes			---	---
-O	Display model in old format			---	(good when there are many classes)
	Cost-sensitive evaluation				

(d)

Figure 3.5: Hyper-parameter configuration of (a)Boosting and (b)Bagging and (c)Stacking and (d)Naive Bayes

in the first section of our study that the goal is to switch between optimal models, the best solution was to use a simplified version of the ROC curve that contains only the maximum performance of each model. Using this approach for selecting the optimal model becomes practical, especially in near real-time model selection and configuration.

For each model, we find the maximum AUC, which is the optimal point of TPR and FPR, and we compare this metric over various variations of datasets (e.g. different drifts and anomaly amplitudes). This technique provides a baseline from which the observation of the behavior of each model and finding the best performing ones in concepts drift becomes feasible.

In order to test the maximum achievable AUC of the model and the model robustness to the changes in the intensity of concept drift, we use the following terms:

Bayes Net					
BayesNet		Description	Effects	Regular setting	Range
-D	Do not use ADTree data structure	Do not use ADTree data structure		False	T / F
-B <BIF file>	BIF file to compare with	BIF file to compare with		False	T / F
-Q	Search algorithm	Search algorithm	weka.classifiers.bayes.net.search.SearchAlgorithm	weka.classifiers.bayes.net.search.local.K2	---
-E	Estimator algorithm	Estimator algorithm		weka.classifiers.bayes.net.estimate.SimpleEstimator	---

(a)

Decision Tree (C4.5)			
J48		Regular setting	Range
-U	Use unpruned tree	False	T / F
-O	Do not collapse tree	False	T / F
-C	Set confidence threshold for pruning	0.25	Any positive double but Computationally expensive (Time exponentially) (C200 take ~15min A:84.38%) (C0.25 take ~9sec A:84.50%)
-M	Set minimum number of instances per leaf	2	(M20 take ~15sec A:81.25%)
-R	Use reduced error pruning	False	T / F
-N	Set number of folds for reduced error pruning. One fold is used as pruning set.	3	V > 0 (N10 take ~5min A: 81.25%)
-B	Use binary splits only	False	
-S	Don't perform subtree raising	True	T / F
-L	Do not clean up after the tree has been built	False	T / F
-A	Laplace smoothing for predicted probabilities	False	T / F
-J	Do not use MDL correction for info gain on numeric attributes	False	T / F
-Q <seed>	Seed for random data shuffling	1	Any positive
-doNotMakeSplitPointActualValue	Do not make split point actual value	False	T / F

(b)

Linear Regression			
LinearRegression		Regular setting	Range
-S	Set the attribute selection method to use. 1 = None, 2 = Greedy	0 = M5' method	
-C	Do not try to eliminate collinear attributes	True	T / F
-R	Set ridge parameter	default 1.0e-8	Very small number
-minimal	Conserve memory, don't keep dataset header and means/stdevs. Model cannot be printed out if this option is enabled	keep data	---
-additional-stats	Output additional statistics	---	---
-output-debug-info	If set, classifier is run in debug mode and may output additional info to the console	---	---
-do-not-check-capabilities	If set, classifier capabilities are not checked before classifier is built	---	---

(c)

Figure 3.6: Hyper-parameter configuration of (a) Bayes Net and (b)Decision Tree and (c)Linear Regression and (d)Neural Networks

- *Maximum AUC*: the highest AUC score a model can achieve when tested over changes of domains or data characteristics.
- *Accumulated AUC*: the summation of the overall area under the curve of the AUC scores when the concept drift intensity increasingly increased.

3.5 Anomaly detection in network traffic data

This section filed as a disclosure claims priority to U.S. Provisional Patent Application No. 62/683,889, filed June 12, 2018, and a utility patent filed Jun 1, 2019 entitled “Pattern detection in time-series data,” the contents of which are incorporated by reference. The second part of the section filed as a disclosure claims priority to U.S. Provisional Patent Application entitled "Unsupervised outlier detection in time-series data" filed Aug 13, 2019. This section generally studies the implementation of proposed methods in real-world high speed backbone network performance monitoring.

In this thesis, we used Layer 3 data. In particular, one of the benchmark test-set data

Neural Network (MLP)				
MultilayerPerceptron		Description	Regular setting	Range
-L	Learning Rate for the backpropagation algorithm		0.3	0 - 1
-M	Momentum Rate for the backpropagation algorithm		0.2	0 - 1
-N	Number of epochs to train through		500	
-V	Percentage size of validation set to use to terminate training		0	0 - 100
-S	The value used to seed the random number generator		0	0 >= V > Long
-E	The consecutive number of errors allowed for validation testing before the network terminates		20	V > 0
-G	GUI will be opened		---	---
-A	Autocreation of the network connections will NOT be done		Works with-G	---
-B	A NominalToBinary filter will NOT automatically be used		---	---
-H	The hidden layers to be created for the network	mean value of input and output layers	a	comma separated 'a' = (attribs + classes) / 2 'i' = attribs 'o' = classes 't' = attribs .+ classes
-C	Normalizing a numeric class will NOT be done		---	not normalize if class is numeric
-I	Normalizing the attributes will NOT be done		---	not normalize the attributes
-R	Resetting the network will NOT be allowed		---	Set not allow
-D	Learning rate decay will occur.		---	---

(a)

Support Vector Machine (SVM)				
SMO		Default	Range	Often change?
-no-checks		checks on	Off - assumes data is purely numeric, no missing values, with a nominal class, and no instance with 0 weight	No (use with caution!)
-C	The complexity constant	1	High = overfit Small = underfit	Yes 1.0
-N	Normalize/ standardize	0=normalize	0=normalize/1=standardize/2=n either	0
-L	The tolerance parameter	1.0e-3		0.001
-P	The epsilon for round-off error	1.0e-12	Optimal generalization	Sometimes 1.0e-12
-M	Fit calibration models to SVM outputs	False	T / F	No
-V	The number of folds for the internal cross-validation	-1, use training data	---	-1
-W	The random number seed	1	---	1
-K	The Kernel to use	weka.classifiers.functions.supportVector.PolyKernel	---	No
-calibrator	Full name of calibration model, followed by options	weka.classifiers.functions.Logistic	---	No
-output-debug-info	If set, classifier is run in debug mode and may output additional info to the console	False	T / F	No
-do-not-check-capabilities	If set, classifier capabilities are not checked before classifier is built	False	T / F	No (use with caution)
-num-decimal-places	The number of decimal places for the output of numbers in the model	2	V > 0 (2)	No
SMO - Options specific to kernel weka.classifiers.functions.supportVector.PolyKernel:				
-E	The Exponent to use	1.0	0 for full cache and -1 to turn it off	Advanced - No
-L	Use lower-order terms	False - no	T / F	No
-C	The size of the cache (a prime number)	250007	0 for full cache and -1 to turn it off	Advanced - No
-output-debug-info	Enables debugging output	False - off	T / F	No
-no-checks	Turns off all checks	False - checks on	T / F	No (use with caution!)
SMO - Options specific to calibrator weka.classifiers.functions.Logistic:				
-C	Use conjugate gradient descent rather than BFGS updates	False	T / F	No
-R	Set the ridge in the log-likelihood	False	T / F	No
-M	Set the maximum number of iterations	-1, until convergence	---	No
-output-debug-info	output additional info	False	---	No
-do-not-check-capabilities	capabilities are not checked before classifier is built	False	---	No (use with caution!)
-num-decimal-places	number of decimal	2	V > 0 (2)	No

(b)

Figure 3.7: Hyper-parameter configuration of (a)Neural Network and (b)Support Vector Machine

consists of real-world anomalous patterns collected by a telecommunications company that specializes in networking equipment and software. It is a service supplier of mainly layer 2 and 3 fiberoptic switches. The main source for producing the network traffic is the Network layer (Layer 3) which is the third layer of the seven-layer organization in an Open Systems Interconnection (OSI) model of computer networking for data transfers on a network. We did not use IP addresses or other network specific attributes for training our models. Therefore, the proposed transformation and optimization methods evaluated on patterns in various types of generated sequential data might be beneficial in other anomaly detection problems outside the computer networking domain.

The method includes obtaining data in a time-series and creating one-dimensional or multi-dimensional windows from the time-series data. The one-dimensional or multi-dimensional windows are created either independently or jointly with the time-series. The

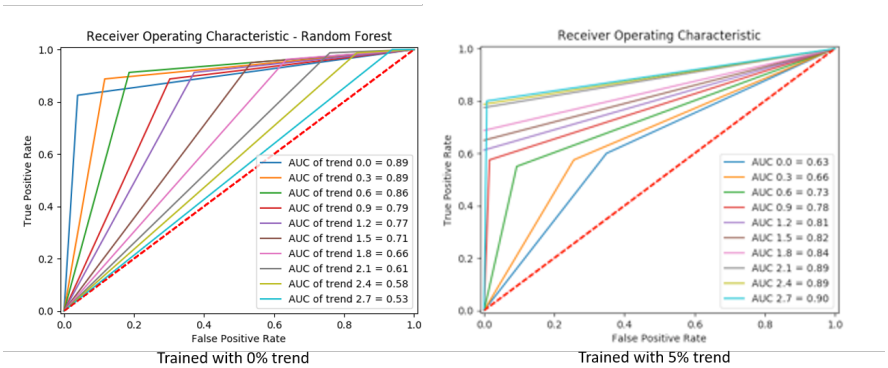


Figure 3.8: Examples of two models trained with no trends in the data and tested against various datasets with different trends. (left) As the data characteristic moves further away from the one in the training set, the performance decreases. (right) As the data characteristic becomes closer to the one in the training set, the performance increases.

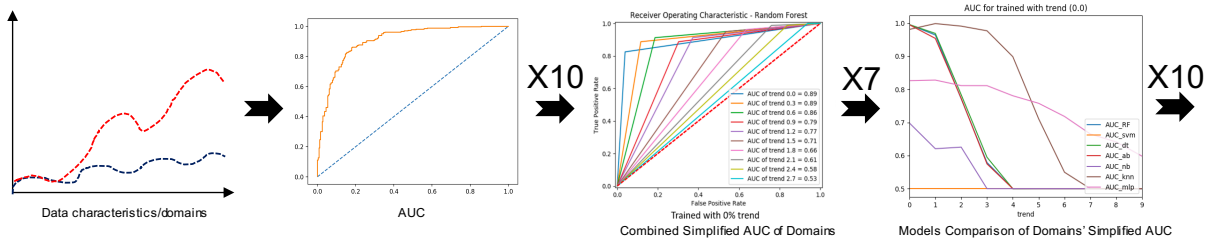


Figure 3.9: The steps of evaluation for comparing several models in parallel when the data concept drifts.

method also includes training a deep neural network with the one-dimensional or multi-dimensional windows utilizing historical and/or simulated data to provide a neural network model. The method further includes processing ongoing data from a network with the neural network model to detect one or more patterns of a particular category in the ongoing data and localizing the one or more patterns in time.

According to another implementation, a non-transitory computer-readable medium configured to store a program executable by a processing system is provided. The program includes instructions to cause the processing system to obtain time-series data and create one-dimensional windows from the time-series data. The program also causes the processing system to train and optimize hyper-parameters of one or more machine learning algorithms with the one-dimensional windows obtained from historical data to create one or more machine learning models. Also, the program causes the processing system to determine an algorithm among the one or more machine learning algorithms with the best performance. The program further causes the processing system to utilize the machine learning model created from the algorithm determined to have the best performance to classify future windows as containing a pattern of a particular category and localize the pattern in time in ongoing data.

Fig 3.11 is a graph of time-series data of network traffic volume shown over time. The graph of the network traffic volume also illustrates examples of anomalies in the

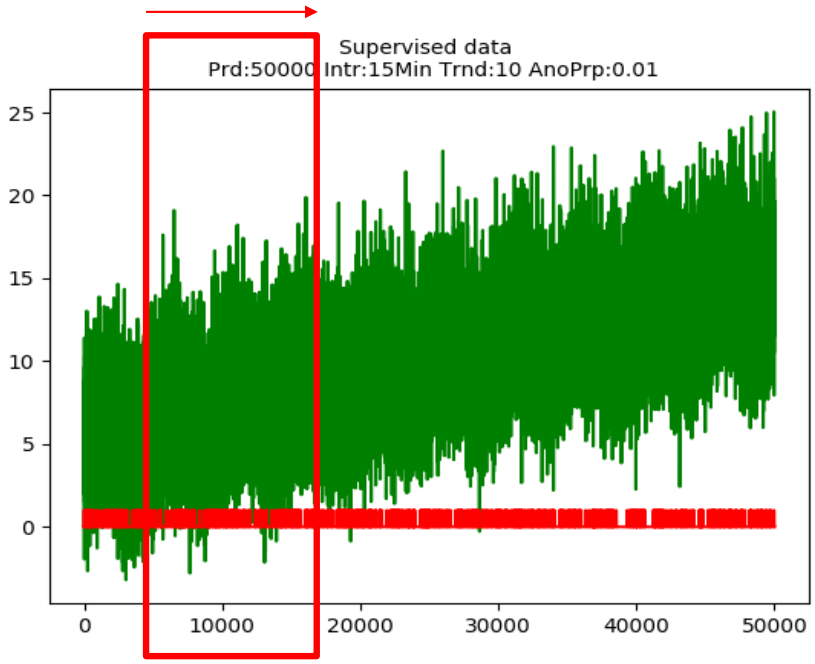


Figure 3.10: Generated data and sliding window, the data characteristics include trend, noise and seasonality then spikes are added as anomalies

data. Pattern detection is trained with historical data and anomalies can be identified and labeled. For example, windows 12 are labeled with "Y" to indicate the existence of an anomaly and windows 14 are labeled "N" to indicate an absence (or non-existence) of an anomaly. Multiple anomaly types can be encoded by using more than a binary classifier of "Y" and "N." In some cases, multiple anomaly types can be detected in the same windows 12, 14 to indicate other types of anomalies or other patterns.

Pattern detection in a time-series can be used for networking applications, telecommunications, as well as many other applications. For example, in the field of networking applications, pattern detection can be used in the following use cases: for forecasting threshold crossings, for forecasting alarms, for forecasting quality-of-experience (QoE), for network anomaly detection, among others. Pattern detection can also be used in other areas (e.g., forecasting engine failure or tire deflation in cars from engine- or tire-collected information, forecasting bridge failure by detecting patterns in a time-series associated with bridge sensors, detecting earthquakes or tsunamis by detecting patterns in seismological time-series data, recognizing that a person is having a heart-attack from heart rate measurements collected by a smart watch, forecasting traffic congestion on streets by detecting patterns in a time-series from video cameras on streets, cars, or traffic detection sensors, etc.).

Time-series data can also be one-dimensional or multi-dimensional. For example, multiple sensors can provide data at about the same time, whereby this sensor data can be stacked together to provide a time-series that has multiple types of measurements associated with each time point. The patterns described here are detected across this potentially multi-dimensional time-series.

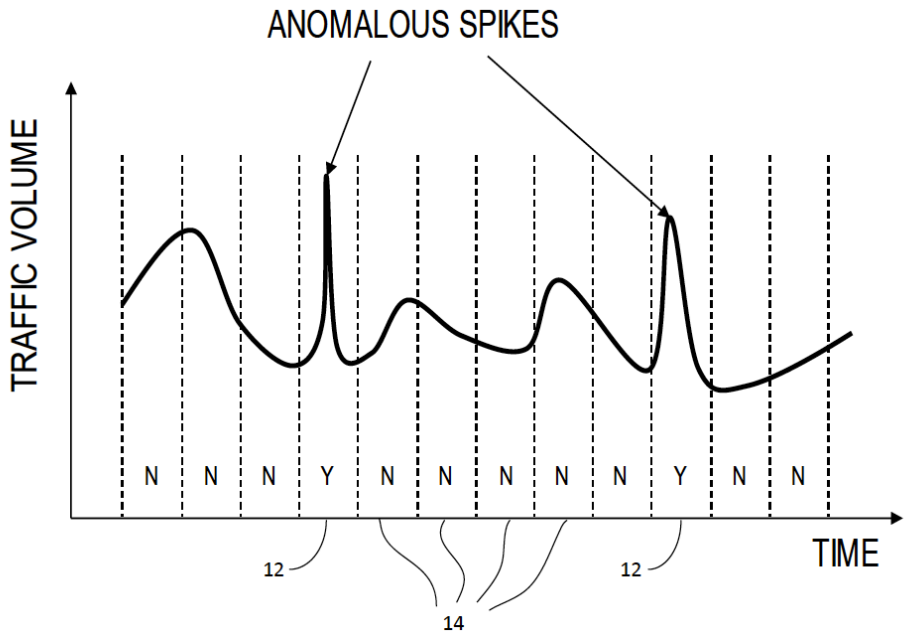


Figure 3.11: A graph of traffic volume plotted over time illustrating example anomalies.

Pattern detection has two distinct life stages. The first life stage includes (a) the training of the underlying machine learning algorithm and (b) in the case of classical approaches, optimization of the hyper-parameters. The second life stage is the real-time, online use of the algorithm for pattern detection applied on new data.

In particular, the systems and methods of the implementation includes classical machine learning algorithms (C4.5, regression trees, Bayesian nets, etc.) and deep neural networks, such as Convolutional Neural Networks (CNN), to detect patterns in time-series. Based on testing, it has been determined that CNN-based pattern detection is much simpler and quicker to train and has a better detection performance than the classical approaches. It is proposed that recurrent neural networks (RNNs) be used on time-series due to their ability to hold past values, despite the fact that CNNs have a much larger capacity (and therefore better performance) and has the ability with the Regional Convolution Neural Network (R-CNN) approach to detect multiple co-existing patterns.

Again, examples of use cases in networking applications may include forecasting threshold crossings, forecasting alarms, forecasting quality-of-experience (QoE), network anomaly detection, among others. Threshold crossing forecasting may be used to solve problems in the context of adaptive modulation technologies in optical networking, which allow an increase in bandwidth if there is sufficient Signal-to-Noise Ratio (SNR) available at the receiver. For example, an operator needs to be confident that increasing the rate will not result in an outage sometime in the future, due to SNR dropping below a Forward Error Correction (FEC) limit for the higher rate modulation. During training, pattern detection for threshold crossing forecasting examines historical time-series (e.g., of SNRs) to discover patterns during a time interval, associated with values of the time-series dropping below the threshold at a later time. If there is a correlation between measurements and subsequent threshold crossings, machine learning may be used to discover this correlation

and associate the correlation with a pattern. During online usage of new data, pattern detection functions include examining the time-series to find the previously discovered patterns. If a pattern associated with threshold crossing is not found with high confidence, the threshold crossings will not be detected in the future.

As a contrived example, a pattern may include a downward slope of 0.1dB/week that results in the value of the SNR dropping 2.0dB over a period of next 20 weeks, which would be below a prescribed threshold. While the threshold crossing forecast in this example can be solved with linear regression, the power of using machine learning is its ability to (1) discover other unknown patterns and (2) generalize to more complicated patterns than a simple straight line.

Fig 3.12 is a graph 20 of time-series data where Signal-to-Noise Ratio (SNR) measurements are taken over time. A pattern detection model that is modeled from the historical training data can be used with new data for predicting when the SNR curve crosses over a threshold 22. Using the pattern detection model, new data can be plotted, and patterns may be detected to predict when the SNR in the future may cross the threshold 22. Pattern detection includes analyzing an upward slope pattern 24 or other curve characteristic to predict a future result 26 of a threshold crossing.

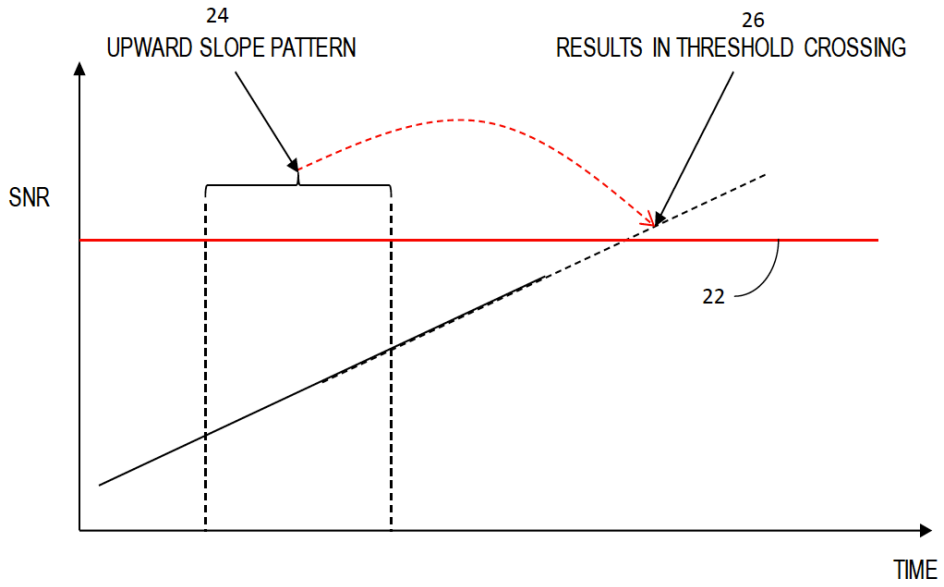


Figure 3.12: A graph for predicting threshold crossings with pattern detection.

Alarm forecasting can be used to give an advanced warning that an event that would result in an alarm is going to happen. This use case enables proactive network maintenance, which can be particularly useful for operators. During training, pattern detection for an alarm forecast examines the time-series of a network measured performance indicator to discover patterns that are associated with future alarms. If there is a correlation between performance indicators and subsequent alarms, pattern detection using machine learning is configured to discover it. During the online phase, pattern detection finds the patterns associated with the failure, which can be used to notify the network operator which equipment to service pro-actively.

Fig 3.13 is another graph 30 of traffic volume (e.g., in a network) over time. The data may be analyzed with pattern detection for predicting congestion events 32 (e.g., when traffic volume exceeds a threshold for an extended length of time). Pattern detection is trained with traffic measurements (or CPU utilization measurements) and labeled on graph 30 as patterns 34 that represent a "start of busy period," which may be indicative of or may result in congestion 32 in the future. One set of data (e.g., queue sizes) can be used for measurements, while another (e.g., end-to-end performance) can be used to generate labels (e.g., "congestion" or "no congestion"). Patterns can then be further correlated with the network at the time for root cause analysis. Congestion 32 can be periods of time when packets are dropped or latency increases beyond a bound. In a virtualized network setting (e.g., 5G), CPU utilization may be a greater indicator of congestion 32 than packet queues. Traffic other than packet data can be used to detect congestion 32 such as video re-buffering events at a player device (e.g., User Equipment (UE)).

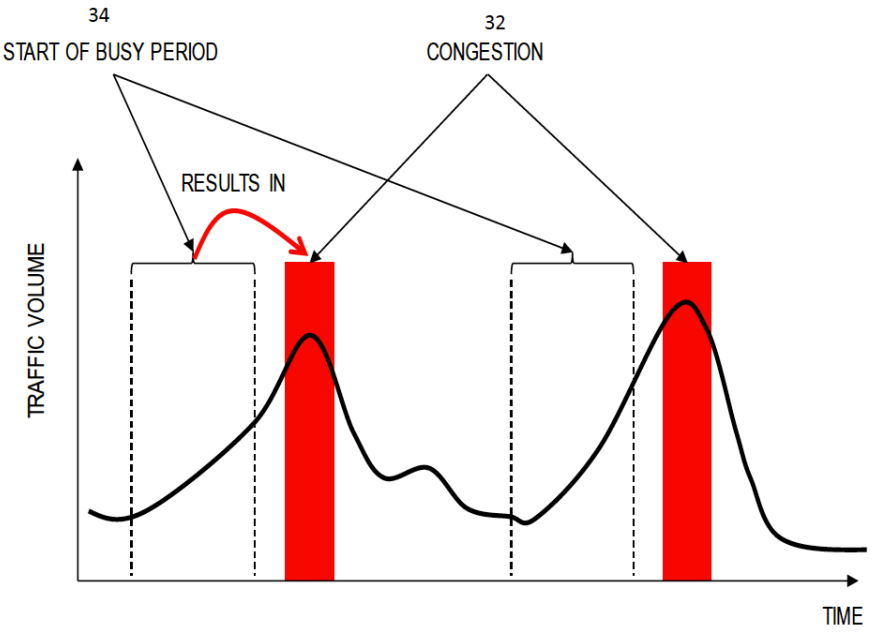


Figure 3.13: A graph for predicting congestion with pattern detection.

A special case of alarm forecasting is if an alarm is triggered due to a threshold crossing, which could be accomplished by using a threshold forecast (see above). However, the advantage of this more general approach is that it is not dependent on the simple well-known causes of alarms and can therefore discover more complex non-obvious network patterns that result in alarms. As an example, the alarm may indicate a Loss of Signal (LOS), which is due to equipment failure. During training, pattern detection uses historical network measurements to discover patterns associated with future loss of signal alarms. During the online phase, pattern detection searches incoming network performance measurements for the previously found patterns and notifies the user if one is found.

Fig 3.14 is a graph 40 of performance monitoring (PM) and associated alarms over time. The data of graph 40 may be used for predicting alarms before they happen. Pattern detection may be trained with traffic measurements and labeled as patterns (e.g., windows

A1, labeled 42, followed by windows A2, labeled 44). These changes 46 (e.g., from window A1 to window A2) in PM activity may be analyzed in pattern detection analysis to predict a start of congestion in the future, corresponding to alarm A3, which may be a critical alarm 48. One set of data (e.g., queue sizes) can be used for measurements, while another (e.g., end-to-end performance) can be used to generate labels. Patterns can then be further correlated with the network at the time for root cause analysis.

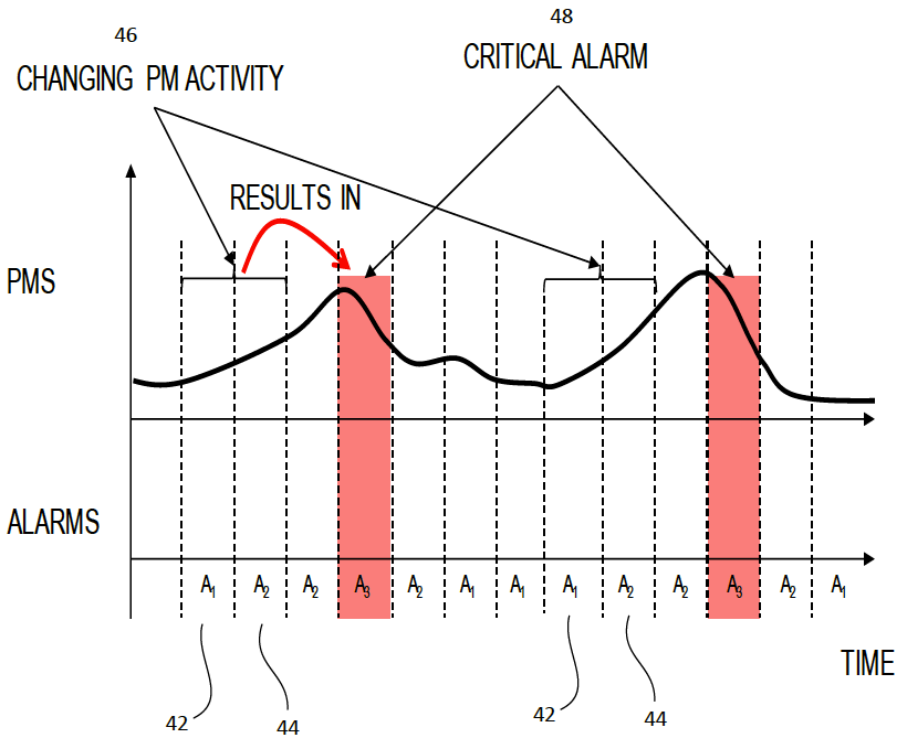


Figure 3.14: A graph for predicting critical alarm conditions with pattern detection.

One way to detect congestion in a network is by observing users' quality-of-experience (QoE). For example, network congestion may result in re-buffering events at a User Equipment (UE) video player. Pattern detection can be used to give advanced warning when the traffic exceeds network capacity, by associating network traffic measurements with bad quality-of-experience. During training, pattern detection discovers the network pattern of one or more characteristics or parameters (e.g., buffer status, traffic load, etc.) associated with subsequent video player re-buffering events. During the online phase, pattern detection finds the pattern and this knowledge can be used to forecast video player re-buffering. The advanced warning can be used to change the network configuration, so that congestion is avoided, such as by invoking higher rates with liquid spectrum, re-routing traffic, changing overbooking parameters, among other actions.

Examples of anomaly detection may include drops in SNR due to thunder strikes, detection of traffic pattern shifts (from packet counter data and call admission control data), network intrusion detection (from an examination of packet counter data), equipment failure prediction (from performance monitoring data), etc. Pattern detection for anomaly detection associates labeled anomaly periods with the anomalous measurements in the

time-series. During the training phase, pattern detection learns the patterns of anomalies, which it can use later during the online phase. The foregoing description assumes anomaly detection as a primary embodiment for developing pattern detection on time-series. However, other use cases, not limited to the ones mentioned herein, are also contemplated.

In our studied domain, we describe an early pattern detection when 10 to 20 percent of the footprint of the anomalous pattern has appeared; a mid-pattern detection when 45 to 65 percent is detected; and a full-pattern detection when the whole pattern is detected. The results of an experiment of various stages of pattern detection is illustrated in the following Table. We generated full patterns and then labeled the first 15% (early) and 50% (middle) of the length and the 100% (full) of the patterns and evaluate our models.

In addition to the use of pattern detection techniques for detecting patterns in the field of networks and telecommunications, the pattern detection techniques described in the implementation may also be used in multiple other fields as well. For example, a heart monitor (e.g., a wearable wristband or other suitable monitoring device) may monitor the heart rate of a person over time. Historically, certain patterns in the heart rate may be representative of an imminent heart attack. In this case, an alarm can be sent to the user or to medical personnel so that preventative measures can be taken to prevent or treat the person's heart condition in a timely manner.

In the field of monitoring vehicular traffic, patterns may be detected in the roadways to identify problem areas. For example, time-series data from previous trips may be used to detect pot holes or other undesirable road conditions at certain points along the roads, and then using the obtained time-series information to warn the driver or take evasive self-driving maneuvers to avoid the problem spots. Also, blind areas may be detected to alert the driver to use caution at these areas. Vehicular data may also be used for measuring lanes of traffic or other patterns.

In the field of finances, the pattern detection techniques of the implementation uses time-series data to determine spending patterns of a person. If credit card activity is detected as an anomaly with respect to the person's regular spending patterns, alerts can be provided to further monitor whether or not current purchases are authorized. A known spending pattern associated with suspicious activity such as a set of suspicious purchases (a spending signature) can be used as for training a machine learning model to recognize these suspicious patterns in customer data. These and other fields of technology may benefit from the machine learning methods for training neural network models described in the present implementation and utilizing these models with current (online) time-series data for detecting patterns and anomalies.

Conventionally, performance monitoring, problem detection, and root cause analysis are performed in a manual fashion after a failure has occurred. This approach is taken across various application areas, such as manufacturing, vehicle maintenance, airplane maintenance, healthcare, building maintenance, road and other infrastructure maintenance. This manual approach is very expensive, time-consuming and requires a human expert with the knowledge of the given system to debug the problem after the failure. At the same time, the number of monitors is increasing, as the Internet of Things (IoT) is now connecting things to the network, which would not conventionally be connected or monitored. The manual approach to performance monitoring with the failure and debug cycle is not feasible. At the same time, it would be desirable to decrease the cost even in current manual

approaches by introducing machine learning methodologies for pattern detection to enable new approaches to detecting and forecasting faults before they occur and to find patterns in time-series that can be used to pin point the causes of failures.

As an example, network performance monitoring is described, but the approaches provided here can be applied to any of the areas mentioned above. Conventionally, problem detection (i.e., anomaly detection) in networks is implemented after a failure has occurred. Specifically, following a failure in a network, an operator or technician would log into the system, perform a manual investigation, and provide remediation. Of course, this approach is reactive and typically involves a traffic hit, traffic loss, protection switching, etc., followed by network maintenance. Another approach to anomaly detection is to re-implement the failure scenario via a piece of software that can run and analyze the scenario in an offline manner. For a handful of Performance Monitoring (PM) metrics relating to the problem, alarms would be raised if any given PM crosses some pre-defined threshold. This is typically achieved using a rule-based engine with hard-coded "if... then... else..." statements specified by a human expert.

Disadvantageously, with these conventional approaches, the reaction time is slow, engineering time is expensive, and experts are rare. Also, this approach only finds known failures that are also easy to specify. The approach presumes that the human expert is able to articulate the specific reason for a network failure and that this network failure happens due to the threshold crossing at one point. The approaches cannot and are not used to finding failures that span multiple network elements, links, etc. Further, these approaches do not scale with large and complex networks. Also, these conventional approaches require a lot of expertise, work, and time to implement. Further, defining and updating complex "if... then... else..." rules is complicated and time-consuming, and there is limited accuracy if limited to simple rules, such as one-dimensional thresholding.

Conventional approaches using PM metrics focused on trends from individual PM metrics, such as simple linear fits and relying on subject matter experts to interpret the values of the trends. Of course, these conventional approaches do not use all available information, result in lower accuracy, and require expertise to interpret trend values.

Current approaches in pattern detection are limited to finding objects in images, recognizing letters, speech-to-text conversion, text or speech translation, etc. Pattern recognition in audio has some similarities to network applications, but these approaches only ever use Recurrent Neural Networks (RNNs). The vast majority of currently published network anomaly detection algorithms are not based on machine learning. Typically, these approaches use Principal Component Analysis (PCA), or its derivatives, to find outliers in multi-dimensional data. As shown by a large body of previous literature, this approach does not work with typical time-series data since the data is not stationary and the distribution at each time sample is not normally distributed.

The typical use cases in networking include forecasting threshold crossing of Performance Monitoring (PM) data, forecasting alarms, forecasting Quality-of-Experience (QoE), anomaly detection, etc. Conventionally, these use cases are addressed with regression techniques. Regression techniques are the classical "forecasting" algorithms. Forecasting algorithms require a high touch approach where an expert in the use of these algorithms is able to choose the approach best suited for the forecasting, based on their observations about the time-series. Another problem with the regression approaches is their low ca-

pacity. Capacity is informally defined as the ability of the algorithm to fit a wide variety of functions. For example, linear regression has a low capacity as it cannot fit a highly varying time-series. Also, a higher order polynomial regression will typically overfit the time-series due to its low ability to generalize.

A variety of data sources can be employed to obtain information about every component of the network, from the physical (or virtual) devices, to the communication channels, the usage patterns, the environment, and the business context. Network devices (e.g., network elements) generate Performance Monitoring (PM) information, alarms, and/or logging data. These include things like power levels, error counters, received, transmitted or dropped packets, Central Processing Unit (CPU) utilization, geo-coordinates, threshold cross, etc. Communication channels (or "services") also generate PM data, for all layers of the Open Systems Interconnection (OSI) model (ISO/IEC standard 7498-1, 1994). For instance, layer-3 network performance is characterized by bandwidth, throughput, latency, jitter, and error rate. Data from end-users, from the environment, or from businesses that typically comes from third-party databases.

Each time any of the above data is collected, it is useful to record a timestamp associated with it. Time is unique in that it can be used to correlate independent data sources. For instance, data from different sources can be associated if they were all taken during the same time interval, to define a "snapshot." Furthermore, sorting data in chronological order is frequently used to measure time-series trends to anticipate future events.

Most communication networks connect to a plurality of device types. Also, different types of devices from different equipment vendors tend to produce different data in different formats. Hence, communication networks are said to generate a wide variety of data. In addition, the frequency at which the above data is collected (a.k.a. Velocity) can vary for each source. Likewise, the amount of time during which the data is kept in storage can also vary. When networks contain a large number of devices and services, with high-frequency data-collection and/or long storage periods, the result is large data volumes. The combined Variety, Velocity, and Volume is often referred as "Big Data."

Equipped with sufficient infrastructure, a common approach is to collect and store all available data and enable ad-hoc analysis after the fact (i.e., in a reactive manner). When this is not possible, tradeoffs have to be made to only pick the most relevant data for the targeted application(s). For example, an optical networking effect was explained more accurately when using additional inputs such as weather data (see D. Charlton et al., "Field measurements of SOP transients in OPGW, with time and location correlation to lightning strikes", Optics Express, Vol. 25, No.9, May 2017). However, with the systems and methods described herein, wider variety, larger velocity, and larger volumes of data will broaden the coverage and increase the accuracy of ML-driven applications.

3.6 Proposed Methods for Sequence to Image Transformation

In one dimensional CNNs we can input time-series data, speech signals, sentences or documents represented as a vector from a matrix. Each row of the matrix corresponds to one

window or a token, typically a certain period of time. For low-dimensional representations we use abstractions such as Fourier transformed signals. Although the precision of 1D-CNN, similar to conventional machine learning models, is limited to detect if a slice of data contains anomalous instances or not, but would not be able to explicitly indicate which instance is resulting in an anomaly. We propose a 2D-CNNs that similar to image detection problems can locate inside a window which instances are anomalous. We create a data transformation than enables 2D-CNNs to deal with time dependency of sequential data.

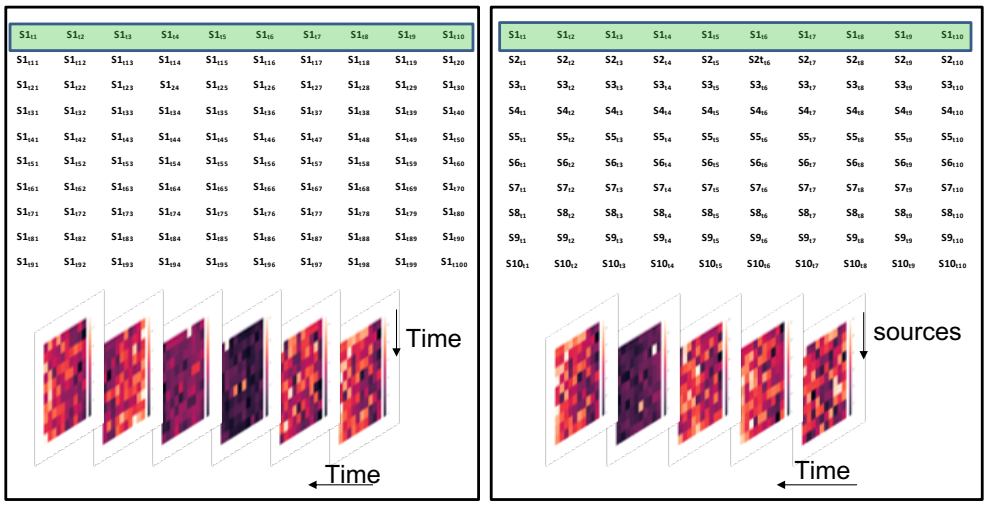


Figure 3.15: (Left) The transformation method for univariate analysis. The x-axis (columns) illustrates samples in time and y-axis (rows) shows slices of data that can match repetitive patters in the data for optimal pattern detection. (Right) In univariate analysis scenario, y-axis includes data from various sources.

Classifications such as supervised anomaly detection, pattern analysis or state categorization can be performed using CNNs, but require positional features for the input as the pooling operations lose information about the localization of the objects. In this case, the input layer can be raw data, embeddings or abstracted data. CNNs have static and dynamic channels that adjust during training, and convolutional layer with multiple filters and a max-pooling layer and a classifier. However time-series data requires special preprocessing steps to become compatible with 2D-CNNs.

We introduce a method that transforms sequential data into an image representation while keeping the notion of time. Using this method with a combination of pixel-wise input model, the location, shape or drift present in the data has no effect on the detection performance. First, a preprocessing block transfers the sequential data in raw format from one or several sensors to a 2D heat-map representation. Second, an optional frequency transformation box applies a Fourier transformation (see Fig 3.17).

Suppose that $x = [x_0, \ldots, x_{N-1}]$ is an N dimensional time-series vector. We define a sliding window with size $\pm t$, as well as step size s which describes strides and the number of rows r indicating the size of two dimensional representation. The final image sizes are $2t \times r$ and the total number of images in the tensor is $\frac{x}{(2t \times r)}$ while there is no overlap

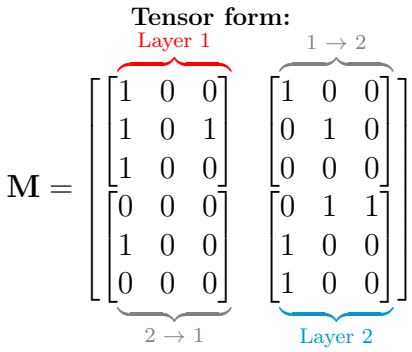


Figure 3.16: The data representation transforms to tensors of various channels of the images.

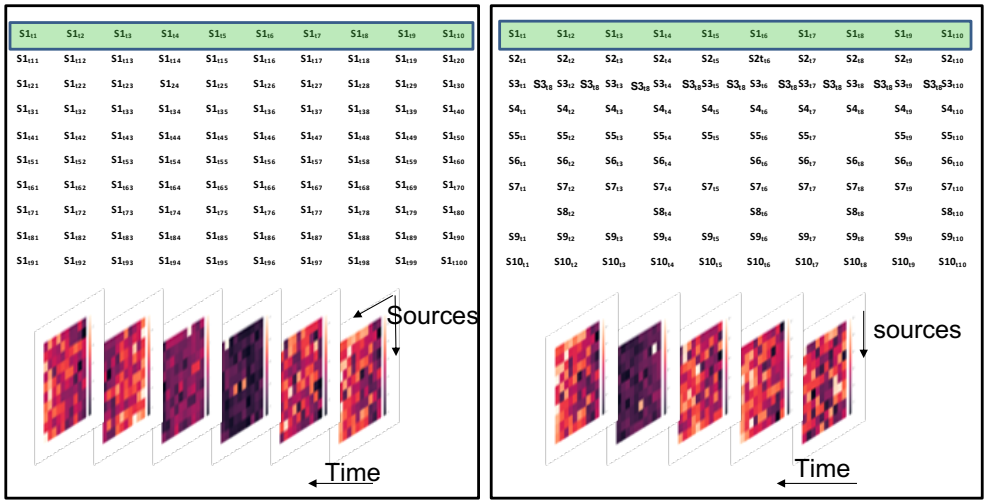


Figure 3.17: (Left) The data can be also transformed to snapshots of all sensors and data generating sources. Both x-axis and y-axis are one sample from each source. A model can be trained to detect relational patters between data sources in every timestamp. (Right)The pixel-wise transformation provides robustness to missing data points and enables the machine learning model to learn data with various sampling rate ratios.

in the slides. In case of strides not equal to the window size, (1) is the total number of images in tensors. The main advantage of the windowing approach combined with the 2D CNN is the ability to localize the pattern in time. The preprocessing block transfers the time-series data in raw format from one or several sensors to a 2D representation. The second (optional) box applies a Fourier transform to the data, and a 2D CNN learns the normal and anomalous data. A feature map creates rectangular boundaries that surrounds anomalies, and the last block is an adjustable threshold that defines the boundary between normal and abnormal classifications.

$$\text{Total number of images} = 1 + \frac{x - (2t + ((r-1) \times s))}{r \times s} \tag{3.5}$$

The key invention that enables pattern detection in network time-series is the use of

CNN for specific way we window the data to obtain 2-dimensional representation. A time window passed over the time-series with a lag, obtaining a series of horizontal vectors that are grouped and stacked to obtain matrices. A matrix representing an image is obtained for every lag, resulting in a series of matrices, which we refer to as images. We create heat-map from time-series data (see Figure 3.17.a), and in another type we applied Fourier transform. Let m be an integer and let $\omega = \exp(\frac{-2\pi i}{N})$. The Fourier transform is given by $\mathcal{R}\left\{\frac{1}{2\pi}\int_0^\infty 2F(\omega)\mathrm{e}^{i\omega t}\,d\omega\right\}$, as illustrated in Figure 6.1.b. The approach is also possible for multi-dimensional tensors for the case where multiple time-series are analysed jointly. Our synthetic raw data and transformed images for benchmarking can be obtained from our repository [94]

Two-dimensional CNNs have a serious downside: their requirement for a fixed size input. To overcome this issue, we introduce a method that transforms sequential data into an image representation. When using a pixel-wise input model, the location, shape or drift present in the data has no effect on the detection performance. Let $x = [x_0, \ldots, x_{N-1}]$ be an N dimensional sequential vector. We define a sliding window of size $\pm t$, a step size s, which describes the strides; and a number of rows r, which indicates the size of the 2D representation. The final image sizes are, thus, $2t \times r$ and the total number of images in the tensor is $\frac{x}{(2t \times r)}$ assuming there is no overlap in the slides.

3.7 Proposed Deep Learning Methods

In the second part of our study, we investigate the performance of deep learning methods, and compare their generalization capability in the presence of concept drift, particularly we focus on the consistency rather than the sub-optimality of ANNs as well as the demonstrated performance of their recent deep learning extensions and set to find out whether Deep Learning approaches applied to data streams subject to concept drifts are superior to conventional methods while also demonstrating the same kind of consistency as ANNs. In particular, Convolutional Neural Networks (CNNs) have shown themselves to be a superior approach over traditional algorithms in various contexts. The abstraction power and the location invariance capabilities offered by CNNs, make them a suitable model in our context as well. Notably, CNNs produce high-level features by automatically learning new abstraction layers which allows them to outperform conventional ANNs in a broad range of problems [28] [73] [77] [125].

From our preliminary experiment we found that one-dimensional CNNs (1D-CNNs) provide a very stable AUC performance when data characteristics are subject to concept drift. However, their detection precision is limited to entire labeled sequences and cannot pinpoint the anomalous instances. For instance, in cybersecurity each network trace is labeled as either normal or attack. There is a chain of warnings and patterns of anomalous instances that led to a complete attack. 1D-CNNs are only able to detect attacks after the network trace is complete, and they are unable to pinpoint the position of the warnings that resulted in the final outcome. A solution to this problem could consist of sliding the trace into small time cuts using a sliding windows and labeling them. However, this solution could still fail despite the quality of labeling as longer attack footprints can go undetected. Therefore, if a chain of warnings that is considered to lead to a failure falls into

different slides, the system would not be able to detect the attacks. Moreover, high-quality labeling in many real world applications can not not be provided for every individual instance. We therefore require a representation that provides a larger view of the data without removing the important positional information of the events while precisely detects anomalous instances. This problem is what led us to consider 2D-CNNs. In this section, our aim is to provide results of the 2D-CNN and proposed image based methods for sequential data, exploiting their modeling performance to improve the resulting anomaly localization performance.

To detect patterns in a time-series, historical data or training data from the time-series are used and labels associated with time periods are created. There may be several different labels corresponding to different patterns. Historical data and labels are used to train one or more machine learning algorithms resulting in a model. Historical data is windowed and windows are associated with labels. Machine learning algorithms are trained with windows as exemplars and labels as what the output could be. The trained model is used for pattern detection, new data is windowed, and windows are given to the machine learning algorithms whose output is the label.

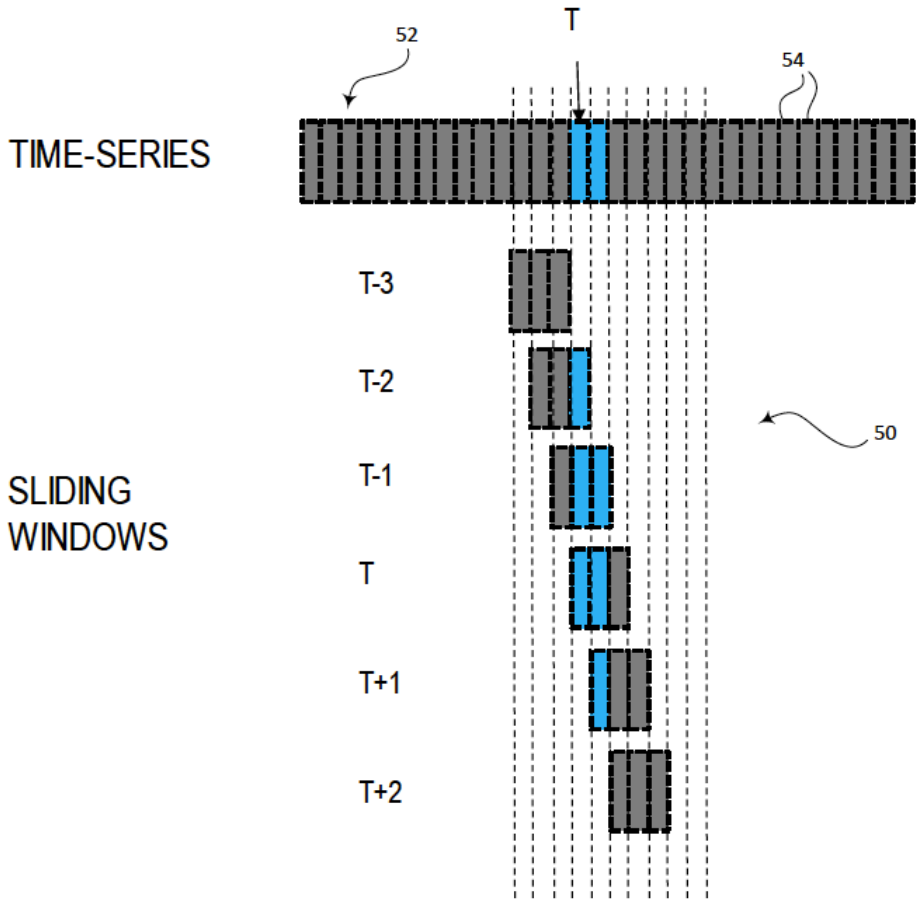


Figure 3.18: A diagram of a one-dimensional sliding (or moving) window, according to various embodiments.

First, the approach used in the implementation includes a "windowing" technique to create inputs for CNN and R-CNN deep neural networks. Conventional ML system do not use this present windowing technique, which utilizes deep neural networks (CNN, R-CNN) on data over a time-series. More specifically, the present systems and methods may include utilizing deep neural networks with a transformed time-series for pattern detection in time-series data. Second, the windowing approach allows localization of anomalies in time, whereby the present systems and methods perform localization to overcome conventional problems with pattern detection in time-series. Third, the present systems and methods use machine learning for pattern detection in time-series, which is a new application of this type of machine learning. Fourth, the windowing approach also works on one-dimensional windows using a classical approach and hyper-parameter optimization. Fifth, the approach can be used for pattern detection across multiple time-series, jointly. Sixth, pattern detection is provided for the use cases described herein, which were only ever addressed with regression forecasting techniques.

Fig 3.18 is a diagram of a one-dimensional (e.g., one variable) sliding window. Sliding windows 50 are stepped through/passed over the time-series 52 resulting in a sequence of related, overlapping windows. For each window in the sequence of windows (T-3, T-2, T-1, T, T+1, T+2), a figure of merit is found (i.e., the probability that an anomaly or other significant pattern is present in that window). The sequence of figures of merit is examined for overlapping segments. In the example of Fig 3.18, the pattern may have the highest figure of merit, for instance, in windows T-1 and T. The conclusion is that the anomaly exists in the overlapping windows T and T+1.

In general, the approach of setting up machine learning for pattern detection is to identify and associate two elements during the training of the machine learning algorithms: (1) the time-series that contains the pattern and (2) the indicator to be associated with the pattern. A time-series is used to define training instances using the windowing approach, defined in more detail below, while the indicator is used to associate a class with the instance. Due to the classification capacity of deep neural networks (DNN), it is not necessary to be precise with selection of the duration of the time-series. With sufficient training, the network can self-adjust to find the pattern. In the example of pattern detection in SNR analysis, the time-series included measurements and the indicator was the threshold crossing. Notice that the indicator can be something completely different from the time-series, such as the loss of a video signal, when the time-series relates to the fill level of network buffers. For example, for the car example, the time-series can be measurements from the engine, while the indicator may be that the car does not turn on. In addition to network use cases and the use cases described above, pattern detection using data obtained from a time-series can have other applications, as will become evident from an understanding of the description in the implementation.

Fig 3.19 is a diagram of a two-dimensional (e.g., two variables) moving (sliding) window. The sliding windows 60 are stepped through/passed over the time-series resulting in a sequence of related windows, which are stacked together to form two-dimensional matrices. Fig 3.19 illustrates stacking of two rows 64, but multiple rows (e.g., multiple variables) can also be stacked together. For each matrix in the sequence, a figure of merit is found (e.g., probability that an anomaly or other pattern is present). A sequence of matrices is examined to detect the matrix with the highest value and the figures of merit are examined

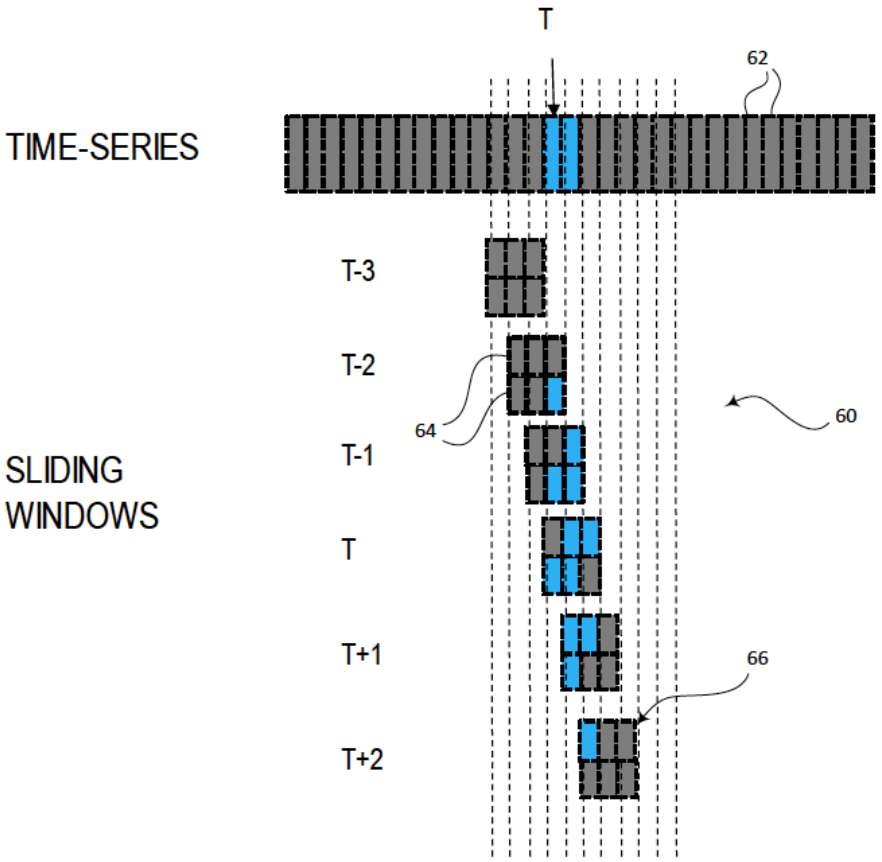


Figure 3.19: A diagram of a two-dimensional sliding window.

for overlapping segments. In the example of Fig 3.19, the pattern with the highest figure of merit, for instance, may be in windows T-1 and T. Thus, the conclusion is that the anomaly exists in the overlapping windows T and T+1.

For illustration, pattern detection is shown using two-dimensional windows 60 over the time-series and deep learning networks. An aspect that enables pattern detection in network time-series is the way the data is windowed to obtain the chunks of time-series and then combine this into two-dimensional windows, applicable to pattern detection.

In addition, Fig 3.19 illustrates the process of obtaining two-dimensional windows from time-series data. The time-series is sampled with even samples that are δ seconds apart. A time window 62 of length m is stepped through/passed over the time-series with a lag l, obtaining a series of horizontal vectors with length m. The horizontal vectors are grouped in groups of n (where n = 2 in the example of the two-dimensional matrices) and then stacked to obtain matrices of size $m \times n$. A matrix is obtained for every lag, resulting in a series of overlapping matrices i_k, which can be referred to as images and can be processed using image processing techniques.

The systems and methods use the two-dimensional windows and a deep convolutional neural network (CNN) for pattern detection. The pattern detection training procedure can be summarized as follows: (1) obtain two-dimensional windows from the time-series, (2)

use a back-propagation algorithm to train a CNN with the windows, details of which are well known in the machine learning area. The pattern detection online procedure can be summarized as follows: (1) upon receipt of a new time-series, obtain new two-dimensional window and pass it to the trained CNN, which provides the classification at its output.

In one embodiment, image pattern recognition CNN is used. This means that the time-series is converted to an image. Fig 3.18 shows how the windowing is performed. The time-series is shown with vertical bars 54, where each bar 54 may correspond to a time-series sample. If a multi-dimensional time-series is used, the vertical bar 54 may be a column vector. A sliding window 50 is used to select a subset of time-series values, which are close together in time.

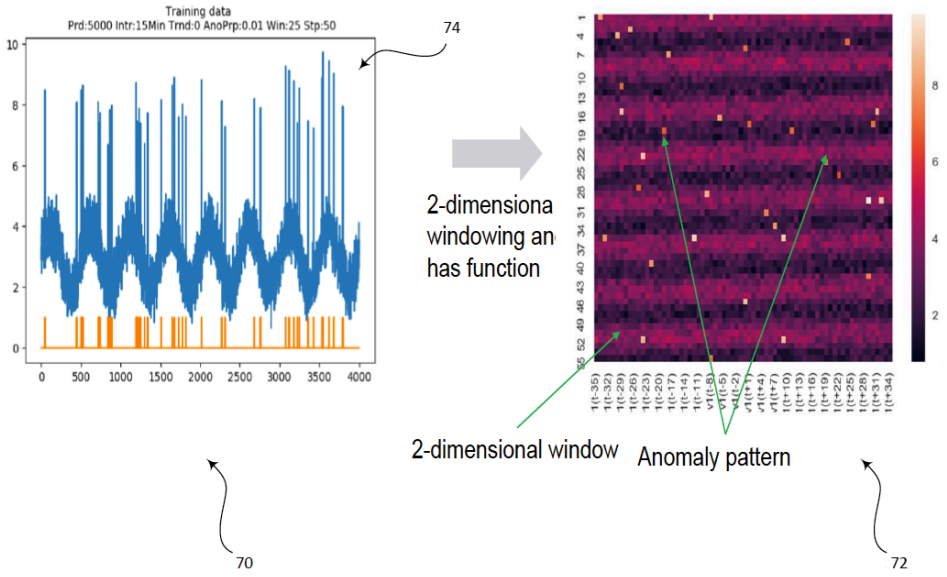


Figure 3.20: A diagram of pattern detection with object identification in images.

In Fig 3.19, two-dimensional sliding windows are shown for times T-3,T-2,T-1,T,T+1,T+2. A two-dimensional sliding window 60 can be obtained from multiple one-dimensional time-series windows 50 by stacking consecutive windows on top of each other to obtain matrices 66, as shown in Fig 3.19.

A special feature of the windowing procedure, combined with machine learning, is that it can be used to localize the pattern in time. In Fig 3.19, the windowing procedure obtains several windows T-3 to T+2. As the pattern may be mostly localized in window T in this example, the conditional probability of the anomaly or pattern presence is the highest in that window, thus localizing the pattern as starting at time T.

A procedure can be devised on top of this procedure to search for the optimum window size as well. That procedure will repeat the search for the pattern using a number of window sizes W for each of the time slots T. The window size W with the highest conditional probability at time T is the best window size for the anomaly. This procedure is used during the training of the classifier, so in fact the classifier is trained with multiple window sizes W on the training data set and the windowing procedure T is used on the testing set to select the best W by picking the combined classifier and window size.

Going beyond a simple CNN, a similar procedure can be used with a regional convolutional neural network (R-CNN), which may be one of the preferred implementations. The R-CNN conceptually takes the two-dimensional image 66, separates out multiple non-overlapping image regions and applies pattern detection to each region in parallel. Using this approach, it is possible to examine the time-series for multiple different overlapping patterns. The training and usage procedure for R-CNN is the same as for the CNN, but instead of training and using a CNN, R-CNN is used. Since the conceptual version may be computationally expensive, other R-CNN procedures such as "faster R-CNN" and "mask R-CNN" may be used instead, but with the same general functionality. For example, the concept of "faster R-CNN" is defined in Faster R-CNN: towards real-time object detection with region proposal networks, by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15), 2015, C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 1, MIT Press, Cambridge, MA, USA, 91-99. Also, the concept of "mask R-CNN" is defined in Mask R-CNN, by K. He, G. Gkioxari, P. Dollár and R. Girshick, IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

Generally speaking, mask R-CNN has the highest pattern detection capabilities. It uses the special structure of the underlying CNN to find a very precise border around the pattern in the image. This contrasts with the CNN or other R-CNN procedures, which uses a square bounding box, which may introduce noise. Other advantages of using a mask R-CNN is that it can examine larger two-dimensional windows and find multiple types of patterns. The larger window may result in better precision. While finding multiple patterns is possible with a CNN, this must be done in series. One advantage of the R-CNN is that it can find multiple patterns in parallel.

The approach in creating two-dimensional windows can be used to create multi-dimensional matrices (e.g., tensors) as well. A tensor is obtained when two-dimensional windows 64 are stacked on top of each other. This can be used to discover patterns that exist across multiple time-series. For example, suppose that it is determined that congestion occurs if two or more related or dependent actions occur at the same time, such as if a first group of specific buffers are over 80 percent utilization and another specific buffer is over 40 percent utilization. An approach that examines buffer time-series independently would not discover this correlation resulting in congestion.

Fig 3.20 is a diagram of a graph 70 using pattern detection with object identification in images. Fig 3.20 shows how the sliding window can be used to detect patterns in time-series. For the purposes of an example, a hash function is used to convert real number values into 3-color (shaded) pixels using a color map 72. The spikes 74 on the graph 70 show up as bright spots on the color map 72. The dark horizontal areas on the color map 72 correspond to the seasonality shown on the graph 70. Other functions (e.g., Fourier transforms) are also possible.

The systems and methods of the implementation provide an improvement over classical machine learning algorithms, which do not perform particularly well with regard to time-series data, especially since time-series data includes certain characteristics that most algorithms are not designed to handle. However, the models or algorithms that may be developed according to the teachings of the implementation may use image processing tech-

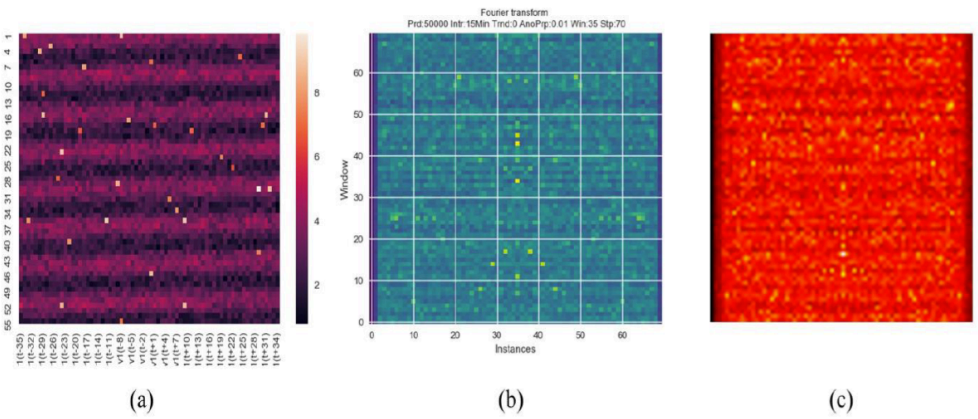


Figure 3.21: A block diagram of another two-dimensional architecture using a Special-Masked CNN (SMCNN), according to various embodiments of the present thesis.

niques for processing the time-series data. By processing the time-series data a certain way, the present systems and methods can produce an image, such as a feature map or color map, and utilize the image information to detect patterns. Thus, it has been discovered that patterns in the time-series may show up as an object in the image generated from the time-series data. By using object detection methods, it is possible to detect patterns in the data.

To prevent errors due to distortion, the window is selected to be large enough to contain the pattern, which introduces the problem of localizing the pattern in the window where it was detected. The problem can be solved with a "sliding window" approach. A sliding window is used to generate a sequence of inputs to the trained machine learning algorithm. The pattern is localized by detecting which windows in the sequence contains the pattern.

Fig 3.22 is a flowchart showing a method 80 of pattern detection and real-time detection. The method 80 includes receiving network measurements (step 82). The network measurements are stored (step 84). Steps 82 and 84 represent a data collection phase. After storing measurements, as indicated in block 84, the method 80 branches into two parts of a pattern detection phase. A first part of pattern detection includes training and a second part includes detection.

In training, the method 80 includes reading network measurements (step 86) and time-bin measurements 88. For time-bin measurements, tags are created (step 90). Also, window measurements are performed, and labels are added (step 92). The method 80 also includes training an algorithm (step 94). From creating tags (step 90) and training the algorithm (step 94), the method 80 includes producing a model (step 96).

In the detection portion of the pattern detection phase, the method 80 includes obtaining time-bin measurements (step 98) of new data. From the model produced in block 96 and the time-bin measurements 98, window measurements (block 100) are performed. From the model (block 96) and window measurements (block 100), the method 80 includes classifying windows (step 102). Then, the patterns may be reported (block 104).

In the detection portion of the pattern detection phase, the method 80 includes obtaining time-bin measurements (step 98) of new data. From the model produced in block

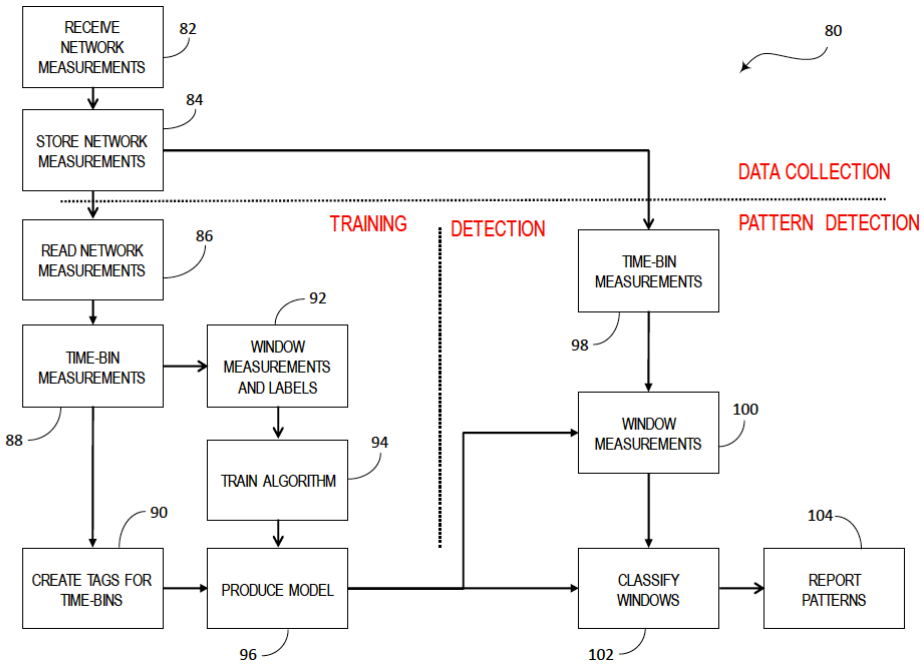


Figure 3.22: A flowchart of pattern detection and real-time detection.

96 and the time-bin measurements 98, window measurements (block 100) are performed. From the model (block 96) and window measurements (block 100), the method 80 includes classifying windows (step 102). Then, the patterns may be reported (block 104).

Fig 3.23 is a flowchart of a procedure 110 for searching for optimum parameters and transformations. Hyper-parameters of interest are provided to the procedure 110 before pattern detection starts. Transformations are also provided before the procedure 110 starts. The procedure 110 is executed to find the best transformation for optimized hyper-parameters. Key Performance Indicators (KPIs) include Accuracy, confusion matrix (False Positive Rate, False Negative rate), or functions of these.

The procedure 110 includes selecting hyper-parameters (step 112). For each hyper-parameter (block 114), the procedure 110 includes finding the best transformation (block 116) and recording the KPIs (block 118) for the hyper-parameter. The procedure 110 is repeated for each of the hyper-parameters. The best hyper-parameters and transformations are returned (block 120).

Fig 3.24 is a flowchart showing a method 130 of training to select a single best transformation. Every data transformation is evaluated with the same hyper-parameters given to the machine learning algorithm and the best transformation is chosen for the classification. Note that each training pipeline can be performed in parallel.

The architecture in Fig 3.24 includes preparing the training data (step 132) and copying the training data into data streams (step 134). In parallel, the method 130 includes performing transformation # 1-4 (blocks 136-1 through 136-4), training the machine learning algorithm (blocks 138-1 through 138-4), and validating and saving the model KPIs (blocks 140-1 through 140-4).

Fig 3.25 is a flowchart of a method 150 for combining multiple transformations. The

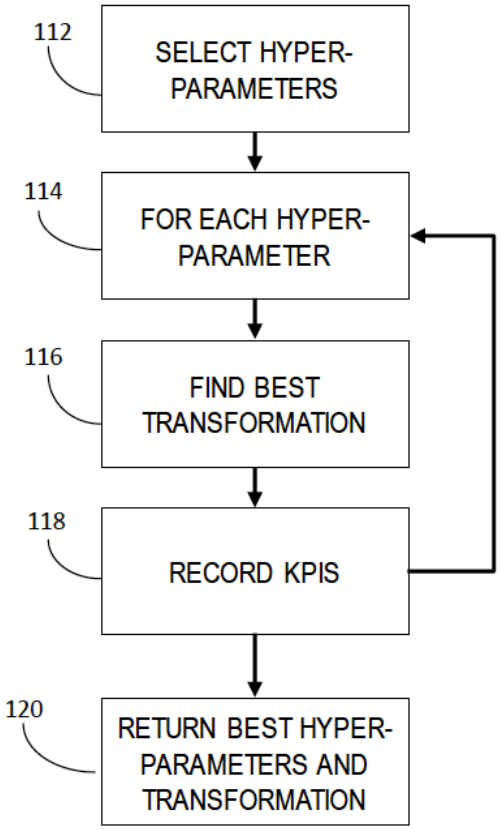


Figure 3.23: A flowchart of a search for optimum hyper-parameters and transformations.

method 150 include preparing the training data (block 152) and copying the data into data streams (block 154). Multiple parallel combinations of data transformation (blocks 156-1 through 156-4) can be used. In this example, the combinations include a first combination ($1 \oplus 2$) for training a first machine learning algorithm 158-1, a second combination ($1 \oplus 2 \oplus 3$) for training a second machine learning algorithm 158-2, a third combination ($1 \oplus 3 \oplus 4$) for training a third machine learning algorithm 158-3, and a fourth combination ($3 \oplus 4$) for training a fourth machine learning algorithm 158-4. In other embodiments, transformations can be used in series. The method 150 also includes validating and saving the KPIs (steps 160-1 through 160-4) for the four algorithms.

Fig 3.26 is a flowchart of a method 170 for combining parallel data transformations. Input data is copied into data streams (block 172). Multiple data transformations (blocks 174-1 through 174-4) can be combined into a single transformed data. Each component data transformation changes the dimensions of the input data, i.e., final data is aligned to the same dimension matrix. Multiple transformations with multiple dimensions may be combined. The method 170 also includes creating (block 176) a transformed data matrix of the data transformations, which can be a simple copy, linear operator (weighted sum, matrix multiplication), or non-linear operator to produce final transformed data.

A preparation step may involve taking the transformed data streams and producing a multi-dimensional stream to be consumed by a machine learning algorithm. The prepa-

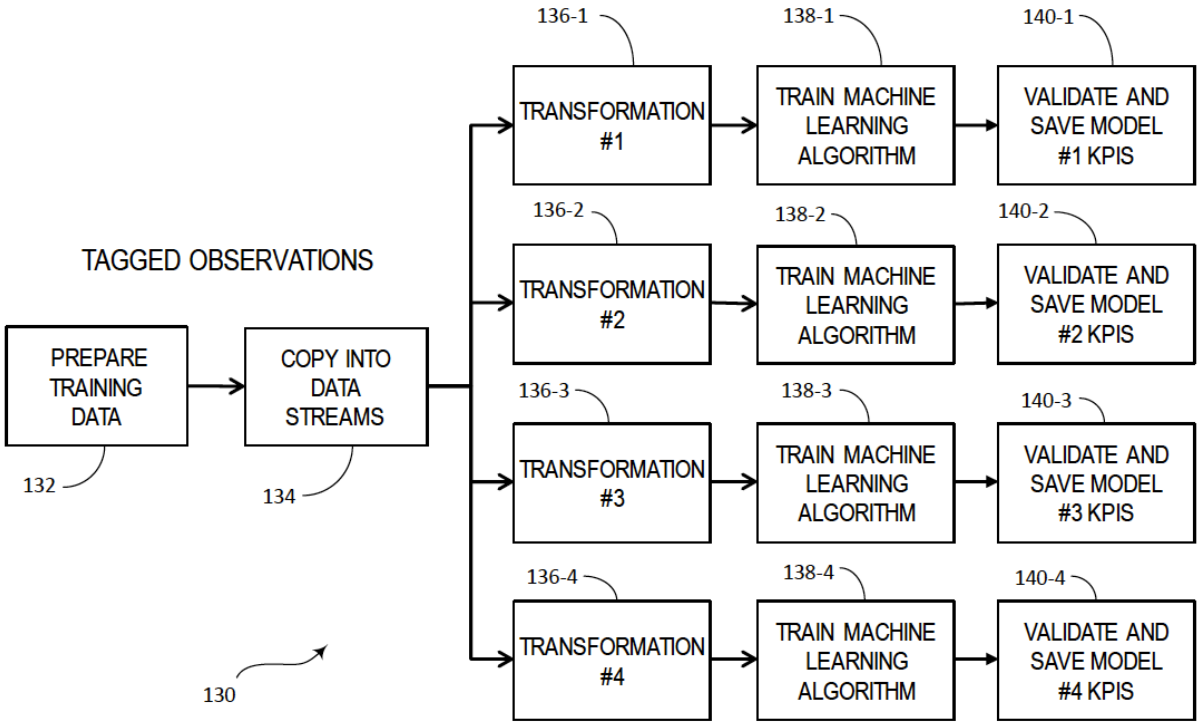


Figure 3.24: A flowchart of training to select a single best transformation.

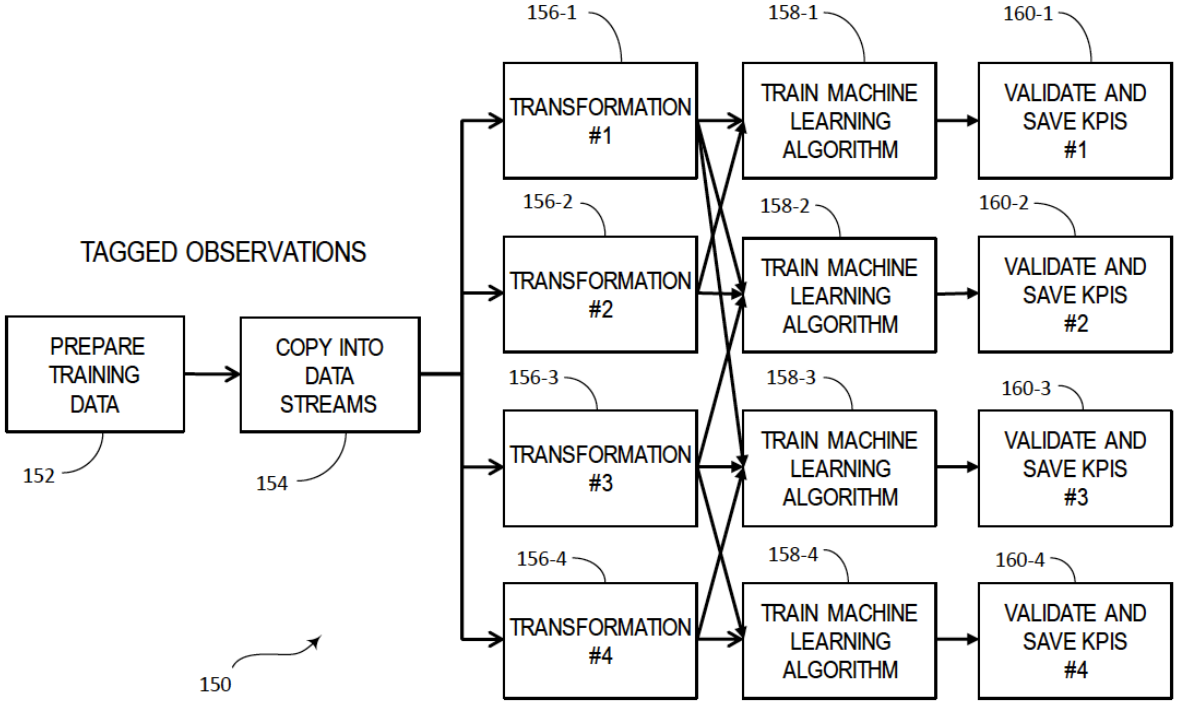


Figure 3.25: A flowchart of combining of multiple transformations.

ration step is selected during the training of the machine learning algorithm. The multi-

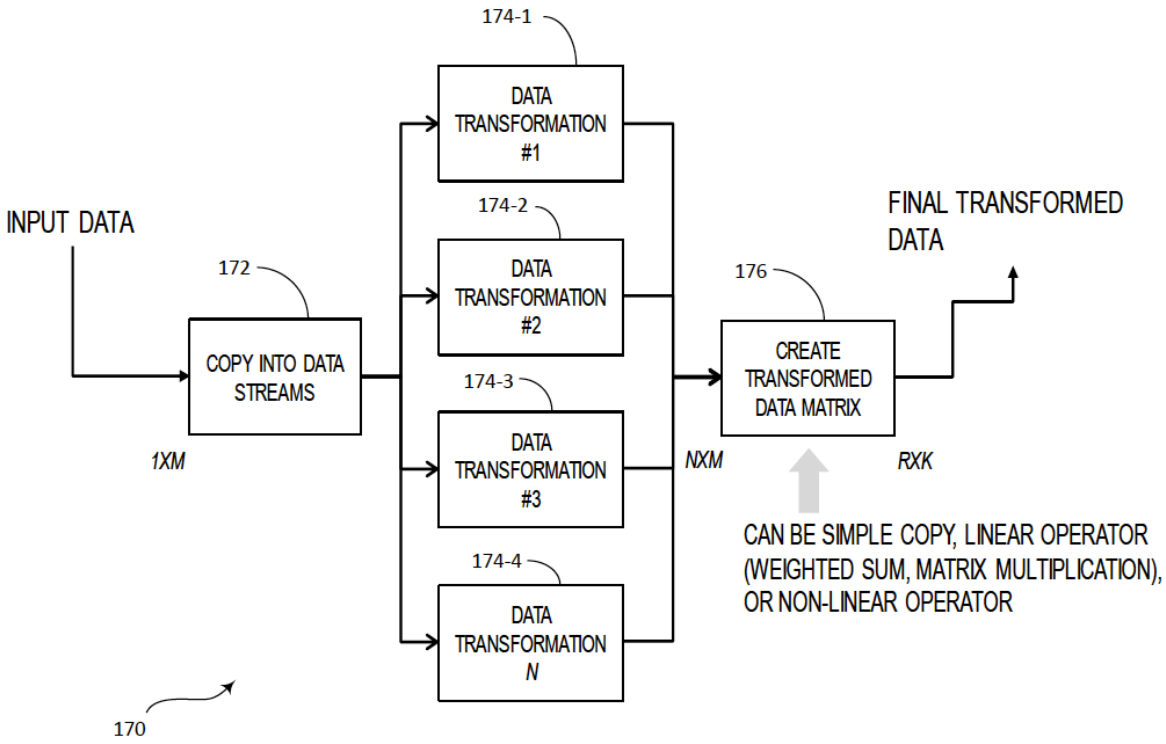


Figure 3.26: A flowchart of combining parallel data transformations.

dimensional scheme may be produced in many ways, such as by: stacking transformed streams without modifications; selecting one transformed stream and return it; obtaining a weighted sum of transformed streams; multiplying stacked streams by the matrix (multidimensional weighted sum); and passing stacked streams through a non-linear function (e.g., neural network).

Fig 3.27 shows graphs of examples of data transformations. A first graph 180 shows the distance between maximums; graph 182 shows the distance between minimums; graph 184 shows the accumulated change; and graph 186 shows the rate of change. Data transformation includes converting obtained time-series data into a time-series more appropriate for a machine learning algorithm. Other basic transformations may include time-bin measurements, feature extraction (e.g., principal component analysis - PCA), detecting first difference of samples, etc. Fig 3.27 illustrates other example transformations and can be thought of as dimensionality reduction on the time-series data.

Fig 3.28 is a flowchart of a process 190 for anomaly detection in network data. First, network observation data is prepared (block 192). Data may be cleaned to handle missing values, time-bin, etc. Next, optimization or a search is performed for both the hyperparameters and transformations (block 194). The algorithm is trained with the multiple transformed data. Since many transformed data or their derivatives are given to the algorithm, this may result in multiple models. Data may be transformed into prepared data to improve machine learning performance. A compound data transformation may be constructed from multiple other data transformations. One or more data transformations may be provided to determine, which one, or which combination of them is the best to use with

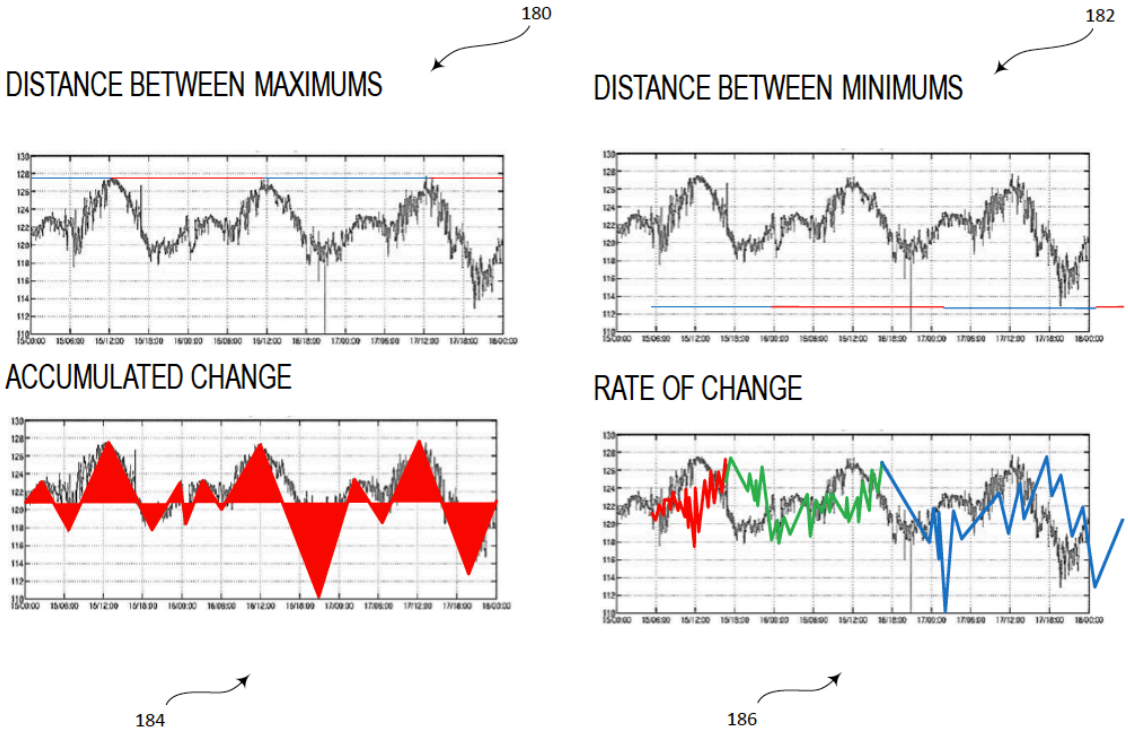


Figure 3.27: Graphs of examples of data transformation.

network observations. The machine learning algorithm coupled with a data transformation becomes a new enhanced machine learning algorithm. Third, the best performing model is chosen (block 196). The best model determines the best data transformation, or best combination of data transformations. The best model is selected based on a key performance indicator (KPI) relevant to how the model is going to be used for prediction/classification (e.g. smallest false positive rate, smallest prediction latency, highest true positive rate for a given maximum false positive rate, etc.). It is noted that selecting the model in this way is in fact searched over a hyper-parameter space of models and results in the "optimal" model for the machine learning task at hand. The selection may be performed during the validation stage of the training. Finally, anomalies are detected (block 198) using the best model.

is a block diagram of yet another two-dimensional CNN architecture 400, using a special masking technique. The architecture 400 of Fig 3.29 may be referred to as a Special-SMCNN. The pre-processing block 402 transfers the time-series data in raw format from one or several sensors to a two-dimensional representation. The frequency-bands block 404 may be used for applying a Fourier Transform. A two-dimensional CNN block 406 learns the normal and anomalous data. A feature map 408 is created with rectangular bounding boxes that surrounds the anomalies. The rectangular bounds may be reshaped to fixed squares within a fix feature map 410. A masking block module 412 creates flexible boundaries that may explicitly surround the anomalies. A fully connected block 414 provides classification and box regression. A meta learner 416 receives input from the masking branch module 412 and classification from the fully connected block 414 and provides models to one or more

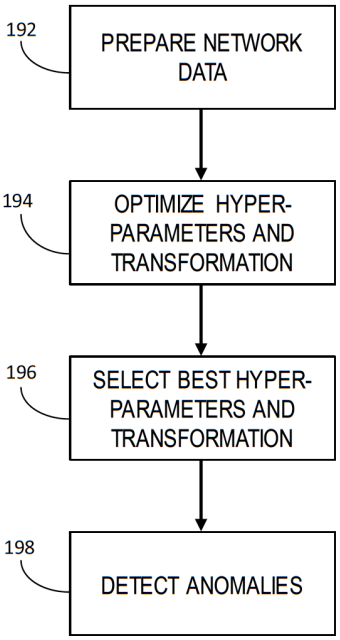


Figure 3.28: Graphs of (a) a heat-map of two-dimensional representation of time-series data - seasonality can be seen as vertical shades; and (b-c) Fourier transformed data.

special CNNs 418, which may include special convolutions. Output from the special CNNs 418 and fully connected block 414 are provided to an anomaly detection block 420, which may be configured to adjustably define the normal/anomaly threshold of classification.

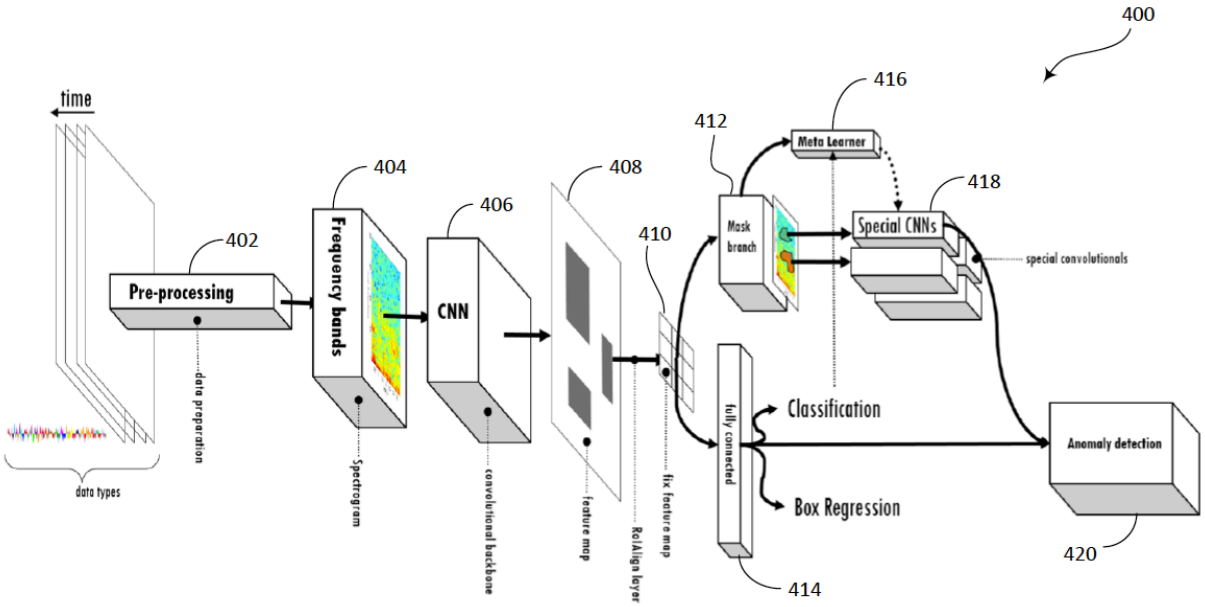


Figure 3.29: A flow diagram of a method for detecting patterns in time-series data.

Fig 3.30 is a flow diagram illustrating the method for detecting patterns in data. The method includes obtaining data in a time-series, as indicated in block 432. From the time-series data, the method includes creating one-dimensional or multi-dimensional windows, as indicated in block 434, wherein the one-dimensional or multi-dimensional windows are created either independently or jointly with the time-series. The method 430 further includes the step (block 436) of training a deep neural network with the one-dimensional or multi-dimensional windows utilizing historical and/or simulated data to provide a neural network model. Ongoing data from a network is processed with the neural network model (block 438) to detect one or more patterns of a particular category in the ongoing data. The method 430 also includes localizing the one or more patterns in time, as indicated in block 440.

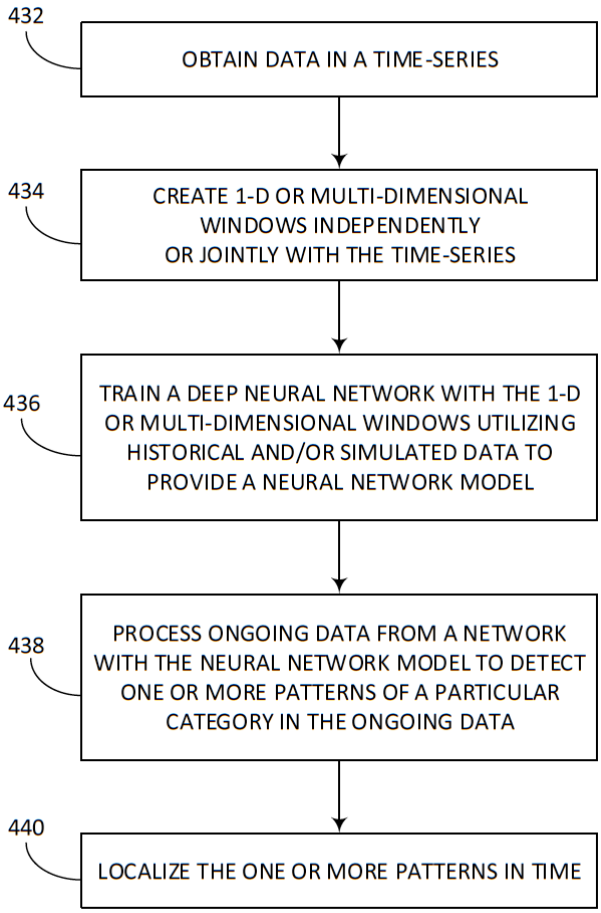


Figure 3.30: A block diagram of a server which may be used to implement the systems and methods described herein.

3.8 Proposed Method for Automatically Removing Repetitive Patterns

Sequential data containing patterns and concept drift result in poor performance of generic anomaly detection models. Majority of systems are not able to differentiate between anomalies and legitimate pattern changes and concept drifts. We introduce a method that automatically removes repetitive patterns and applies similar preprocessing methods that need careful adjustments with no need for a domain expert or data scientist. The following algorithm explains steps in the method for finding optimal transformation configurations.

3.8.1 Automatic pattern matching and sliding (APMS)

We created a solution that automatically fits to the repetitive patterns and subtracts the drift in the data. The solution we introduce here not only solve GAN's concept drift problem, but also can be used for our previous CNNs models to automatically find the optimal 2D image sizes. We introduce a method to automatically create sliding windows that fit the pattern in the data and remove the normal concept drifts. This feature enables GANs to be applicable to data characterized by concept drift. Our approach enables GANs to be applied to sequential data by reformulating the regular sequential data analysis task as an image analysis task. For this purpose, our approach includes a data transformation step that transforms sequential data to an image representation, thus resulting in a higher performance, better generalization and more intuitive outputs than other common approaches in the literature.

Algorithm 1 Automatic Pattern Matching and Sliding

procedure PERIODICITY DETECTION
    Initialize Y
    $i \leftarrow 1$
    Initialize and store values in string S(i) ← WS, FW, NS
loop:
    WS ← randomly generate *window size*
    FW ← randomly generate*start point of window*
    NS ← randomly generate*number of slices in image*

    Generate *X images* using above hyperparameters
    vector V(i) ← subtract every other image*(X(n+1)-X(n), X(n)-X(n-1), ...)*
    $i \leftarrow i + 1$

    if $i < Y$ **then goto** *loop*
    H ← *S(i)* **where***Min(vector V(i))*
close;
Return *vector H* a set of optimal hyperparameters

The method automatically finds the optimal sliding window size, the starting point of

the slides, and the number of slices in an image that fit to the underlying patterns in the data. The solution works by randomly generating window sizes (WS), and start points of the window (FW) and number of slices (NS). Using the random hyperparameters we generate X images from the above parameters and then subtract every second image from the previous one $X(n+1) - X(n), X(n) - X(n-1), \ldots$. We repeat the aforementioned steps for Y times and select the parameters the X generated images produce the overall lowest value (more darker or more black values) as Minima Candid ($MinC$) and select the parameters the X generated images produce the overall highest value (brighter or more white values) as a Maxima Candid ($MaxC$) value. For fine tuning we select the Minima Candid and the Maxima Candid and randomly generate WS, FW, NS between the $MinC$ and first lower generated parameters closest higher generated parameters in previous steps and repeat steps. The Automatic Pattern Matching and Sliding algorithm is explained below.

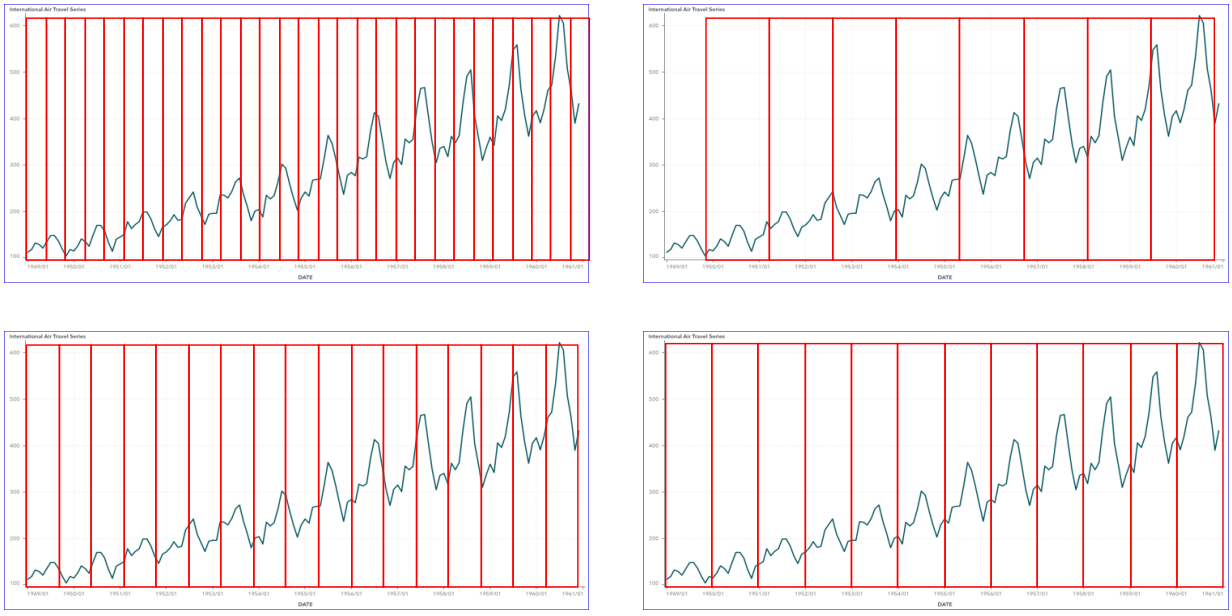


Figure 3.31: Transforming the slicing window to two dimensional representation. The method enables CNNs to be able to capture the notion of time. The transformation also can provide an standard input to CNN from various size data slices and time durations. In some data types the total duration of the 2D representation is clear, one year for this example. In some other data the hight of the 2D window also needs to be adjusted (width is equal to the window slice size. We can perform the same method here to fined the number of window slices in one 2D image.

3.8.2 Flexible automatic pattern matching and sliding (FAPMS)

In many applications, the repetition of the data patterns are not exactly aligned in the same slice of time. A travel from point A to B might have similar characteristics to a flight from C to D, but with various time lengths. Lantao Yu et al. [126], propose a sequence generation framework, that is able to handle sequences and generate new sequences. Zenati

Algorithm 2 Flexible Automatic Pattern Matching and Sliding

procedure PERIODICITY DETECTION
    Initialize Y
    $i \leftarrow 1$
    Initialize and store values in string S(i) ← WS(1), WS(2), ..., WS(n), and FW, and NS
loop:
    WS ← randomly generate *window sizes each randomly generated from 1 to n*
    FW ← randomly generate*start point of window*
    NS ← randomly generate*number of slices in image*

    Generate *X images* using above hyperparameters
    vector V(i) ← subtract every other image*(X(n+1)-X(n), X(n)-X(n-1), ...)*
    $i \leftarrow i + 1$

    if $i < Y$ **then goto** *loop*
    H ← *S(i)* **where***Min(vector V(i))*
close;
Return *vector H* a set of optimal hyperparameters

et al. [129], introduce an efficient anomaly detection using GAN. Our sequential to image transformation technique, enables a deep convolutional GAN model to learn the distribution of normal data and perform pixel-wise segmentation on anomalies. One of the biggest advantage of our method over all previous methods is the ability to define various size sliding sizes in one data while keeping the repetition points at relativity similar points. For better understanding let us explain with an example. Lets assume we would like to track a person's EEG over time. The person might enter various stages of action that affects the measurements. For example, high values in the time series might be normal if accompanied by high brain activity in certain time frames (e.g. due to watching an interesting scene in a movie) or anomalous. As it would be much more challenging to observe all other activities that might effect our measurements, the more reliable and feasible method is to only consider the differentiation between each two brainwave. However, the repetitions might not aligned equally. A slight modification in the Automatic Pattern Matching and Sliding algorithm provides flexible window sizes. In this variation of our original APMS algorithm window slides can be unequal and our pixel-wise method provides a solution that can deal with the inequality of the sliding windows.

Selecting an efficient window size for sequential data is crucial for high performing models. The data requires to cover a full episode of action to capture all meaningful patterns. Common approaches to select the optimal window size generally require prior knowledge about the data and the domain or application at hand. Also the size of the window can be fixed or vary over different inputs and actions. In air travel data the input can be equal to the duration of the flight that can be vary from several minutes to several hours.

The method can automatically find the repetitive patterns and fit the sliding window to each repetition and then our pixel based input model deals with the variation in the

size of slides and result in better performance. As shown in Fig 7.11, Flexible Automatic Pattern Matching and Sliding produce precise window slides that fits the periodicity of patterns and improve the model detection performance when the data contains concept drifts, see Fig 7.12.

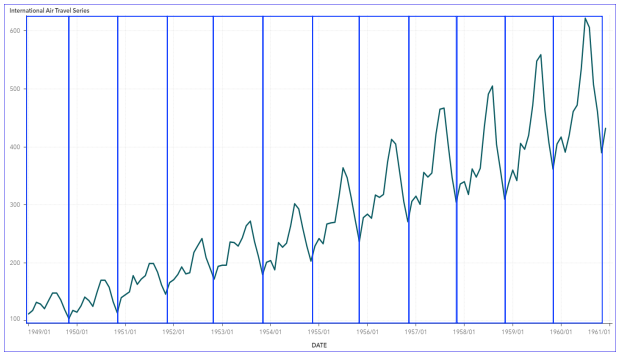


Figure 3.32: Our flexible automatic pattern matching and sliding approach leads to a further improvement in GANs anomaly detection performance with respect to random window transformation.

3.8.3 Evolutionary optimization algorithm

Adaptive systems are often shaped by evolutionary dynamics. The mechanism of evolution starts with variation. Then there is selection of elements that are fit for the changed conditions. These elements flourish and multiply in the system. They may also change the external environment of the system, causing new variation. A new cycle starts and there is no movement to a knowable end point or equilibrium. We apply this concept to generated data to create variations in models to improve their generalizations capabilities. The method is illustrated in following. Using this method we propose a method for creating various types and ratios of model instances similar to real-world systems. Some of the models are significantly deviated from the regular models and configurations to improve finding global minima more efficiently. Unexpected large random variation in the model configurations can produce dramatically worse or maybe surprisingly better results. Using our method we can escape the issue of falling in local minima and the time consuming stochastic and purely random methods. Our method mimics biological evolution by generating slightly different variations or the original models and a few mutations as is observable in the nature, illustrated in below. A more detailed Figure is presented in follow.

In order to create an unsupervised mechanism for crating realistic evolutionary steps both to generate data and to evolve models, we designed a system that creates variation of initial algorithm. Our method is similar to the Daan Wierstra et al. [120], [119] Natural Evolution Strategies, but with a very important and intuitive modification. We have added anomalous mutations to avoid trapping in a local minima. Mutations (anomalies) are often considered as an undesired event, but in the context of evolution they might improve the search for the global-minima, as the mutation might change the regime to a new concept

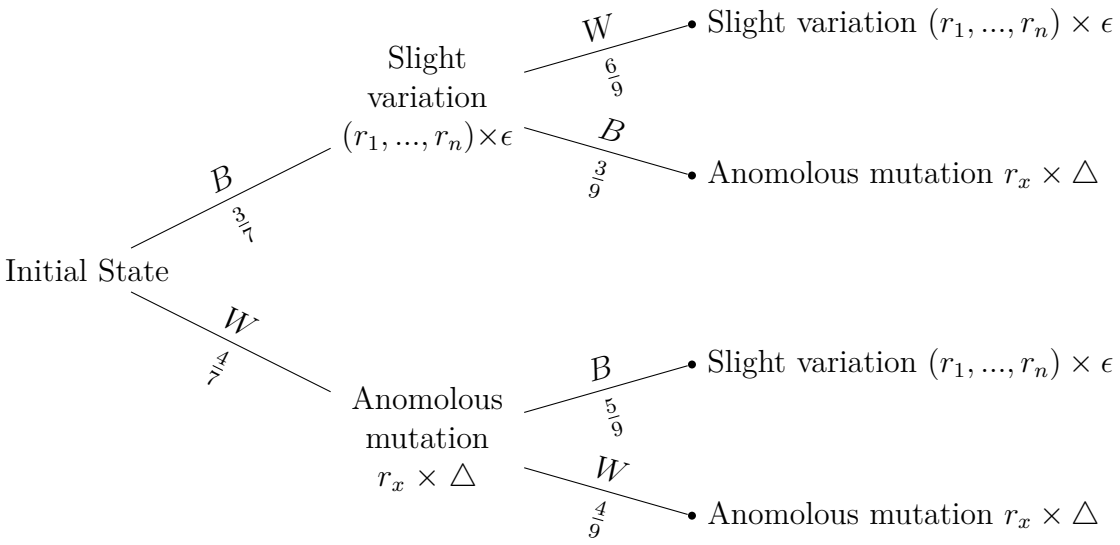


Figure 3.33: Proposed methods for generating realistic data and anomalies for training model from original data *natural generator*

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 50, 50, 32)	896
conv2 (Conv2D)	(None, 50, 50, 64)	18496
pool1 (MaxPooling2D)	(None, 25, 25, 64)	0
flat1 (Flatten)	(None, 40000)	0
dense1 (Dense)	(None, 128)	5120128
dense2 (Dense)	(None, 1)	129

Total params: 5,139,649
Trainable params: 5,139,649
Non-trainable params: 0

Figure 3.34: Our basic convolutional neural network architecture.

that performs far better than the current state. The below illustration defines how in each child a mutant version can be produced and if it performs better that the parents will remain to reproduce with modifications. The gradual evolution in nature formed human for example, but there were some anomalous mutation that created variations that are not necessarily in the path of gradual involvement. Here we are implementing a simplified version of the evolution in machine learning concepts. Only if the mutant's fitness is at least as good as the parent one they can reproduce, otherwise the mutant is disregarded.

Source code: for the source code and documentation visit Github repository or go to the url: `https://github.com/sidryan`.

Chapter 4

Result

4.1 Conventional Models

The plots here are presented based on models that trained with different preprocessing configurations and tested against variety of data characteristics to define the consequences of each configuration. Y-axis represent the performance of each algorithm based on AUC which seems to be an efficient performance score choice for imbalance data and anomaly detection. X-axis indicates the data characteristics of the test set. Figures provide examples of plots obtained by models trained with 10 increasingly changed base-data characteristics and tested against 20 variations in characteristics and concept drifts in the test-set.

Table 4.1 shows the average performance of seven conventional algorithms. The models were trained with data following a specific distribution, and were tested against data subject to concept drift. When data samples are scarce, the NB algorithm performs better than the others. A similar NB's high performance is achieved when there are large waves of seasonality in the data or the size of the sliding window that indicates the input sequence is very large. However, the NB algorithm performs poorly in the presence of high trends and when the steps between the sliding window are relatively high. By contrast, the SVM algorithm, which did not perform better than merely guessing the labels by chance in

Table 4.1: Average AUC obtained by models with different experimental settings. Best results are marked in bold.

	Few Data Instances	High Trend	High Amplitude Seasonality	Low Anomaly Probability	Large Window	Large Steps
Random Forest	51.97	67.13	92.02	**99.0**	96.99	50.93
SVM	50.0	**88.96**	50.0	50.0	50.0	50.0
DT-C4.5	59.79	66.57	87.23	**99.0**	78.57	**92.16**
Adaimprove	54.38	66.50	90.04	95.89	96.87	90.71
Naive Bayes	**92.58**	50.0	**94.27**	**99.0**	**98.98**	50.0
kNN	50.0	70.79	66.19	56.05	93.92	64.03
MLP	66.10	84.66	90.05	95.0	85.14	55.18

almost all situations, outperforms all algorithms when the data contain many trends. When anomalous instances are scarce, the Random Forest, Decision Tree and NB algorithms show superior performances with respect to the other algorithms considered in the analysis.

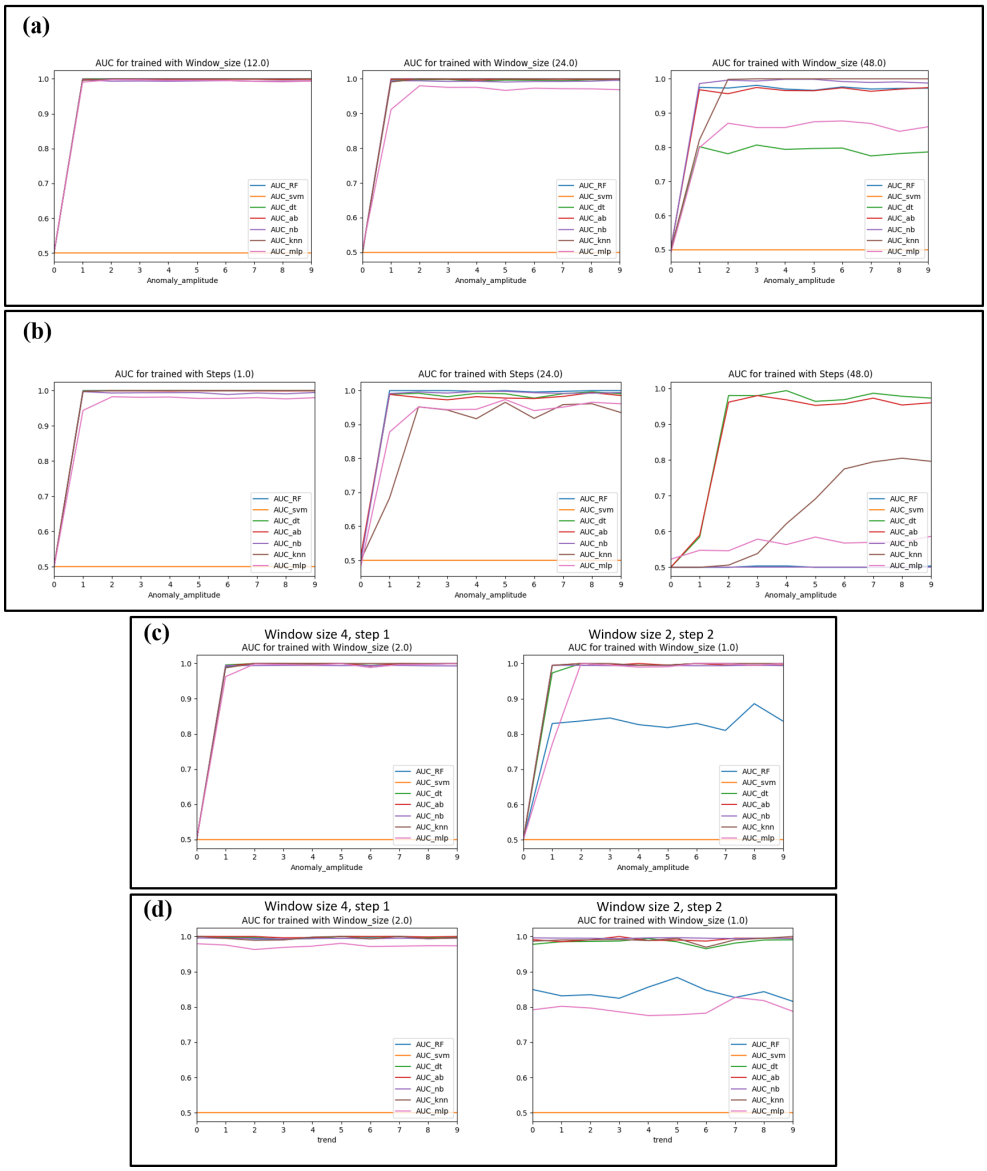


Figure 4.1: (a)Comparison of area under the curve (AUC) with using different sliding window sizes, from left to right sizes 24, 48 and 96 used. The smaller sliding window sizes have higher performance. (b)The overlap sizes of sliding window effects on the AUC. The smaller strides result in higher AUC. (c)The comparison between sliding window size and sliding steps when amplitude of anomalies are changing. Higher overlap results a better performance than sliding window size. (d)Comparing effects of window size and overlap with various trends in the test-set.

In Figure 4.1, (a)is the comparison of area under the curve (AUC) with using different sliding window sizes, from left to right sizes 24, 48 and 96 used. The smaller sliding window

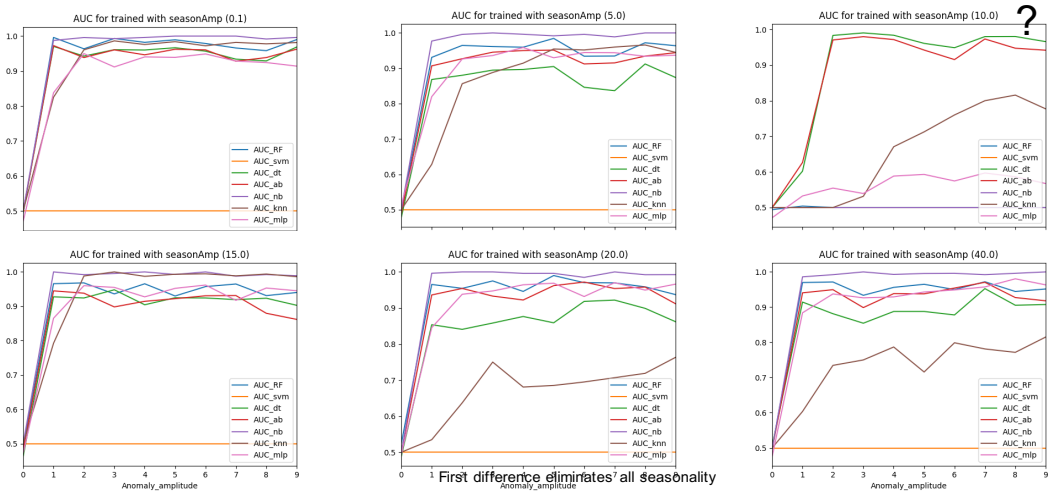


Figure 4.2: Various models trained with different amplitude of seasonality of training compared to anomaly amplitude in test. The seasonality amplitude increasingly increased from top -left to bottom-right. By increasing the amplitude of seasonality in training-set models perform with lower AUC while the anomaly amplitude increase. therefore, it is better to use a training-set that has low seasonality amplitudes in such scenarios.

sizes have higher performance. (b) is the overlap sizes of sliding window effects on the AUC. The smaller strides result in higher AUC. (c) is the comparison between window size and sliding steps when amplitude of anomalies are changing. Higher overlap results a better performance than windows size. (d) is comparing effects of window size and overlap with various trends in the test-set.

Figure 4.2 shows the effect of changes in the seasonality of training compared to changes in anomaly amplitude in test. Plots for all models trained using different pre-processing configurations and tested against a variety of data characteristics. A first-difference estimator was used to simplify the data removing the underlying trend, and consequently improving the performance of the algorithms. Although this scenario appears unrealistic in many real-world applications, it helps to directly observe the effects of individual changes in the data characteristics without being concerned about the performance of the models.

Figure 4.3 shows the effect of Size of training compared to anomaly amplitude in the test set.Training and test in all examples have same seed to get reproducible results and conclusions. Intuitively, the more data samples is training set results in a better performance. However, higher trends and seasonality amplitudes in the data decrease the AUC. Nevertheless, the SVM algorithm start to outperform other algorithms in contexts with a high amount of trend in data. Additional experiments revealed that a higher probability of existing anomalies and related amplitudes in the data have positive effects on the AUC.

Fig 4.4 shows the trend in training set compared to trend of test.

Figures 4.5 shows how the trend in training set affects the performance of the model compared to trend of test set.

The sliding windows size/overlap of training compared to trend in test set as shown in Fig. 4.6 shows, a bigger window with higher overlap is better than smaller with less

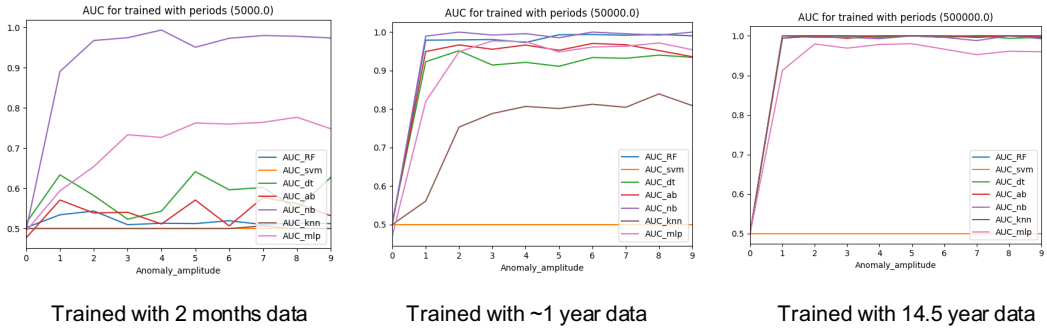


Figure 4.3: Size of training compared to anomaly amplitude in test.Training and test have same seed to get robust conclusions. From left to right, three training-sets containing 5000, 50000 and 500000, used for train models. Higher number of training samples produce better performance. When training-set is small Nabive Bayes performed the best and MLP performed average.

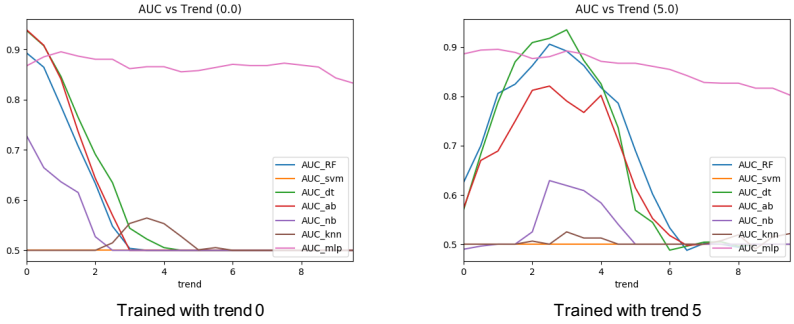


Figure 4.4: Trend of training compared to trend of test. a models trained with zero trend in the training-set performed best in similar data characteristics in the test-set, while a model trained with trend 5 (5% growth in one year) performed best when the trend in the training set is lower that test-set, around 3% growth.

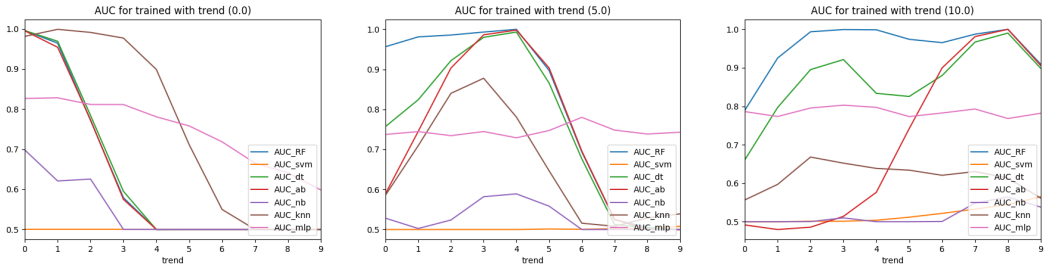


Figure 4.5: Trend of training (14.5 years data) compared to trend in test.

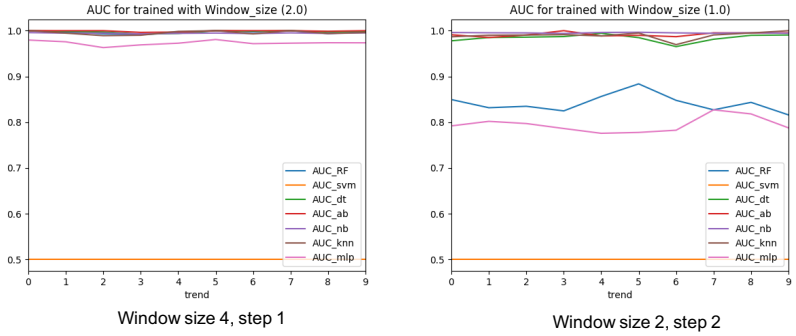


Figure 4.6: Sliding windows size/overlap of training compared to trend in test. A bigger window with higher overlap is better than smaller with less overlap, meaning that to increase the number of training samples, the overlap in sliding windows is more effective that choosing a smaller size sliding window.

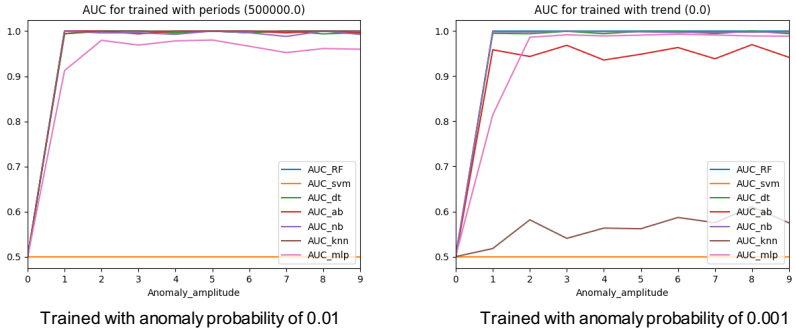


Figure 4.7: Anomaly probability of training compared to anomaly amplitude in test. The more probability of anomalies, the better. We used 1 anomalous pattern in every 100 samples and 1 anomaly among 1000 samples, and the former performed better.

overlap, meaning that to increase the number of training samples, the overlap in sliding windows is more effective that choosing a smaller size sliding window.

Anomaly probability of training compared to anomaly amplitude in test and the result is shown in Fig. 4.7, the more probability of anomalies, the better. When there is higher number of anomalous patterns in the training-set the models perform better. For example we used 1 anomalous pattern in every 100 samples and 1 anomaly among 1000 samples, and the former performed better. When the is not many anomalies in the training set, lowest performance belonged to kNN followed by Adaimprove.

Anomaly amplitude of training compared to anomaly amplitude in test and is shown in Fig 4.8. Very small anomaly amplitudes, dramatically drops performance.

The trend of training set compared to anomaly amplitude in test (see Fig 4.9), the smaller the trend, the better. Three models trained with 0%, 5% and 10% trends in training data and tested against different datasets with a variety of anomaly amplitude.

Figures 4.10 shows how sliding window size of training set affects the accuracy comparing to the various anomaly amplitudes in test, the smaller window size, the better.

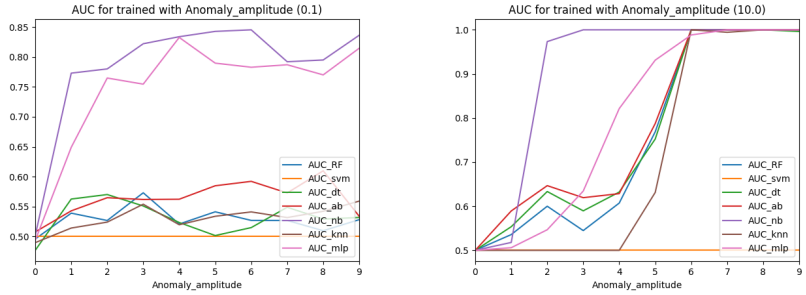


Figure 4.8: Anomaly amplitude of training compared to anomaly amplitude in test. Very small anomaly amplitudes, dramatically drops performance.

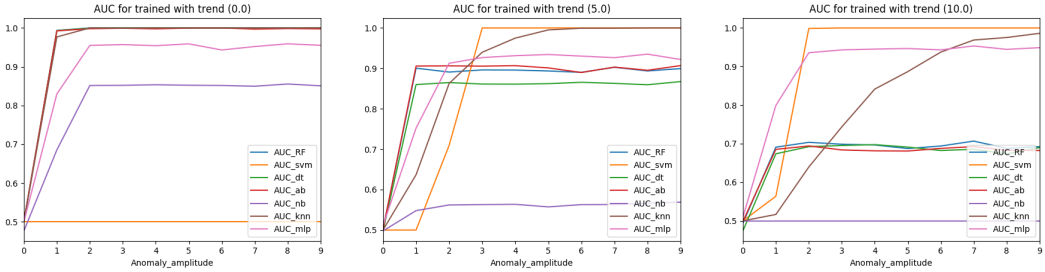


Figure 4.9: Trend of training compared to anomaly amplitude in test. The smaller the trend, the better.

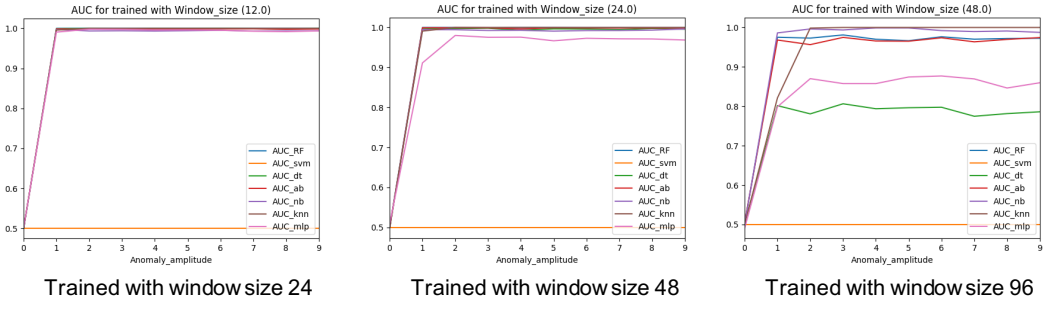
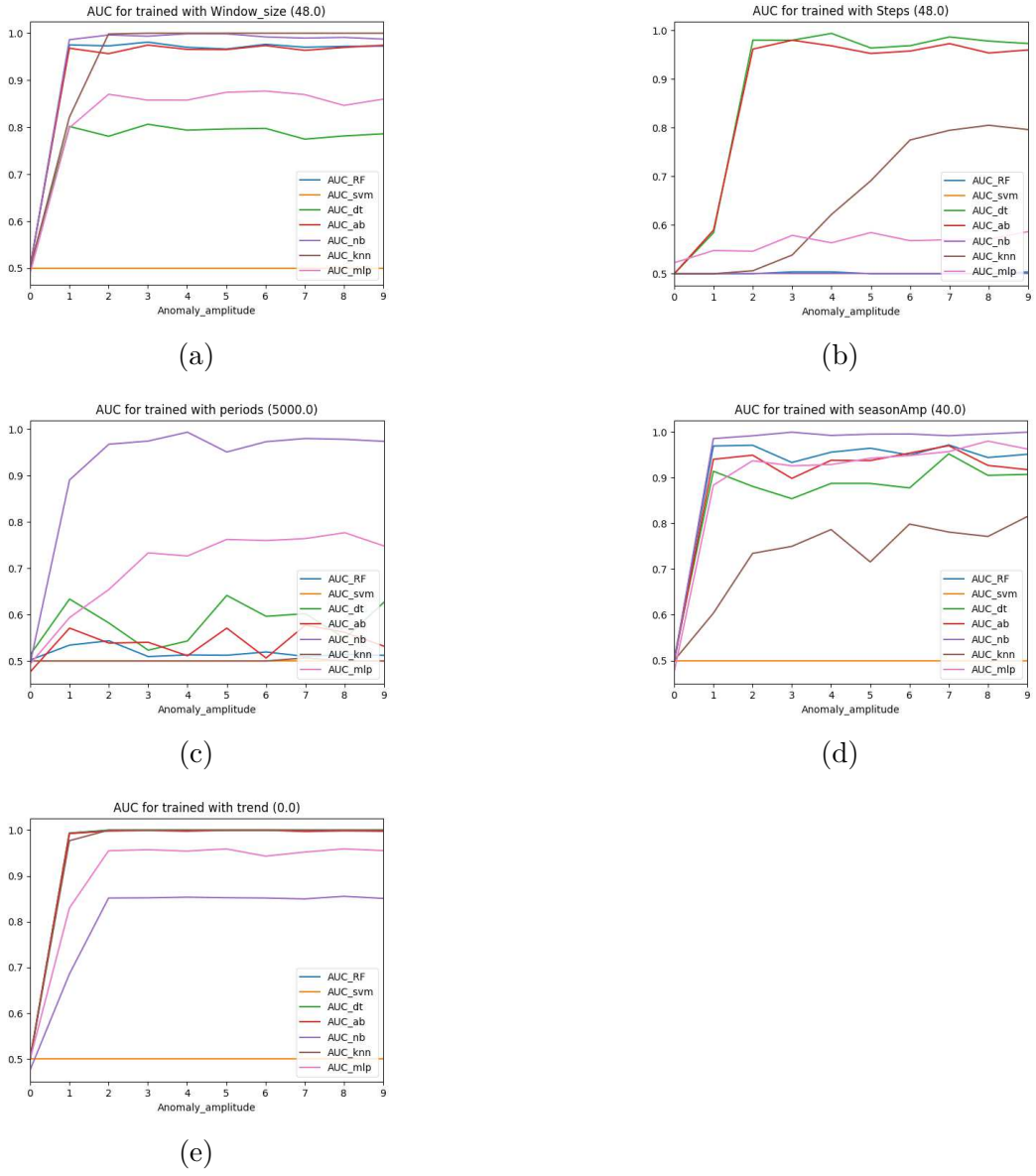


Figure 4.10: Sliding window size of training compared to anomaly amplitude in test. The smaller window size, the better.

A smaller sliding windows exhibits a higher score, similarly, increasing overlap results in a higher AUC, and overlaps have a higher impact on the AUC compared to smaller window sizes. Results showed that the higher performance produced by simpler algorithms when data instances are limited.



(a) (b) (c) (d) (e)

Figure 4.11: Comparison of AUC while applying first-difference estimator (first row). Smaller sliding windows exhibit a higher score (second row). An increased overlap results in a higher AUC, and overlaps have a higher impact on the AUC compared to smaller window sizes (third row). Results show the higher performance produced by simpler algorithms when data instances are limited.

Overlap of training compared to anomaly amplitude in test and the result is show in Figures 4.12, the more overlap in sliding window the better.

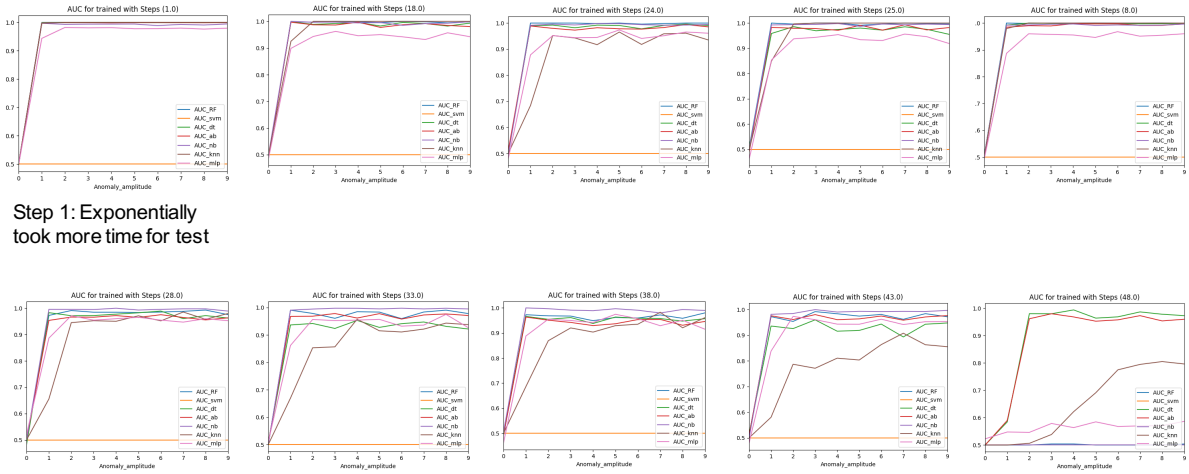


Figure 4.12: Overlap of training compared to anomaly amplitude in test. The more overlap in sliding window the better.

Figure 4.13 shows examples of plots for models trained with base data characteristics and tested against their variations. From the top row, left to right, the models that were trained with a low amplitude in the training set were tested against ten test sets where the amplitude of the anomaly was progressively increased. These tests illustrate the effects of the anomaly amplitude, and these data contain no trends or seasonality. In very low amplitudes in the training set, NB and MLP outperform the other methods. The plots obtained by performing early detection as described in the Methodology Chapter, in pattern analysis and finding anomalies in sequential data.

Figures 4.15 shows the effect of sliding windows size/overlap of training on performance when anomaly amplitude in test set varies. Bigger sliding window with higher overlap is better than smaller with less. Overall, we can observe that a smaller window size and a higher overlap in the sliding window increases the AUC. Moreover, the overlap affects more significantly the results compared to the window size.

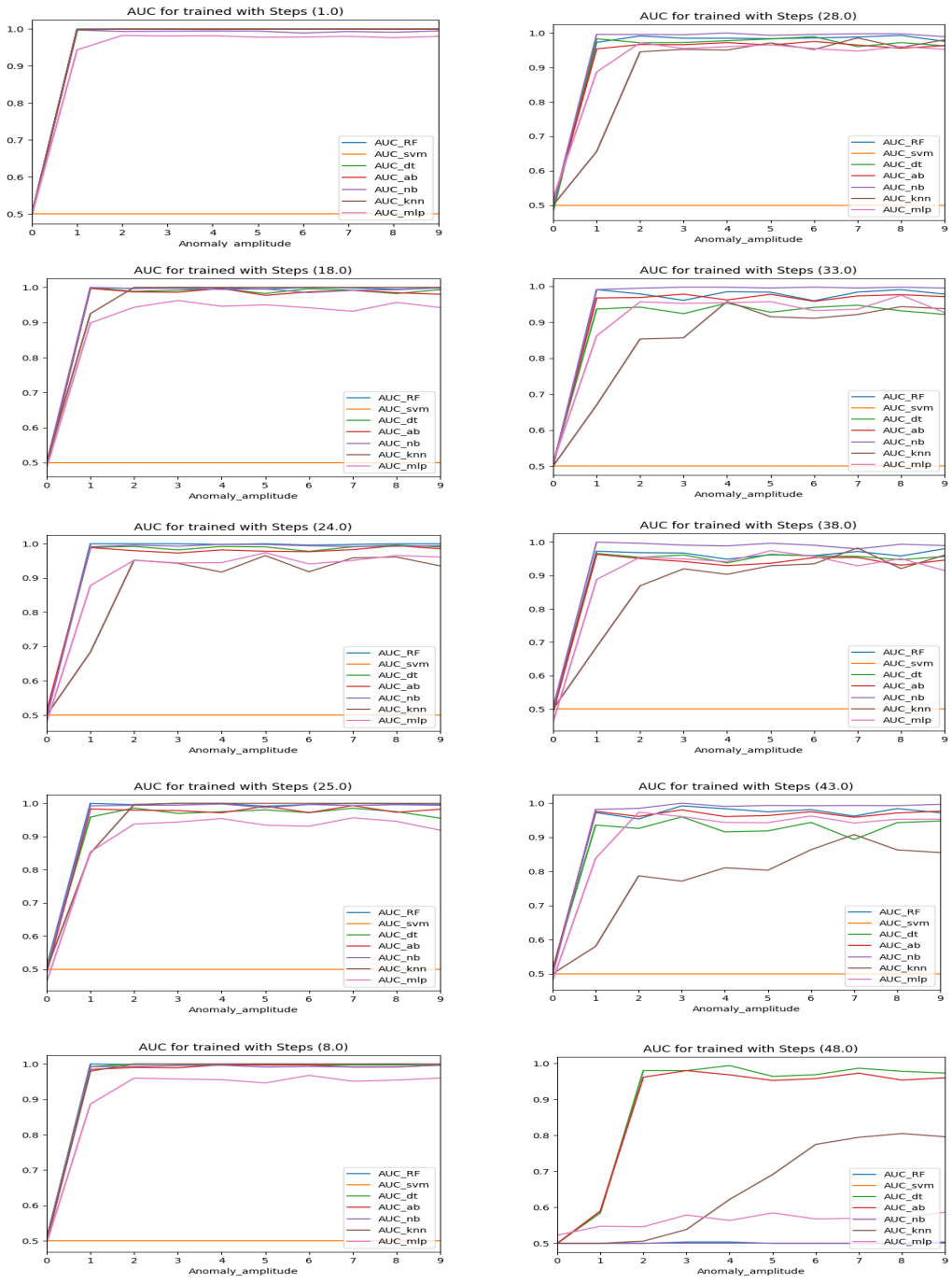


Figure 4.13: Examples of plots from various models trained with base data characteristics and tested against their variations. From the top row, left to right, the models that were trained with a low amplitude in the training set were tested against ten test sets where the amplitude of the anomaly was progressively increased. These tests illustrate the effects of the anomaly amplitude, and these data contain no trends or seasonality. In very low amplitudes in the training set, NB and MLP outperform the other methods.

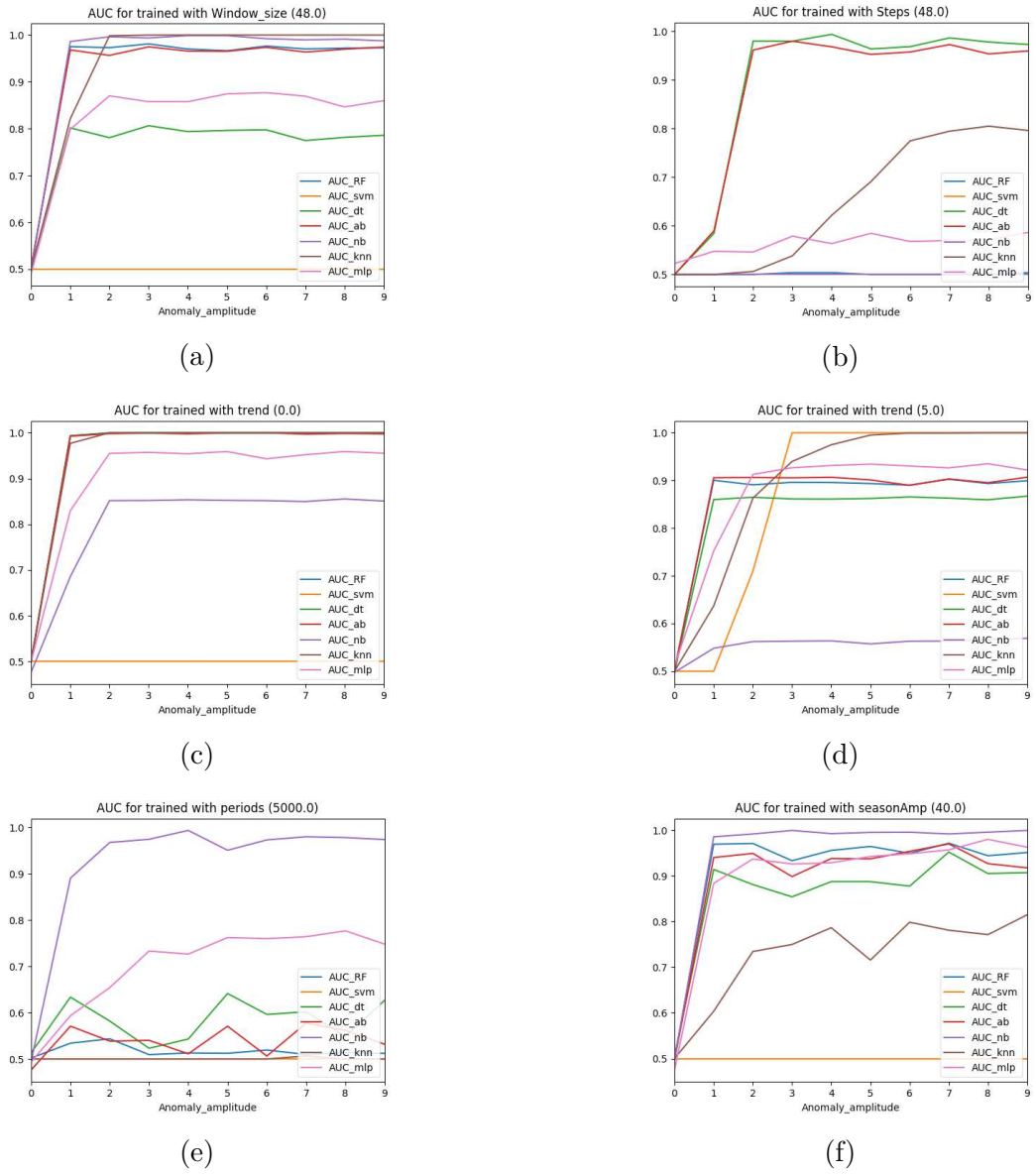


Figure 4.14: Comparison of AUC while applying first-difference estimator (first row). Smaller sliding windows exhibit a higher score (second row). An increased overlap results in a higher AUC, and overlaps have a higher impact on the AUC compared to smaller window sizes (third row). Results show the higher performance obtained by simpler algorithms when data instances are limited.

Fig 4.14 shows a comparison of AUC while applying first-difference estimator (top row). Smaller sliding windows exhibit a higher score (middle row). An increased overlap results in a higher AUC, and overlaps have a higher impact on the AUC compared to smaller window sizes (bottom row). Results show the higher performance resulted from simpler algorithms when data instances are limited.

Fig 4.16 shows the effects of low anomaly amplitude in the training set while tested against ten test sets, with a progressively increasing anomaly amplitude. At very low

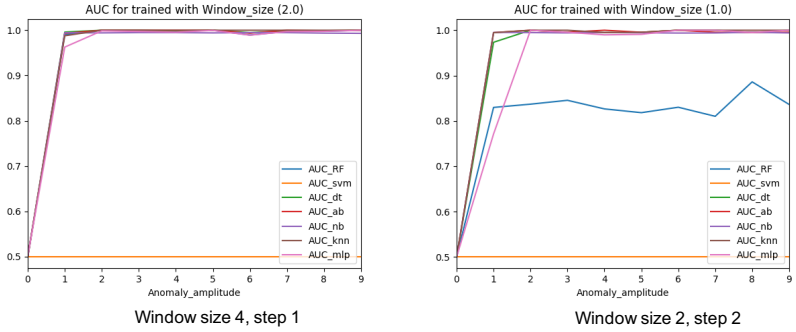


Figure 4.15: The sliding windows size/overlap of training compared to anomaly amplitude in test. Bigger window with higher overlap is better than smaller with less.

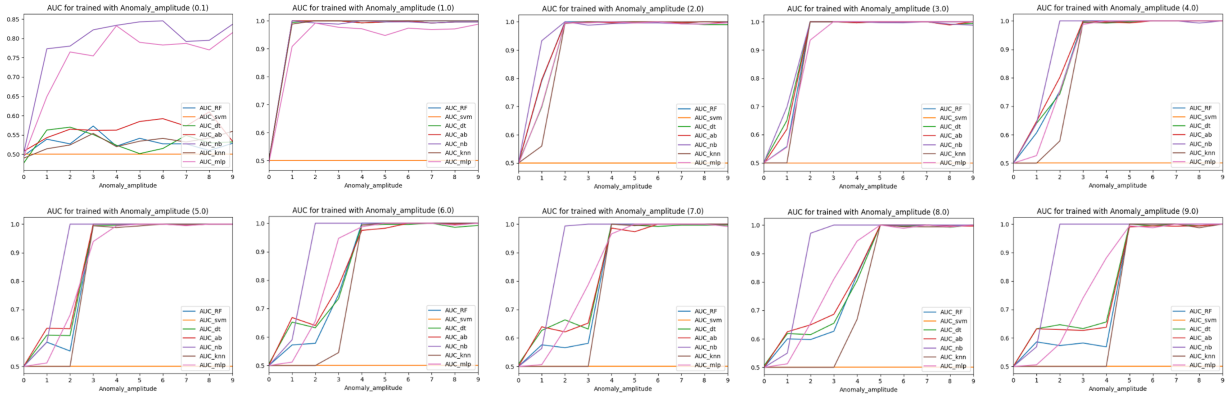


Figure 4.16: Top row, left to right, the models trained with a low anomaly amplitude in the training set were tested against ten test sets, with a progressively increasing anomaly amplitude. The results illustrate the effects of the anomaly amplitude, when data are not characterized by trends or seasonality. At very low amplitudes in the training set, the NB and MLP algorithms outperform the others.

amplitudes in the training set, the NB and MLP algorithms outperform the others.

Fig 4.17 shows a comparison of the performance when a model trained with no trends in the training set and tested against various concept drifts in data. The dashed line indicates the characteristics of the training set and shows that the highest performance is obtained at that point. The experimental results shows that a mechanism that selects model based on their historical performance is better that training a model with current data characteristics as in Blind Adaptation while using conventional machine learning algorithms. Contrary to the assumption that models generalize better to data characteristics that are similar to the training set, the highest performance was achieved with some of the data trends in the test sets that were missing in the training set.

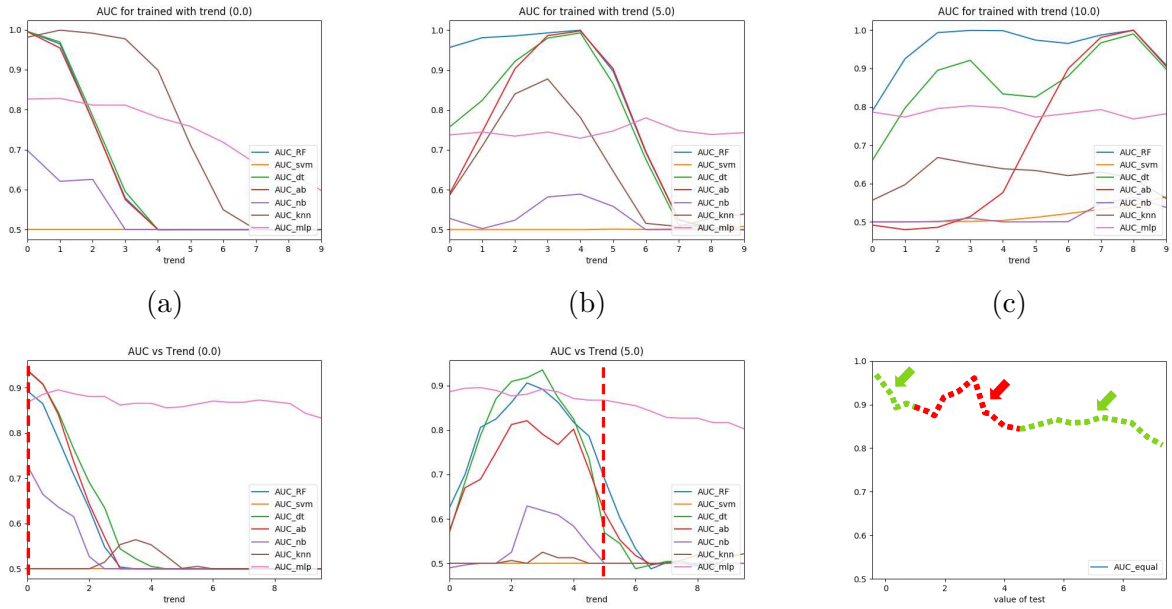


Figure 4.17: Comparison of the performance with concept drifts (a- c). A model trained with no trends in the training set and tested against various concept drifts in data. The dashed line indicates the characteristics of the training set and shows that the highest performance is obtained at that point (d). The experimental results confirm the necessity of meta-learning while using conventional machine learning algorithms. Contrary to the assumption that models generalize better to data characteristics that are similar to the training set, the highest performance was achieved with some of the data trends in the test sets that were missing in the training set (e). The overall upper bound of the performance of all trained models in their highest achieved generalization (f).

4.2 Baseline, Meta-learning and Deep-learning Models

The accumulated performance of all conventional models was 88.98% while 1D-CNN performed with 90.10% overall AUC.

Table 7.1 shows the 1D-CNN higher performance compared with Meta-Learning in concept drifting data. In more details shown in Fig. 4.18, among the conventional models, MLP has the most stable performance, although its average performance appears poor. The results show that CNNs can exceed the MLP and create robust models that perform well with any data characteristics. In the presence of data drifts, the CNN model appears superior to every other algorithm considered in our analysis, even when they are combined by means of a meta learning approach. Moreover, the CNN model requires less supervised pre-processing than conventional machine learning algorithms, and despite its superior generalization ability, the CNN model speeds up the learning process significantly. As Fig. 4.3 shows, the CNN model also has fewer hardware requisites for hyperparameter optimization, as default model architectures could generalize to all unseen data characteristics.

Table 4.2: Performance of Meta-Learning and Deep Learning approaches in terms of average AUC

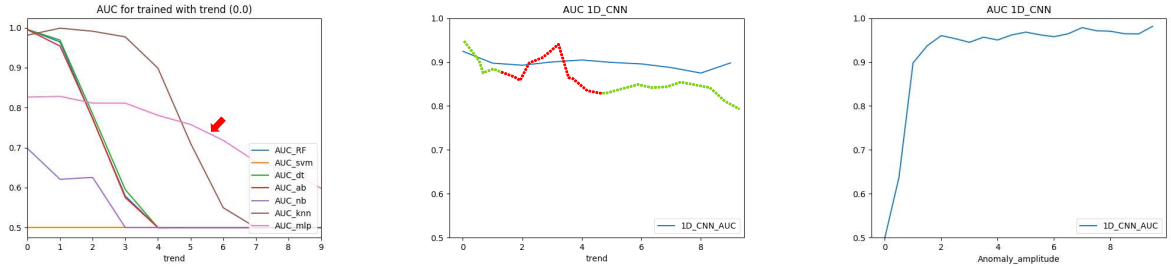
Method	AUC (Accumulated)
Meta-Learning	88.98%
Deep Learning (1D-CNN)	90.10%

Table 4.3: Performance of Neural Networks based techniques in the anomaly detection task.

Algorithm	AUC (max)	Optimization Time	Hardware used
MLP	82.72%	8620 sec	40*Xeon 2.5GHz
1D-CNN	93.09%	50 sec	GTX 970M

The results from comparing various baseline models and the proposed models are presented in this section including different Neural Networks (NNs) architectures when attempting to localize anomalies and patterns in sequential data with concept drift.

Conventional Neural Networks were compared with a deep learning CNN model when data characteristics were changing (see Table 4.3). Deep learning models were run on GPUs and the optimization of hyperparameters of conventional models including MLP was performed using a cluster of 40 CPUs. 1D-CNN consists of four convolutional layers, each of which is followed by a max pooling layer. The size of the layers inherently depends on the dimensionality of the data fed into the model, which varies according to the different experimental configurations considered.



(a) (b) (c)

Figure 4.18: (a) When data characteristics change MLP showed the most stable performance overall compared to other conventional models, but it showed little improvement in a meta-learning setup due to its moderate performance. (b-c) A deep learning approach can outperform overall average performance of all conventional models. 1D-CNN model shows highly stable level of performance in various data characteristic changes.

In Fig 4.18 when the data characteristics change MLP showed the most stable performance overall compared to other conventional models, but it showed little improvement in a meta-learning setup due to its moderate performance. (b-c) A deep learning approach can outperform overall average performance of all conventional models. 1D-CNN model shows highly stable level of performance in various data characteristic changes.

Table 4.4: Performance of Blind Adaptation, Meta-Learning and Deep Learning approaches in terms of average AUC

Method	Average of AUCs (Accumulated)
Blind Adaptation (12hr intervals) - Random Forest	63.33%
Blind Adaptation (12hr intervals) - SVM	58.14%
Blind Adaptation (12hr intervals) - DT	81.59%
Blind Adaptation (12hr intervals) - Adaimprove	82.91%
Blind Adaptation (12hr intervals) - Naive-Bayes	77.83%
Blind Adaptation (12hr intervals) - kNN	61.60%
Blind Adaptation (12hr intervals) - MLP	80.93%
Meta-Learning	88.98%
Deep Learning (1D-CNN)	90.10%

Table 4.5: Performance of Neural Networks based techniques in detecting anomalies.

	Performance		
Algorithm	AUC (max)	Optimization Time	Hardware
MLP	82.72%	8620 sec	40 * Xeon 2.5GHz
1D CNN	93.09%	50 sec	RTX 2080ti
2D CNN (Heatmap)	63.89%	123 sec	RTX 2080ti
2D CNN (Fourier)	73.04%	97 sec	RTX 2080ti
SMCNN (Heatmap)	**95.89**%	343 sec	RTX 2080ti
SMCNN (Fourier)	95.09%	298 sec	RTX 2080ti

We evaluated the performance of blind adaptation and meta learning and deep learning baseline models. The blind adaptation method was adjusted to run every 12 hours (data sampling ratio of 15 minutes). The meta-model was selecting top performing models based on their previous scores in similar data. The 1D-CNN deep model was trained once and had no retraining during the evaluation phase. The accumulated area under the curve (i.e., AUC of AUCs) over various data characteristics are presented in Table 4.4 which was added to the Result Chapter. Among conventional models, Adaimprove performed better than other, while second rank was Meta-learning and the top performing model was 1D-CNN model.

The results show that deep learning has a higher performance in detecting anomalies and it provides a stable detection quality for any data characteristics. A CNN-based pattern detection model feature engineering is much simpler and the model quicker to optimize compared to conventional models. Moreover, it exhibits a higher performance in terms of AUC than other classical approaches. The theoretical justification for the higher performance of CNNs compared to other algorithms, appears to be their inherent data abstraction and feature extraction capabilities. In fact, the compactness of CNNs convolutional layers enables the system to create a powerful representation of data, that results in an accurate predictive performance which is stable in diverse experimental conditions, including the presence of concept drift.

Table 4.6: Performance of anomaly detection performance (Maximum AUC) of proposed methods in various types of concept drift.

	Concept Drift Type			
Algorithm	Gradual	Incremental	Recurrent	Sudden
MLP	62.72%	60.03%	76.26%	58.99%
LSTM	92.60%	87.34%	**97.89**%	77.68%
1D CNN	93.09%	92.09%	96.21%	**82.77**%
2D CNN (Heatmap)	63.89%	58.02%	72.55%	62.12%
2D CNN (Fourier)	73.04%	62.02%	70.90%	69.92%
SMCNN (Heatmap)	95.89%	93.32%	97.20%	71.15%
SMCNN (Fourier)	95.09%	90.91%	89.83%	72.15%
SMCNN (APMS)	96.17%	90.11%	97.36%	72.26%
SMCNN (FAPMS)	**98.66**%	**92.72**%	96.24%	73.00%
DriftGAN (Heatmap)	56.74%	59.80%	61.38%	74.38%
DriftGAN (Fourier)	58.01%	62.68%	50.05%	68.44%
DriftGAN (APMS)	93.01%	92.35%	92.38%	67.98%
DriftGAN (FAPMS)	94.01%	90.12%	93.44%	68.82%

MLP maximum AUC was 82.72% while using a deep learning model improved the performance. 1D CNN performed with 93.09% which is a high AUC compared with conventional models. 2D CNN using Heatmap images as input achieved maximum AUC of only 63.89%, as well as 73.04% for Fourier transformed inputs. Our proposed SMCNN performed higher than all other models with 95.89% and 95.09% for the heat-map and Fourier images respectively.

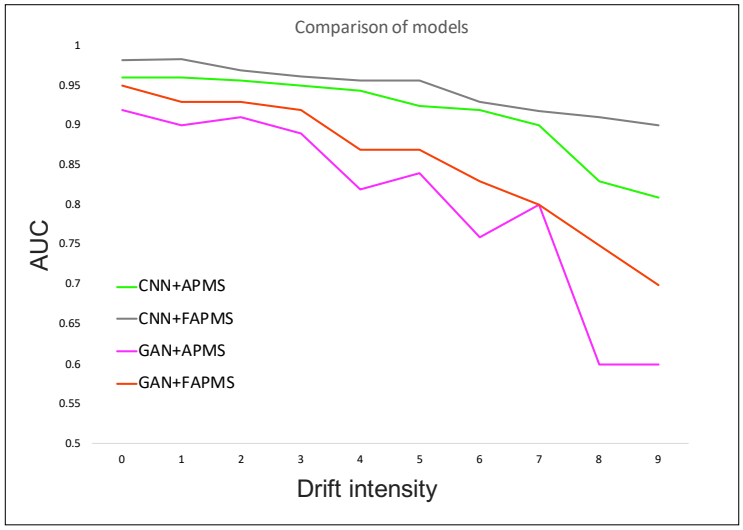


Figure 4.19: Supervised Vs. Unsupervised models using APMS and FAPMS algorithms.

A comprehensive comparison of model in various types of concept drift shows the maximum AUC achievable by each model in Table 4.6. In gradual and incremental types of

Table 4.7: Performance of anomaly detection in various stages of pattern existence in the data.

Algorithm	15% of pattern	50% of pattern	100% of pattern
1D CNN	72.92%	91.17%	93.09%
2D CNN	51.04%	59.73%	63.89%
SMCNN	95.83%	97.86%	98.66%
DriftGAN	89.71%	93.75%	94.01%

concept drifts, our proposed SMCNN method performed higher than the other models, while in recurrent concept drift LSTM performed best. In sudden concept drift 1D-CNN model performed better than the other models.

We performed an experiment to compare the performance of proposed models in detecting patterns in the three stages of the patterns appearance in the data. We use real cyber-attack long-term anomalous patterns as a benchmark and generated artificial concept drift and characteristic changes to evaluate the generalization capabilities of the models. The main goal is to detect the anomalous patterns as early as they start to appear. In our studied domain, we describe an early pattern detection when 10 to 20 percent of the footprint of the anomalous pattern has appeared; a mid-pattern detection when 45 to 65 percent is detected; and a full-pattern detection when the whole pattern is detected. The results of the new experiment of various stages of pattern detection is illustrated in the following Table. We generated full patterns and then labeled the first 15% (early) and 50% (middle) of the length and 100% (full) of the patterns and evaluate our models. Among deep learning models, SMCNN achieved the highest performance in the three phases of early and middle-pattern and full-pattern anomaly detection. Table 4.6 and its description were added to the Result chapter.

In terms of performance of models in detecting anomalous patterns in three phases of their appearance, SMCNN performed better than all other models followed by DriftGAN.

The results showed that deep learning achieved a higher performance in detecting anomalies and it provided a stable detection quality for any data characteristics. A CNN-based pattern detection model appears much simpler and quicker to train. Moreover, it exhibits a higher performance in terms of AUC than other classical approaches. The theoretical justification for the higher performance of CNNs compared to other algorithms, appears to be their inherent data abstraction and feature extraction capabilities. In fact, the compactness of CNNs convolutional layers enables the system to create a powerful representation of data, that results in an accurate predictive performance which is stable in diverse experimental conditions, including the presence of concept drift.

Chapter 5

Discussion

We performed two main groups of experiments: 1-conventional machine learning models, and 2-deep learning models. In the conventional models set, we used mainstream models and performed Grid Search over a wide range of selectable values with relatively small steps to find the maximum AUC each model can achieve. In the deep learning set we used the same number of parameters similar to all base models and evaluated them with similar data and concept drift.

In the conventional models, as each model was able to perform well in a very limited range of data characteristics, we propose a meta-learning model that selects high performing models among trained models based on their historical performance. The meta-model calculates the current data characteristics based on samples of data to selects the optimal model. The confidence interval granularity, therefore, depends on the sampling ratio of the meta-model to derive the results. One of the limitations of meta-models is their sensitivity to setting of the sampling ratio, which we will address in the deep learning part.

In the deep learning experiments on the other hand, our proposed models where able to provide relatively higher performance for a larger range of data characteristics, the cost however is the need for a lot more training data. In addition, the pixel-wise input of our transformation technique and SMCNN algorithm enabled the model to change the number of samples in each image without a need for changing the model. Using variation in the number of samples in inputs enables the model to search for patterns with various granularity without blindly sampling from data that might remove important information about the pattern especially rare anomalous instances. In the unsupervised deep learning setup, we proposed DriftGAN which performed relatively well in detecting novel patterns using our image transformation technique and AMPS/FAPMS optimization algorithm for selecting hyperparameters.

5.1 From Conventional Models to Meta-learning

We chose the main conventional machine algorithms to test their performance in the data described above and evaluate whether some models have superior performance to others. Although, all algorithms perform similarly on average over all data characteristics [124]. However, the distribution of the performances varies from algorithm to algorithm. While

the majority of models perform well in some characteristics and poorly in others, in our pilot study NNs displayed a mediocre performance in all characteristics. Fig 5.1 shows the performance of each algorithm in various data characteristics. As highlighted cells indicate, none of the algorithms outperforms the others in all cases. In particular for example, a higher trend results in a decreased performance, particularly for simpler algorithms such as NB. Notably, SVM, which generally shows a weaker performance than the other algorithms, scores the highest AUC when the data contain high trends. From the result of comparing probability of observing anomalies in training-set while changing the anomaly amplitude in test-set, we conclude that the higher number of anomalies in the training-set increases the performance of models. When there is 1 anomalous sliding window among 100 samples, almost all models performed well (except SVM). However, once the probability of observing an anomaly decreased to 1 anomaly in 1000 samples, then performance of models also decreased, especially the kNN model (see Fig. 4.7). Further, the anomaly amplitude from training-set to test-set affects the performance (see Fig 4.8). A very small anomaly amplitudes, dramatically drops performance. However, MLP and Naive Bayes still perform relatively well. Interestingly models show robustness to the slight concept drift. In Fig 4.8 for example, a model trained with a training-set that anomalies are high amplitude spikes (amplitude 10), starts to perform well almost when the spikes in test-set are around 60% of those in the training-set (amplitude 6). We also trained three models with 0%, 5% and 10% trends growth in year and tested against test-sets with a variety of anomaly amplitudes. From the plots we can also conclude that lower trend in training-set produces higher perfromance when the anomaly amplitude changes in the test-set (see Fig 4.9). We therefore suggest that, using a meta-learning mechanism, a framework can find the upper-bound performance of algorithms in detecting outliers and anomalies (see Table 7.1).

In the case of a single stream with changes in the forms of patterns (drifts appearing in only one of the data characteristics), none of the algorithms retains its performance, and each algorithm only performs best with a limited portion of the data. Thus, with drifts in the data, each algorithm could become sub-optimal. This results illustrates that there is a need for a model updating system when a drift occurs in one or more data streams. More importantly, we observe that a model trained with certain characteristics does not necessarily perform in its best conditions when the application data is relatively similar to the trained characteristics. In our experimental results, we observed that models trained with characteristics that are different to the trained characteristics had a higher performance. This result demonstrates the importance of using an informed adaptive mechanism (Meta-learning) that detects current characteristics of data to select the best performing models in a specific scenario without imposing a bias. Without meta-learning, eventually the hypothesis space of each individual learner increases very quickly and its performance becomes sub-optimal. Depending on the structure and configuration of the learners, models ultimately underfit or overfit the input data. To maintain the learner's optimality in case of a concept drift, especially in conventional models, the models have to be adapted according to the new input characteristics.

The size of training-set is one of the important factors to produce high performing models. In our experiment with network traffic anomaly detection, 500000 samples were required to produce high performances. When the data for training is limited, the highest performance belonged to Naive Bayes, followed by MLP. An approach for effective data

	Few Data	High Trend	High Amplitude Seasonality	Low Anomaly Probability	Large Window	Large Steps
Random Forest	51.97	67.13	92.02	99.0	96.99	50.93
SVM	50.0	**88.96**	50.0	50.0	50.0	50.0
DT-C4.5	59.79	66.57	87.23	99.0	78.57	**92.16**
Adaimprove	54.38	66.50	90.04	95.89	96.87	90.71
Naive Bayes	**92.58**	50.0	**94.27**	99.0	**98.98**	50.0
kNN	50.0	70.79	66.19	56.05	93.92	64.03
MLP	66.10	84.66	90.05	95.0	85.14	55.18

Figure 5.1: Each model only perform better in a certain data type and not in all types of data characteristics.

science, therefore, can be selecting models based on the size of training-set (see Figure 4.3). The size of sliding window is also another import factor that affects the performance of models. We evaluated various sizes (e.g., sizes 24, 48 and 96) and sliding steps (i.e., how many samples are overlapped between two consecutive window). Smaller sliding window sizes produce higher performance. Similarly the smaller strides (higher overlap) result in better performance. Also as Fig 4.10, Fig 4.14, and 4.12 show, the smaller window size and the more overlap in sliding window the better. When comparing window size or sliding steps to find which one is more effective to improve the performance, we found that higher overlap results in a higher performance improvement than the windows size (see Figure 4.1). The sliding windows size/overlap of training also compared with various trend levels in the test-set as shown in Fig. 4.6. A bigger window with higher overlap is better than smaller with less overlap, meaning that to increase the number of training samples, the overlap in sliding windows is more effective that choosing a smaller size sliding window. In addition, training a models when there is low amplitude of seasonality in the training-set, results in a higher performance compared to models that were trained on data with high amplitude of seasonality.Therefore, it could be beneficial to either use a data that has smoother seasonality or to remove seasonality of data by a normalization method, to enhance the anomaly detection capabilities of models. However, when there is an amplitude of 10 which is relevantly equal to the amplitude of the anomalies the performance decreases. Among conventional models, kNN was most sensitive model to the changes of seasonality amplitude, followed by the decision tree (see Figure 4.2).

In order to limiting the effect of hyperparameter selection, we performed Grid Search over a wide range of selectable values with relatively small steps. For example in SVM, we particularly performed a GridSearch, decision-function-shape and the result is based on the highest performance achievable. The Kernel choice also could be the cause of the low performance of SVM. We performed a GridSearch with linear, rbf, poly, sigmoid, and, in almost all cases, the performance of the rbf kernel was superior to that of the other SVM kernels. The SVM in our dataset detects everything as normal which could be because of the nature of our data that has difficult anomalous pattern footprints and high levels of noise that could be very similar to the anomalous spikes. To conclude the findings in conventional models, the Table 5.1 shows the average performance of seven conventional algorithms. The models were trained with data following a specific distribution, and were tested against data subject to concept drift. When data samples are scarce, the NB and

MLP algorithms performs much better than the others. Similarly in very low anomaly amplitudes in the training-set, NB and MLP outperform the other methods. The plots obtained by performing early detection as described in the Methodology Chapter, in pattern analysis and finding anomalies in sequential data. A similar high performance is achieved when there are large waves of seasonality in the data or the size of the sliding window that indicates the input sequence is very large. However, the NB algorithm performs equally to flipping a coin in the presence of high trends. The poor performance of the NB algorithm is also observable when the steps between the sliding window are relatively high. By contrast, the SVM algorithm, which did not perform better than merely guessing the labels by chance in almost all situations, outperforms all algorithms when the data contain many trends. When anomalous instances are scarce, the Random Forest, Decision Tree and NB algorithms show superior performances with respect to the other algorithms considered in the analysis. In addition, the more data samples is training set results in a better performance. However, higher trends and seasonality amplitudes in the data decrease the AUC. Nevertheless, the SVM algorithm start to outperform other algorithms in contexts with a high amount of trend in data. Additional experiments revealed that a higher probability of existing anomalies and related amplitudes in the data have positive effects on the AUC. We compared baseline methods, several conventional machine learning models in the first part of this thesis. We tested various combinations of data characteristics, preprocessing methods and perform grid-search for finding optimal hyper-parameter. The result suggest that none of the conventional models is able to outperforms others in all conditions. The result conform the NFL theorem that no statistical advantage exists between any two model when averaged across all domains.

$$\sum_{f} P(d_m^y | f, m, a_1) = \sum_{f} P(d_m^y | f, m, a_2) \tag{5.1}$$

In the NFL theorem shown in Eq. (5.1), d_m is an ordered set with size m and cost values y, a_1 and a_2 are arbitrary optimization algorithms of function f that run m times with m iterations. The NFL theorem describes that the summation of the conditional probabilities is always equal. Our empirical results conforms the NFLT that there is no conventional machine learning model that performs well in all data types and concept drift intensities (see Fig 5.1 .

In a broad type of application that produces sequences or streams of data, machine learning requires continuous adaptation to cope with the data changes over time. Providing models with high accuracy in detecting anomalies is generally a complex Nondeterministic Polynomial (NP) hard empirical process. Models have to be extensively replaced by other algorithms and optimized to avoid under-fitting when the input evolves to a more complex data. Nevertheless, however, as the data grows with time, the models tend to become more complex and eventually over-fit the data. Techniques to avoid the expansion of data to an explosion point are regularization techniques such as removing old instances or assigning a higher weight to the recent inputs, randomly setting neural network weights to 0 during training, and limiting the magnitude of the weights during training. Occasionally, in tasks such as anomaly detection, imbalanced data, or lifelong learning, removing or degradation of old data might not be admissible as the act removes previous rare but valuable instances. In such cases, a better solution is an adaptive model selection and

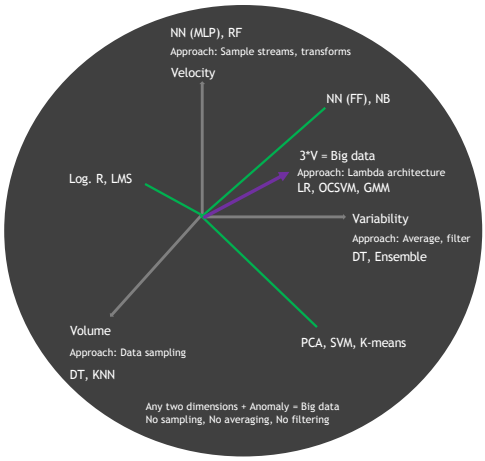


Figure 5.2: Machine learning models perform optimal only in a limited range of data types or domains. Our initial experiments showed that some models perform better than others depending on the characteristics of the data in terms of ratio of samples in a time period, different types of measurements, number of streams, or their combinations.

reconfiguration, which is formulated as meta-learning or learning to learn. Temporal data often exhibits cyclic patterns that frequently combine with trend and noise as they correlate to humans, machines, or environmental seasonal produced data. Anomalies are deviations from regular patterns of data profiles. Unexpected bursts in time-series data might indicate an engine failure in the context of the Internet of Things (IoT), an intrusion activity or cyber-attack in network traffic data, a heart-attack in ECG data, a record-breaking temperature in winter, etc. Detecting, localizing, and classifying various types of anomalies are important in many applications as they can alarm future failures, protect assets, or change the current path of progress. The real-time anomaly detection in large scale streams constructed by complex systems is an open research question. The rapid progression of Artificial Intelligence (AI) to a new variety of applications challenges of defining appropriate machine learning solutions in novel environments to leverage interactions with human experts and its associated expenses. Moreover, selecting optimal models and configuring hyper-parameters are also NP hard empirical processes, involving exhaustive searches of the entire hyper-parameter space. Commonly, several iterations of trial and evaluation are required to gradually achieve an optimal set-up. However, in temporal data, models become suboptimal as the data can shift drastically. Meta-Learning models have long been suggested in the context of complex hypothesis spaces and to reduce manual intervention. Rudimentary meta-learning models with algorithm ensembles could alleviate the bias and variance of individual models on static data sets. As data sets became dynamic, meta-learning addressed the issue of real-time model selection and auto-configuration through the use of a generalized representational schema. It has been shown that promising results can be provided using meta-learning as a mechanism to progressively describe the model's architecture and provide more effective and adaptive hyper-parameter optimization. A meta-learning approach can enable learning in environments that not only the learning is a black-box, but also the nature of data changes over time and between domains (see Fig 5.3.

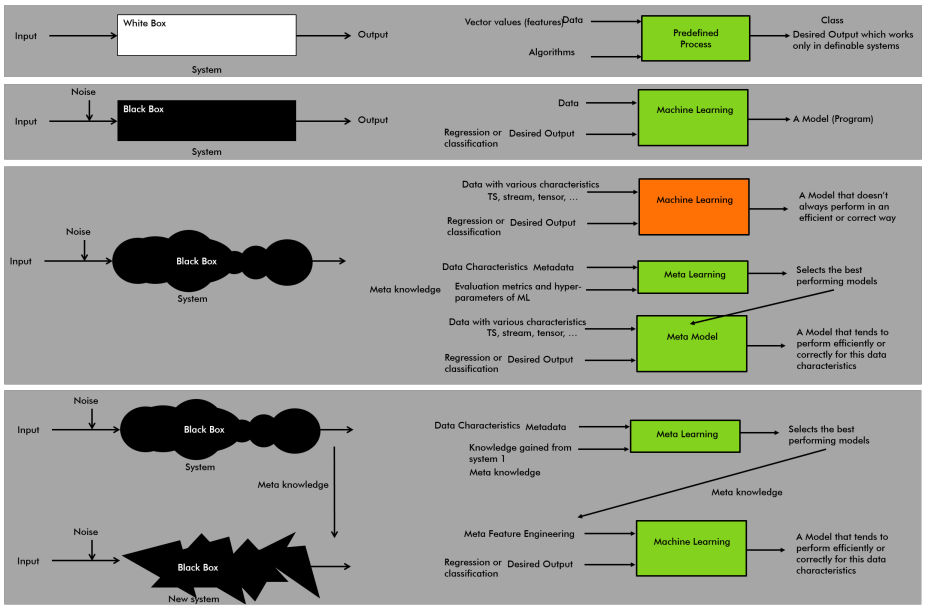
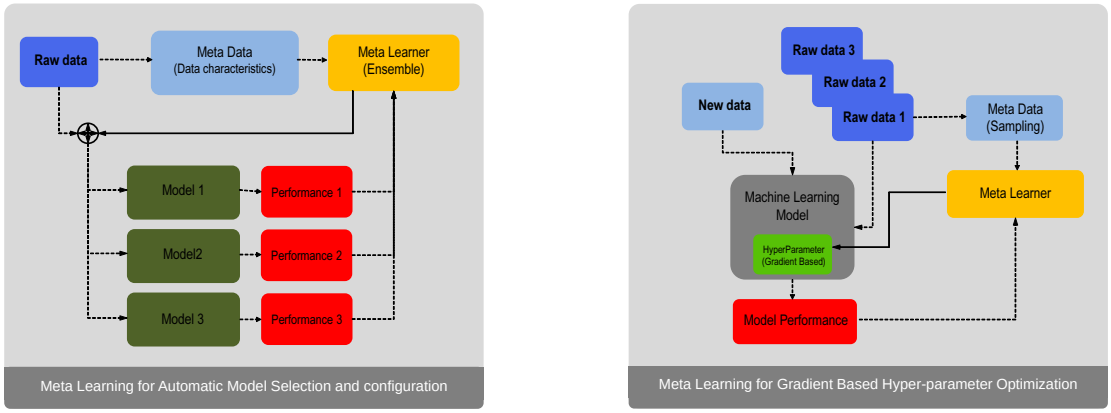


Figure 5.3: Comparison of white-box and black-box approach and the necessity of adaptive systems in complex and dynamic systems.

In Fig 4.17) the dashed line indicates the characteristics of the training set and shows that the highest performance is obtained at that point. The experimental results shows that a mechanism that selects model based on their historical performance is better that training a model with current data characteristics as in Blind Adaptation while using conventional machine learning algorithms. Contrary to the assumption that models generalize better to data characteristics that are similar to the training set, the highest performance was achieved with some of the data trends in the test sets that were missing in the training set. Our results show that, despite the usual assumption that a model is able to generalize better to data characteristics that are similar to the training set, the highest performance was achieved with fewer data trends in the test sets than in the training set. Therefore, our models do not necessarily perform in their best conditions on test sets that contain the same characteristics of training sets. Conventional machine learning algorithms require extensive pre-processing steps when they are used to classify sequential data. Using a model capable to adapt to a changing data distribution is of crucial importance. Since each model trained with a certain trend performs better with a certain type of data characteristics. The results also show the importance of a system that selects models and configures hyperparameters without imposing a bias. Our results show that, despite the usual assumption that a model is able to generalize better to data characteristics that are similar to the training set, the highest performance was achieved with fewer data trends in the test sets than in the training set. Therefore, our models do not necessarily perform in their best conditions on test sets that contain the same characteristics of training sets. Conventional machine learning algorithms require extensive pre-processing steps when they are used to classify sequential data.

Andrychowicz et al. [4] suggested that specialization to a subclass of problems is the only approach that allows to obtain a performance improvement in learning algorithms.

Nevertheless, in this study, we cover broader families of algorithms which fit more into the definition of meta-learning. Unlike in purely ensemble methods, such as bagging [14] and boosting [40], in which the idea is to improve the predictive accuracy by producing variations either in the data or in the algorithms, the core concept of meta-learning is to exploit knowledge acquired during the learning process. Meta-data abstraction creates an essential representation of information about the data that enables systems to work in a real-time manner. Automatic hyper-parametrization allows the reduction of data-science intervention. Evolution based model selection architecture optimizes the performance based on previous experiences for promising hypothesis spaces. Meta-knowledge transformation brings the knowledge gained from all other experiments and offers solutions to open questions about unsupervised learning (see Fig 5.4. The meta-learning method provides even better results by selecting most successful algorithm according to the data properties. Despite the fact that the first-difference estimator eliminates several important features of data and in several cases it is not a suitable preprocessing step, it still is not able to entirely boost the performance to the maximum in a higher percentage of trend or amplitude of seasonality. A comprehensive comparison was performed between the performance of meta-learning of machine learning algorithms and deep learning models in detecting anomalies. The findings confirm the necessity of applying meta-learning as an optimal strategy when using traditional models.



(a) (b)

Figure 5.4: In a meta-learning mechanism, several models with various performance are being selected based on current data characteristics and previous experience of each model performance (left). New meta-learning systems are designed for fine-tuning hyperparameters, often for few-shot learning using gradient-based optimization (right).

5.2 From Meta-learning to 1D Deep-learning

Conventional machine learning methods provide simple ways to represent a mapping between sliding window input sequence to the label. The sliding window size often remains constant throughout the training procedure and is estimated before training using trial and error along with some heuristics which requires injection of human expertise. In many sequences, however, fixed windows sizes do not capture the actual temporal locations of

the time dependencies. Finding optimal models and hyperparameters are often default and time consuming during which the current models in use might remain suboptimal. Using meta-learning to find optimal configurations requires an extra processing level to train meta-models. In addition, one of the limitations of meta-models is their sensitivity to setting of the sampling ratio. Also, a meta-learning in a new setup requires quiet amount of training history to learn the behaviour of various models. Deep learning algorithms, on the other hand, are able to overcome the model weakness of traditional ANNs in presence of variations in patterns, anomalies and data representations. Generally, deep learning algorithms provide learning strategies that outperform conventional machine learning algorithms [73], [77]. We compared a meta-learning that can find upper-bound of best performances in real-time and a one-dimensional convolutional neural network. Overall, 1D-CNNs show a better abstraction capability that preserves the features of the input data representation. Essentially, this consideration means that the overall performance of CNNs for a large amount of data types appear higher than that of any pure conventional or meta-learning of conventional algorithm. However, we conjecture that any model of high complexity, such as deep learning models, would exhibit poor performance without sufficient training data.

As shown in Fig. 4.18.(a), among the conventional models, MLP had the most stable performance, although its average performance appears poor. MLP performed constantly average, not similar to the other methods that performed very poor when the concept drift is very intense. Now the question is that if we use deeper neural network with abstraction layers as we describe in deep learning, would it improve the performance in each test while the total generalization capabilities remain the same or even improved and if the model can overcome the average performance of the regular neural networks. we used various models to evaluate how each neural network based model performs in drifting data including Multi-Layer Percetron (MLP), Long Short-Term Memory (LSTM), and one-dimensional (1D-CNN). Deep learning creates high-level abstractions of data using architectures that consist of multiple transformation layers. This process is seems to be extremely effective for our data with high amount of noise. While older approaches such as TDNNs, introduced in 1988, are simply represent mappings between past and present values with constant delays throughout the training procedure that require to be estimated before training, deep learning provides a fully automatic abstraction method. Other methods such as TCNNs require trial and error along with some heuristics which need injection of human expertise and the fixed delays might not capture the actual temporal locations of the time dependencies, 1D-CNN required almost no pre-configurations.

Methods based on NNs and tailored for sequential data in form of time series include Recurrent Neural Networks (RNNs), which add the notion of order to instances and solve many tasks that cannot be solved by feed-forward networks, which use fixed size time windows. Long Short-Term Memory Neural Networks (LSTMs) [61], [106], a particular type of RNNs, obviate the need for specified time windows and are capable of accurately modeling complex multivariate sequences characterized by long-term dependencies [81]. However, the most significant limitation of LSTMs is their vulnerability to pattern variance in terms of observation sequence and in locating the target objects [110], [5]. Deep learning particularly CNNs have become popular for their abstraction power and location invariance. CNNs produce high-level features by automatically learning the values of filters. CNNs

are fast and efficient in terms of representation as filters learn automatically and creates a system with ability of detecting objects regardless of their positions. The architecture consists of several layers of convolutions often with non-linear activation functions (e.g., tanh, ReLU, etc.) and the last classifier layer. Furthermore, multiplying $n \times n$ filters, using different or equal weights, and sub-sampling of maximums (max-pooling) creates high-level abstractions with more generalization. Pooling layers provide fixed size output matrices required for classification and keep the most salient information, regardless of the size of filters or input. The pooling layer also provides invariance to shifting and rotation. In addition, the anomalies or objects in corners of the image or edge elements can be detected by adding zero padding and when the filter is relatively large comparing to the input size. Finally, stride size defines the filter shift at each step, larger strides lead to fewer applications of the filter and a smaller output size.

The memory structure of RNN can capture time information by learning dependencies, while the problem with RNNs is that they are impractical to use when trained with traditional techniques (e.g., Backpropagation through time) for learning long term dependencies. This problem arises from the so-called exploding/vanishing gradient which basically means that as the model propagates the error signals backwards through the networks structure they tend to vanish or explode. More advanced recurrent structures (e.g., LSTM) have properties that mitigate the exploding/vanishing gradient problem and can learn long term dependencies and are particularly suited for learning sequential data with long term dependencies. We used LSTM as a baseline model to compare its performance with our proposed models. Traditionally, RNNs and LTMSs are first choice for speech recognition and natural language processing (NLP) because the sequential structure [26] [55]. However, some studies have shown that CNNs can perform much better on these problems due to their location invariance and abstraction capabilities [1], [64], [68], [76], [112], [125]. In our experiment 1D-CNN performed better than MLP and LSTM. While 1D-CNN might be more computationally complex, but the parameters are optimized automatically and have more generalization capabilities. A 1-D CNN with simple hyperparameters (e.g., stride = 1) can perform the same by making the 1-D convolution window wide enough to include the longest delay inside the window and setting all weights for inputs neurons that do not have delay taps into zero with a certain window size. However, higher dimensional CNNs that we introduce in this thesis have higher feature extraction capabilities and are able to tolerate variations in the input representation. In general, 1D-CNN provides improvements over all conventional, meta-learning and baseline deep learning methods, which could be because of 1D-CNNs abstraction power and location invariance over other methods compared with. 1D-CNNs produce high-level features by automatically learning the values of filters. The architecture includes several layers of convolutions often with non-linear activation functions and a last classifier layer. The limitation of 1D-CNN, however, is the requirement for high amount of training data. The results show that 1D-CNNs can exceed the MLP and create robust models that perform well with any data characteristics. In the presence of data drifts, the 1D-CNN model appear superior to every other algorithm considered in our analysis, even when they are combined by means of a meta learning approach. Moreover, the CNN model requires less supervised pre-processing than conventional machine learning algorithms, and despite its superior generalization ability, the CNN model speeds up the learning process significantly. As Fig. 6.2 shows, the 1D-CNN model also has fewer

hardware requisites for hyperparameter optimization, as default model architectures could generalize to all unseen data characteristics.

A 1D-CNN can be divided into two types of layers. The first is a convolution layer that could be defined as multiple filters sliding across the image and mapping to outputs. But theses filters are not defined prior to the task and the 1D-CNN automatically finds optimal filter values that produce representation that can be learned by the second layer of fully connected neural networks, which is more than a simple set of filters. A 1D-CNN-based pattern detection model appears much simpler and quicker to train. Moreover, it exhibits a higher performance in terms of AUC than other classical approaches. The theoretical justification for the higher performance of 1D-CNNs compared to other algorithms, appears to be their inherent data abstraction and feature extraction capabilities. In fact, the compactness of 1D-CNNs convolutional layers enables the system to create a powerful representation of data, that results in an accurate predictive performance which is stable in diverse experimental conditions, including the presence of concept drift. In more details shown in Fig. 4.18, among the conventional models, MLP has the most stable performance, although its average performance appears poor. The results show that CNNs can exceed the MLP and create robust models that perform well with any data characteristics. In the presence of data drifts, the 1D-CNN model appears superior to every other algorithm considered in our analysis, even when they are combined by means of a meta learning approach. Moreover, the 1D-CNN model requires less supervised pre-processing than conventional machine learning algorithms, and despite its superior generalization ability, the 1D-CNN model speeds up the learning process significantly. The CNN model also has fewer hardware requisites for hyperparameter optimization, as default model architectures could generalize to all unseen data characteristics (see Table 4.3). Deep learning models were run on GPUs and the optimization of hyperparameters of conventional models including MLP was performed using a cluster of 40 CPUs. 1D-CNN consists of four convolutional layers, each of which is followed by a max pooling layer. The size of the layers inherently depends on the dimensionality of the data fed into the model, which varies according to the different experimental configurations considered. These advantages make 1D-CNNs a suitable candidate for real-time pattern and anomaly detection in temporal data as they avoid the learner's hypothesis space to become very large over-time. First, the neural network analog architecture creates a spectrum function from input to output which differs from many algorithms with rigid thresholds. And second, the abstraction of input created by layers of Deep Learning, disentangles the details of input that can be considered as noise and creates a simple representation that escape the overfitting of the learner.

5.3 From 1D Data to 2D Representation in SMCNN and DriftGAN

Our result suggests that one-dimensional Convolutional Neural Networks provide an optimal solution for anomaly detection in temporal data as long as the localization of anomaly is not critical. The maximum resolution of detected anomalies in 1D-CNN is equal to concatenated slices. We propose a transformation technique that enables 2-D CNNs to

handle sequential data. In order to localize the exact boundaries of anomalies, tensors are constructed from time-series data into two-dimensional images format. Moreover, anomaly detection is improved by intuitively adapting state-of-the-art image detection that significantly outperforms prior endeavors. Our solution creates a two dimensional representation of time-series data that could be used for 2D-CNN as tensors of images. 2D-CNN has been gaining significant grounds in areas such as visual object detection and speech recognition, benefiting mostly from the use of CNNs. However, the deep learning revolution has not had the same impact in the realm of general sequential or time series data. Further,we showed that anomalies are often sparse, therefore, as we showed a generic 2D-CNN produce low performance and is unable to pinpoint the exact location of the objects.

A 1D CNN can generally be used in sequence datasets for extracting local 1D subsequences from the input sequences and identify local patterns within the window of convolution. Conversely, a 2D CNN architectures mostly is used to learn image data with convolutional filters that move in two (x,y) directions to calculate low dimensional features from the raw image data and output a 2-dimensional matrix. However, the convolutional neural network with a colored image inputs can be imagined as a 3D input with an extra dimension for the colours, but in practice the method consists of a 2-dimensional model that runs 3 times, once for each R, G, B color channel. This extra information source enables a colored image convolutional neural networks to be able to compute more sophisticates features by training filters that process R, G, B channels in parallel. In addition each channel can provide more granularity for features such as edges, shapes, textures and also semantic correlations between channels helping for better separation of anomalies from normal patterns. The convolution layers of a CNN are filters with learnable parameter that have the same property to a sliding window over the input data. The convolution layers produce a weighted sum known as feature space that inputs the next layers. The feature space preserves the spatial and positional relationships between the input data patterns while extracting lower-dimensional features. Moving deeper in convolutional layers they learn key points in the input image and the edges and shapes, and finally the patterns that define if an image contains anomaly. CNNs efficiently reduce the raw data to lower dimensional space still representing the information about points and pixels in first few layer and edges and shapes in middle layers, and finally reduce to classify patterns in the images. Convolutional neural networks exploits the spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers.

The interactions between individual one-dimensional data streams emerge at the higher level and shape a complex data that requires to be studied as a whole. Therefore, the data often needs to be investigated in low level as univariate data streams of individual sources, and in high level as multivariate parallel streams. A 2D representation provides univariate and multivariate analysis with the same methods, and only by altering the input mapping method. As mentioned, the problem with 1-D CNN similar to other sliding window methods is the limited precision of pattern detection. The label can only indicate if a pattern or anomaly exist in a slice of time or not but does not provide enough granularity to explicitly identify the target instances. The dilemma is that larger sliding window can fit a long-term pattern while negatively affects the granularity of the detection and smaller sliding window sizes with higher precision do not capture the actual temporal locations of the time dependencies. A 2-D CNN can localize the target pattern but are bot

compatible with sequential data. Although, baseline generic 2D-CNNs perform poor in anomaly detection. 2D-CNNs, which exploit rigid boundaries to surround the anomalies, must create rectangles that are too large to allow the method to pinpoint the individual anomalies. This prevents the approach from generalizing well in problems of nature in which present a very large class imbalance between normal patterns and anomalies. The major issues arise when applying 2D-CNNs to sequential data, because the method can only accept inputs of a fixed size and can only detect one type of target (anomaly or pattern) per scan in each slice. In the case of sparse targets, a common property of anomaly detection in time series data, 2D-CNN creates a large bound that includes everything that falls between the anomalous patterns. This behavior introduces noise and decreases the detection performance by producing high numbers of false alarms.

We adopted state-of-the-art feature masking R-CNN method for pattern and anomaly detection that has the highest capabilities. It uses the special structure of the underlying CNN to find a very precise border around the pattern in the image. Using our proposed transformation technique and SMCNN we showed that not only the performance could be improved, but also the method explicitly segments data to multiple target objects. The methods also generalize to all type of sequential data, since raw inputs have become images made of pixels. Our method was shown to significantly outperform prior endeavours, and to generalize to a wide variety of sequential data. We evaluated various models including baseline two-dimensional model and proposed Sequential-masked-CNN and DriftGAN which were receiving heat-map and Fourier-transformed inputs. The SMCNN using the heat-map provided the best results in the test. The SMCNN method achieved the highest anomaly detection performance using heat-map input and simultaneously defined boundaries of every individual anomaly without explicit training labels. The advantage of using heat-maps compared to Fourier transformed images, despite less processing, is that the image can be directly mapped back to the input data representation and localize the object in the raw data.

The problem with regular 2-D CNN object detection method is their low performance in detecting anomalies with scarce footprint. We introduced SMCNN and DriftGAN that can mitigate this problem with a flexible masking mechanisms. Further, DriftGAN is an unsupervised anomaly detection is sequential data with long-term time dependencies. SMCNN and DriftGAN are fast and efficient in terms of representation as filters are learned automatically. Padding (wide convolution) can be used, when the filter is relatively large compared to the input size. Applying sliding window functions (e.g., kernel, filter, etc.) to the input matrix often for each channel (i.e., RGB or embedding) results in a system with the ability of detecting anomalies regardless to their positions. Multiplying an n*n filter, adding with different or equal weights, and choosing maximums results in achieving compositionality and high-level abstraction with more generalization. Also, edge elements can be detected by adding zero padding. Stride size defines the filter shift at each step, where larger stride sizes lead to fewer applications of the filter and a smaller output size. Generally, pooling layers apply, after the convolutional layers, to sub-sample their input. The common pooling operation is max-pooling which could pool over the complete output or a window. Pooling provides a fixed size output matrix required for classification, while keeping the most salient information regardless of the size of filters or input. Pooling provides invariance to shifting and rotation and allows use of variable size windows, sentences,

etc.

Among deep learning models, SMCNN achieved the highest performance in the three phases of early and middle-pattern and full-pattern anomaly detection. As shown in Figure 6.8 the results has a significant increase compared to generic two dimensional CNN. The interesting results in which the model not only detecting the anomaly but also pinpointing the exact location of the anomalies. The intuitive results indicates how effective the architecture is in detecting several objects in parallel both on heat-map and Fourier transformed images (see Figure Figure 6.8). The masking CNN creates flexible bonds that can explicitly surround anomalies on a heat-map or Fourier transformed representation of time-series data. The anomalies are detected as horizontal bounds.

We attribute the success of our methods to the fact that they pinpoints the exact boundaries of multiple target objects on the image representation. Further, SMCNN approach greatly reduces the requirement for labels to just one label for several window slices that are aligned on y-axis, instead of one label for each window slice, while DriftGAN can detect anomalies in an unsupervised setup. Benefiting from the transformations through layers of deep learning has made CNNs completely location and representation invariant, resulting in the ability to detect objects despite the order or location of their occurrences. Specifically, CNNs produce high-level features by automatically learning filters values. SMCNN and DriftGAN also provide intuitive methods that can be visualized and be interpreted by a human or an AI-based method. Our data representation algorithm enabled the systems to process multi-dimensional streams of time-series data in near real-time manner and improve the computational complexity from O(t*n2) to O(nlog(m)). Similar to One Class classifiers, DriftGAN requires clean data for the training and then would be able to detect multiple novel patterns simultaneously. The regular GAN methods produce dramatically low detection performance which is motivated by the high number of False Positives in concept drift. DriftGAN architecture can also consume AnoGAN [100] and BiGAN [129] models and enhance them to work in concept drift.

5.3.1 Discussion of transformation and optimization techniques

A sequence is a one-dimensional enumerated collection of possibly infinite number of elements and in form of time series has indexes equally spaced in time. A sliding window with size n reads the n samples of the data starting from sample with index s and maps the values from sequential 1D to the first row of the 2D image. the sliding window then move n sample forward to the next n values (without any overlap) and maps the next n values under the first row and continues until reaches the maximum number of rows r in each image. This means that the total number scanned points in the first image is $n * r$ while skipped s samples. The next images therefore starts from $(s + n * r)$ sample and continues based on the above description. Optionally to increase the number of images there can be an overlap between scans, which means that the next starting point of the image can be adjusted manually so it also cavers part of the previously scanned data.

Example data: 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0

If the target corresponds to detecting the pattern: 1 0 0 0 1
Using a sliding window with size 4 which move one instance each time, produces:

0010, 0100, 1000, 0001, 0010, 0100, 1000, 0001, 0010, 0100, 1000, 0001, 0010

And as we can see there is no 1 0 0 0 1

Now if we use our method with sliding window size 4, we get the following (note: while one might argue that in the previous method, the input could be size 5 and that would have resolved the issue, the challenge is that we cannot choose a window size that for sure contains the pattern).
0 0 1 0
0 0 1 0
0 0 1 0
0 0 1 0

The advantages in particular appear if there are repetitive normal cycles in the data for example in the following example. As highlighted in red, the anomaly stands out that could be detected more effectively.

0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 -> Anomaly

0 0 1 0
0 0 1 0
0 0 1 0
1 0 1 0

One of the other advantages of the method is the ability for the pattern to stand out in some types of concept drift such as incremental and gradual concepts drifts, as illustrated in the following example.

1 1 2 1 2 2 3 2 3 3 4 3 4 4 5 4

The pattern is highlighted that create an edge that is easier to be detected by a CNN model.
1 1 2 1
2 2 3 2
3 3 4 3
4 4 5 4

To transform the data to a 2D image a set of three hyperparameters need to be selected prior to the machine learning task. 1-a sliding window that maps each scan into a new row, and 2- the beginning point of the slices and, 3-the total number of rows in each image. Selecting optimal hyperparameters that would match to the short and long term repetitive patterns in data is difficult. We introduce APMS algorithm that selects a sliding window size, its starting point and the number of rows in each image by optimizing the average difference of pixels for a set of images. We also introduces FAPMS which can have various sizes of sliding windows that adapt to the average of repetitive patterns and is can

match cyclic patterns with different sizes more accurately. For example a heartbeat pulse signal data can have different lengths in normal, sleep or exercise conditions and a fixed size sliding window can be suboptimal. FAPMS provides flexible sliding window sizes that match different shapes of repetitive patterns of the fly.

Compared to the common sliding window method, selecting an optimal set of transformation parameters for our image based method can be quite challenging. Therefore, we introduced two algorithms that provide the optimal parameters for the user. Based on repetitive cycles in the data, our APMS algorithm finds the parameters for a fixed sized scanning window and its beginning point by calculating the cycles of part of the data defined by the user. FAMPS on the other hand provides flexible scanning window sizes that adapt to the lengths of the cycles. In a simple few digits scenario, the algorithms work as follows.

APMS: After defining maximum and minimum number of pixel in each image, the APMS algorithm generates various scanning window sizes and their initial points and transforms them to 2D (images). For example, assume the data is a stream of above pattern (0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 . . .), the algorithm generates the scanning windows as follows.

Tr-a 001000, 100010, 001000, 100010, 001000, 100010, 001000
Tr-b 01000, 10001, 00010, 00100, 01000, 1000, 00010
Tr-c 00100, 01000, 10001, 00010, 00100, 01000, 10001
Tr-d 0010, 0010, 0010, 0010, 0010, 0010, 0010
Tr-e 0100, 0100, 0100, 0100, 0100, 0100, 0100
Tr-f 001, 000, 100, 010, 001, 000, 100
Tr-g 010, 001, 000, 100, 010, 001, 000
Tr-h 00, 10, 00, 10, 00, 10, 00
Tr-i 01, 00, 01, 00, 01, 00, 01

And transforms the scanned windows to 2D (images) with various numbers of rows in each set of images, therefore we get:
Tr-a:
001000
100010
001000
100010
001000
100010
001000

Tr-b:
01000
10001
00010
00100
01000
10001

00010, And continuing in the same way.

FAMPS: The FAMPS algorithm finds optimal image parameters based on continuous evaluation of various size scanning window sizes, by a defined maximum and minimum numbers of pixel in each image. The FAPMS algorithm randomly generates various scanning window sizes and their initial points. Compared to AMPS the flexible scanning windows will be suboptimal if the patterns are repeating in identical cycles, but for domains with various lengths cyclic data can increase the performance, assume this data (000010100100000100000100100000 1 . . .) see the below the example.

Tr-a 0000, 101001, 000, 0010, 01000, 001000, 001
Tr-b 00 001, 010010, 00 01000, 001001, 000001
Tr-c 00001, 01, 001, 000001, 000001, 001, 000001
Tr-d 0000, 10, 100, 100000, 100000, 100, 1000001
Tr-e 0010, 10, 01000, 001000, 00100, 10010, 00001

Tr-a:
0000
101001
000
0010
01000
001000
001

Tr-b:
00
001
010010
00
01000
001001
000001, And continuing in the same way.

The FAPMS algorithm subtracts every images from the previous generated image and calculates the total summation of pixels in all images. The lowest generated total summation defines the optimal images in an unsupervised setup. Alternatively in a supervised learning setup, the optimal parameters are derived from optimizing the loss function using various generated images. The advantages of FAMPS only appear in data with various length cycles, and in data with equal length cycles can results in sub-optimal parameters and increase the computational complexity.

An L defined by user sets the length of data that will be used to evaluate the optimal parameters. The APMS algorithm subtracts every images from the previous generated image and calculates the total summation of pixels in all images. The lowest generated total summation defines the optimal images in an unsupervised setup. Alternatively in

a supervised learning setup, the optimal parameters are derived from optimizing the loss function using various generated images.

The image representation benefit is that if the width of the image fits the cyclic repetitions in the data (which can be optimized using our proposed APMS and FAPMS algorithms) then the target pattern stands out. In addition, the method enables object detection deep learning models to transform a 1-dimensional data to the image domain and detect small variations of patterns and anomalies within a large section of data more with higher accuracy. Our proposed SMCNN method is able to pinpoint the exact target instances in an image. In the opposite way, if reducing an image to row by row data, then the object in the image could be much harder to detect as relation between edges of the object become dispersed, since the spatial relation between repetitive patterns are removed. Aside from enabling CNNs that are designed for object detection in images become applicable to 1D data, the stacking of cycles in the data helps the convolutional layer to learn key points in the input image more efficiently. Our method transforms the repetitive time dependent patterns in the raw data to from edges and shapes and enable the convolutional layers to learn these shapes as normal data and learn distinctive anomalous patterns more easily. As convolutional layers are effective in detecting changes in the spatial representation of the input images, in as unsupervised setup, any deviation from the cyclic patterns can be identified with a high accuracy.

As described in our APMS (Automatic Pattern Matching and Sliding) algorithm the beginning point and axis sizes of the slides are parameters that will be optimized by our algorithms (real-time sliding window sizes for FAPMS). Instead of using a predefined fixed sliding windows input size approach that requires injection of human expertise, APMS and FAPMS algorithms find optimal sliding window sizes, their starting point and the number of rows in each image by optimizing the average difference of pixels for a set of images. APMS runs once at the beginning of the learning and FAPMS runs in equal time intervals to adapt the images with current repetitive patterns in the data. Using our algorithms we can automatically find sizes and starting points that match the repetitive cycles in the data and helps the machine learning models to perform better than random selection. In particular the FAPMS algorithm can set various sizes of sliding windows adapting to the current average of repetitive patterns to match cyclic patterns with dynamic length more accurately. For example in a heartbeat ECG signal data each pulse can have different lengths in various conditions such in normal, sleep or exercise mode and a fixed size sliding window can be suboptimal. FAPMS provides flexible sliding window sizes that match different shapes of repetitive patterns of the fly.

5.4 Designed Evaluation Metrics

A metric that reflects ratios of anomalous and normal instances is crucial for precise comparison of models in imbalanced data. In imbalanced data, the minority class could generate more important insights than the majority class. The standard evaluation metrics are not helpful as they treat all classes as equally important. In addition, we needed an informative metric but simple enough to compare various models while hyperparameters and data characteristics change. As the intensity of concept drift from training-set to the test-

set increases, models tend to start dropping their performance. Therefore, we required the metric representing the highest performance of models on various levels of concept drifts.

In general, metrics can be categorized in three types [58]:
1- The threshold metrics quantify the prediction errors by calculating the matching ratio between predicted class and the expected class in the test-set. Common threshold-based metrics such as Accuracy can lead to sub-optimal classification models in imbalanced data. It might produce misleading conclusions since these measures are insensitive to skewed domains [11]. Metrics such as Kappa, sensitivity-specificity, precision-recall, Macro-Average Accuracy, Balanced Accuracy, Mean-Class-Weighted Accuracy, Optimized Precision and Adjusted Geometric Mean are modified to be able to deal with imbalanced classification. However, they focus on one class and often assume that the class distribution in the training dataset will match the distribution in the test-set [58]. In the problem of detecting anomalies in concept drift, however, this is not the case.

2- The probabilistic metrics quantify the model uncertainty in predictions based on confident wrong predictions. Common metrics for evaluating predicted probabilities in balanced data are Brier score and cross-entropy (log loss) that average the difference between classes probability distributions. However, probability scoring methods are used less commonly in skewed class distributions due to their limitations, such as requiring accurate calibrations [36].

3- The ranking metrics, such as receiver operating characteristics (ROC) and AUC, evaluate classifiers based on how effective they are at separating classes[11]. Instead of a simple positive or negative prediction, the score introduces a level of granularity [35]. A ROC curve is a set of false positive and true positive predictions ratios in different thresholds. A zero-performing classifier (e.g., predicts all instances as normal) with a score of 0.5 is represented by a diagonal line from the bottom left to the top right on a plot, whereas a perfect classifier has a score of 1. The area under the curve (AUC) provides a single score to summarize the several ROC plots that can be used to compare models. An alternative to AUC is the precision-recall curve which only focuses on the positive class. Therefore, it is not suitable for anomaly detection, while AUC can find the optimal point of both normal and anomalous classes [35].

Kappa statistic shows how a classifier is performing compared to a classifier that simply guesses at random, according to the frequency of each class. Kappa and AUC are both main metrics for imbalanced datasets, however, various studies such as Delgado, et al. [32] and McPherson, et al. [84] showed that Kappa's behaviour fluctuates in comparing classifiers and should be avoided in anomaly detection.

Delgado et al. [32] state that: "Kappa is a relative measure of agreement which can be problematic since it is hard to set a threshold for a good agreement. This does not seem to be a problem when it is used as a performance metric in balance data, because Kappa values are compared for each classifier. Notwithstanding, we have shown that if marginal Kappa, to the extent that worse classification results can obtain, however, higher values of the statistic. This is especially dramatic when the entropy of the elements outside the main diagonal of the confusion matrix decreases to zero. Nowadays, in the field of machine learning such situations, in which the number of observations of one of the classes far exceed the quantity of the others, or when the marginal distributions are small, are

very common. Machine learning algorithms automatically scrutinize huge amount of data, classifying it into hundreds of categories or look for an unlikely relevant event. In that framework, the finding of a dependable performance measure to be robust and reliable becomes of the utmost importance. Hence, we believe that it has been sufficiently justified that, unfortunately, Kappa can no longer play this role. “

McPherson et al. [84] state that: “We first followed conventional modelling practice to illustrate how range size might influence model performance. We focused on two measures of accuracy, Cohen’s kappa and AUC of ROC plots. We then demonstrated that this influence is primarily artefactual. Statistical artefacts can arise during model assessment, because Cohen’s kappa responds systematically to changes in prevalence. In anomaly detection AUC, in contrast, is largely unaffected, and thus a more reliable measure of model performance.”

AUC score is an informative metrics when evaluating models on individual datasets. However, we aimed to compare models when the data characteristics progressively changed. Further, the maximum AUC of a model is the point that the True Positive and False Positive rates are in their optimal point. This score indicates the highest performance of a model. With calculating the maximum AUC for each model and increasing the intensity of concept drift, we plotted the overall AUCs curves that can be used to compare robustness of various models. The phrase “Simplified AUC” is used to explain two types of metrics we introduced in this thesis. 1-maximum AUC in which the area under the curve reaches its maximal point. 2-Accumulated AUC in which the area under the curve of all maximum AUCs is calculated. Using these two metrics we can demonstrate the performance of models by simultaneously comparing models in our complex experimental setup.

5.5 Limitations of Proposed Methods

The conventional models are often significantly more efficient compared to deep learning models. While deep learning models are both data-invasive and process-invasive, conventional models require much fewer data and lower processing powers to train which can be an important factor in large scale setups. The general limitations when working with deep learning models is requiring high number of data to train several layers of the models. In many real-world application providing a large labeled and cleaned training set is very expensive, if not impossible. The problem becomes even more acute in highly dynamic environments that the labeled data become outdated quickly. Another limitation compared to simple conventional model is the computational complexity of deep learning models. While as our result suggests that deep learning models are less sensitive to hyperparameter configurations and hence require less injection of human expertise.

Another limitations of the proposed method in that the advantages only appear when the data has cyclic characteristics and contains anomalous patterns, and if the data contains anomalies but its normal patterns are non-periodic then the performance will be equal to the previous methods in the literature. Another limitation of the proposed method, in particular the supervised deep learning models, is the requirement for higher number of samples in the training set which might be impractical in many real-world applications. Although, an alternative option could be using our suggested unsupervised method or the

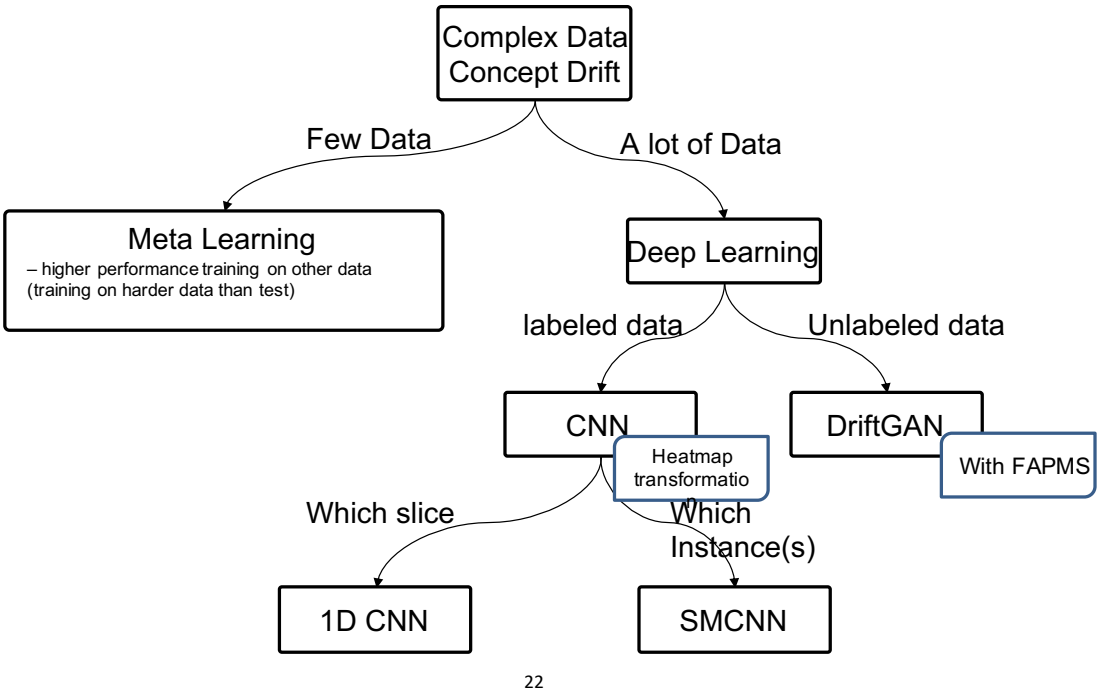


Figure 5.5: The taxonomy of the process and data types.

meta-learning approach. In general, the proposed methods and their suitable data types can be summarized in the Fig 5.5 taxonomy.

Chapter 6

Supervised Anomaly Detection in Complex and Dynamic Data

We deigned various methods to generate realistic data that mimic the variation and randomness of real-world applications to train models. We then test the models on real-world data to evaluate their performance in real applications. Also, in order to enable state-of-the-art deep learning models that are mainly designed for image and video processing to work with sequential data, we designed various methods that we describe in this section.

Following a series of deep learning breakthroughs in the area of image segmentation, multiple objects in an image input can be finely sub-categorized. Although Convolutional Neural Networks (CNNs) are known for their state-of-the-art performance in image classification, they present drawbacks when used to analyze different data types, such as time series. In this Chapter, we propose the Sequential Mask Convolutional Neural Network (SMCNN), a method that overcomes such drawbacks, and leverages CNNs for sequential data analysis. Our method transforms sequential data into an image representation by means of a specialized filter that produces flexible shape forms, and detects multiple types of outliers simultaneously. We evaluate the effectiveness of our method on data containing a variety of anomaly types combined with different concept drifts. The solution shows to significantly outperform prior endeavors and to provide high generalization capabilities on a wide array of data characteristics. We attribute its success to its ability to pinpoint the exact location of patterns and anomalies in parallel and to the invariance of CNNs, which allows them to adapt seamlessly to concept drifts.

6.1 Data transformation

We propose a method for transforming various data into one unified abstracted representation that facilitates learning complex and evolving data. Our method works via visualizing the data to a 2D tensor representation. At first glance, this might seem cumbersome and necessary to transform data (e.g., turning physical readings of a blood pressure, heart beats machine to a picture form), but it provides two main advantages. First, direct adaptation of broadly developed deep-learning models in image domain and harnessing their power requires minimum to other domains require no extra step. Second and more importantly,

this approach generalizes the models to all types of data despite their dimension or size of sequence via pixel-wise learning and extend the solutions to various domains with no significant re-configuration. For example if a sensor sampling ratio change over time or a new data format has various string sizes, the method needs no changes to preform anomaly detection task.

We propose the Sequential Mask CNN (SMCNN) technique by adapting a state-of-the-art feature masking CNN for image segmentation to sequential and time series data [59]. We introduce a fundamentally novel approach for sequential and data stream processing that uses feature masking CNNs. Inspired by recent deep learning breakthroughs in the area of image segmentation [59], [47], [30], our method allows to finely sub-categorize multiple objects in a sequential data input, while simultaneously generating a high-quality segmentation mask for each instance. To the best of our knowledge, there is no similar method for sequential data segmentation that leverages a data transformation step to convert data into an image format. This aspect of our method enables the application of CNN-based algorithms to a richer data representation that includes additional dimensions, resulting in higher generalization capabilities and a higher detection performance compared to state-of-the-art methods.

Two dimensional CNNs have previously been used in the context of time series data but in a limited way. The proposed methods attempted to map the entire data from their Cartesian coordinates to their polar coordinates [75], [91]. These approaches use CNNs for time series analysis, but look at the data as a whole, assuming that all the data (past and present) fits in the computational memory at once. These suggested methods, thus, seem impractical in real-world time series data analysis since they are not scalable, especially when considering the growth rate of certain types of data. Furthermore, these data transformation methods create images with coarser representations, including higher amounts of noise, and remove important aspects of the data, which results in a poor performance in the presence of large-scale data [113]. Furthermore, while 2D-CNNs can localize the anomaly, they provide only a coarse rectangle boundary and exhibit poor performance with sequential data, since the anomalies are usually dispersed in distant segments of the time series.

In this work, we seek to take advantage of 2D-CNNs, which we argue have a strong potential for anomaly detection on sequential data. To enable 2D-CNNs to deal with sequential data and localize and classify various shapes of patterns and anomalies, we propose a method inspired by a state-of-the-art work in the area of video segmentation [59]. In more detail, our method is based on Regional Convolutional Neural Network (R-CNN) architectures [47], [30] and overcomes the limitations of conventional 2D-CNNs, allowing to segment the target and pinpoint the exact boundaries of multiple objects of an image representation. Subsequently, a reversing function maps back the detected pattern and the anomalies from the image representation to their locations in the input sequence.

In large-scale sequential data streams, the issue of anomaly sparsity requires data transformations to abstract data without affecting hidden patterns and anomalies. Therefore, data processing and analytic techniques need to adapt to the growth and the drifts in the data over time. We introduce a novel approach for data preprocessing that transforms time series data into a tensor, which representation incorporates the notion of time.

Without loss of generality, our approach can be applied to all types of data, and gener-

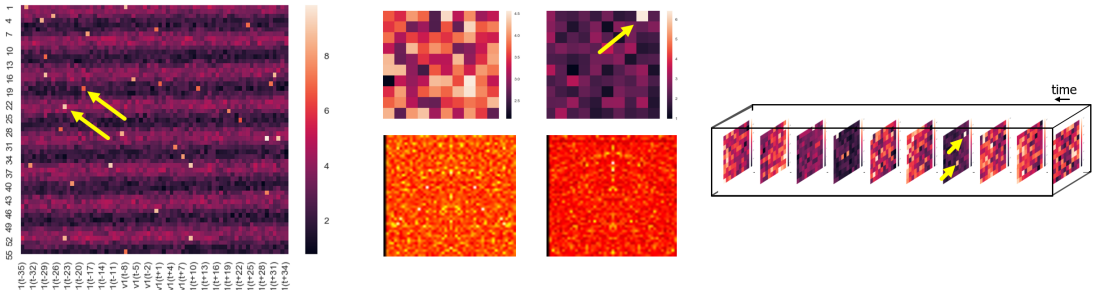
ates a pixel-wise representation that can be processed by image-based deep learning models such as 2D-CNNs without expensive re-configuration steps.

We define a sliding window of size $\pm t$, a step size s, which describes the strides; and a number of rows r, which indicates the size of the 2D representation. Let $x = [x_0, \ldots, x_{N-1}]$ be an N dimensional sequential complex vector. The final image sizes are, thus, $2t \times r$ and the total number of images in the tensor is $\frac{x}{(2t \times r)}$ assuming there is no overlap in the slides. In the cases in which the strides are not equal to the window size, Eq. 6.1 represents the total number T of images in the tensors.

$$T = 1 + \frac{x - (2t + ((r-1) \times s))}{r \times s} \tag{6.1}$$

By extracting tensors of 2D images, our approach allows to overcome the issues associated with the detection and localization of anomalies in time series data.

Specifically, a time window is passed over the time series with a lag, thus creating a series of horizontal vectors which, when grouped and stacked together, yield matrices. A matrix representing an image is obtained for every lag, resulting in a series of matrices, which we refer to as images. We then create heat-maps from our time series data (see Figure 6.1.a), and we apply Fourier's transform as we would for real images. Let m be an integer and let $\omega = \exp(\frac{-2\pi i}{N})$. The Fourier transform is given by $\mathcal{R}\left\{\frac{1}{2\pi}\int_0^{\infty} 2F(\omega)\mathrm{e}^{i\omega t}\, d\omega\right\}$. The approach can also be used in the case of multi-dimensional tensors, where multiple streams are analyzed jointly 6.1.c.



(a) Heat-map of the signal

(b) Normal data (left column). Data with anomalies (right column).

(c) Time-series transform to tensor of 2D image representation of data

Figure 6.1: (a)A heat-map displaying the two dimensional representation of time series data. Seasonality can be seen in the form of horizontal shades. (b) The left column indicates data with no anomalies and the right column represents anomalous data. (c) Turning a one-dimensional time series into a tensor of 2D images.

6.2 Sequential Mask Convolutional Neural Networks (SMCNN)

In many applications, it is necessary to find multiple patterns and anomalies using different types of shapes. To solve the issue represented by the rigid boundaries used by generic

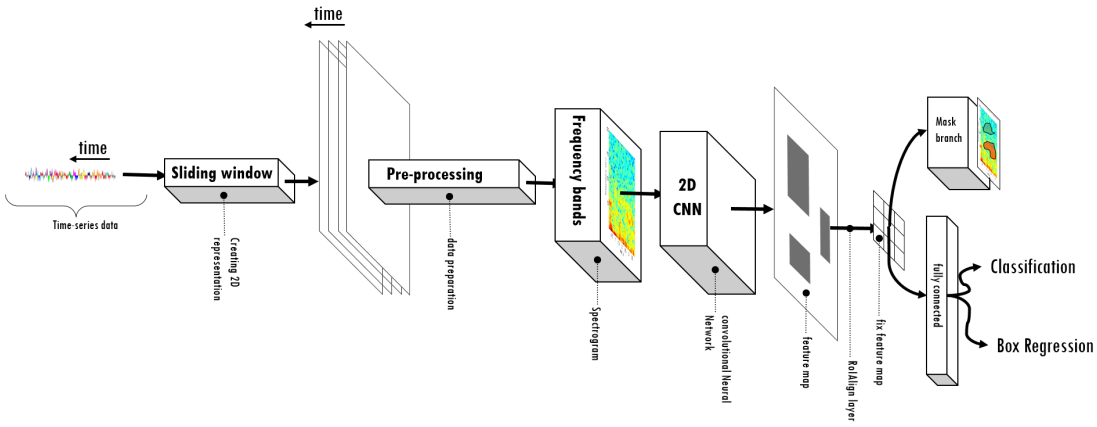


Figure 6.2: The feature masking CNN architecture. The preprocessing block transfers the time series data in raw format from one or several sensors to a 2D representation. The optional second box can apply a Fourier transform, and a 2D CNN learns the normal and anomalous data. A feature map creates rectangular boundaries that surround anomalies, and the rectangular boundaries can be reshaped around the squares of the feature maps. The masking block creates flexible bounds that can explicitly surround the anomalies.

2D-CNNs models, we introduce the SMCNN architecture, which produces flexible shape forms and detects multiple anomalies individually. Figure 6.2 illustrates the SMCNN architecture, based on the Mask R-CNN [59] approach, which enables similar segmentations in the field of object detection in video data. The idea of our approach is to transform sequential data into images so that tools such as Mask R-CNN can be applied to the sequential data.

The steps of our SMCNN method are described as follows. First, the preprocessing stage transforms raw time series from one or several sensors to a 2D representation; second, the Fourier or the heat-map transformations are applied; third, a 2D-CNN model defines the objects pixel-wise and the feature map creates rectangular boundaries that surround the anomalies. These boundaries are reshaped to fit the squares of the feature maps, and the masking block creates flexible bounds that precisely surround the anomalies. The underlying CNN model exploits a special structure to find a very precise border around the multiple types of patterns and anomalies in the image. This approach is different than the general 2D-CNNs, which uses a square bounding box that potentially introduces noise, thus causing high rates of false alarms. Notably, our experiments will show that our method can detect multiple patterns and anomalies in parallel, and define the precise boundaries of each anomaly. Variations in the size of the input image correlated with hidden patterns of the data help to cluster repeated objects located relatively close to each other. Our solution is thus able to detect, localize and classify sparse anomalies individually and in parallel. This approach can also be generalized to tensors of data streams coming from different sources.

The underlying mechanism for multiple segmentation we adopt is the Region of Interest Pooling (RoIPool) method[1], which enables our system to detect multiple targets and

[1] `http://github.com/deepsense-ai/roi-pooling`

anomalies in a single image. To obtain a fixed-size feature map, RoIPool performs max-pooling on inputs of non-uniform sizes. The benefits of RoIPool over one single CNN is that it carries out both region proposals and classification. A region proposal network that shares full-image convolutional features with the detection network, passes a sliding window over the CNN feature map, and at each window outputs k potential bounding boxes, called *anchor boxes*. The Mask R-CNN then creates a binary mask which indicates if each pixel is part of a given object. For different sizes of windows over the image, the method groups adjacent pixels by intensity of the object's identification, such as texture or color, and transforms the detected regions to standard squares. The pixel-level segmentation is a finer-grained alignment than the bounding boxes produced by a fully convolutional network, and the feature map corresponds precisely to the regions of the original image.

6.3 Experimental Results

In this section, we describe the effects of altering the data characteristics, the preprocessing configuration and the models' hyperparameters on the anomaly detection performance of several machine learning algorithms and various CNN-based models. In particular, we compare the maximum AUC, namely, the optimal point of the ratio between True Positive Rate (TPR) and False Positive Rate (FPR) obtained by the different methods. We compare this metric using different variations of the datasets (e.g. different drifts, anomaly amplitudes, window sizes, etc.).

1D-CNNs can process sequential input data in the form of time series, such as speech signals, sentences, as well as documents represented as a vector. In our study, the pre-processing, learning and evaluation steps were performed with a model trained on data presenting specific characteristics, and tested on data subject to concept drifts. As illustrated in Figure 6.3, sequential data are subject to a pre-processing stage that creates slices of data, then the CNN block learns the representations of normal and anomalous data, and returns labels. The last block is an adjustable threshold that defines the boundary between normal and abnormal classifications.

The results in Figure 6.5 show the anomaly detection performance of 1D-CNNs when a model generated with no trend (long-term direction) in the training set is evaluated with various types of trends in the test sets. As illustrated on the figure, 1D-CNNs provide very stable AUC performance for any data trend characteristics and are simple and quick to train.

6.4 Discussion

Supervised anomaly detection in sequences requires positional features to localize the anomalies. However, the 1D-pooling operations in CNNs lose all the information related to the localization of the targets. Moreover, they also suffer from the inability to detect and categorize multiple targets in one slice of the sequence.

Therefore, we investigate the performance of 2D-CNNs, which should preserve the localization information of the anomalies. Figure 6.4 illustrates the architecture adopted in our study. First, a preprocessing block transforms raw sequential time series data from

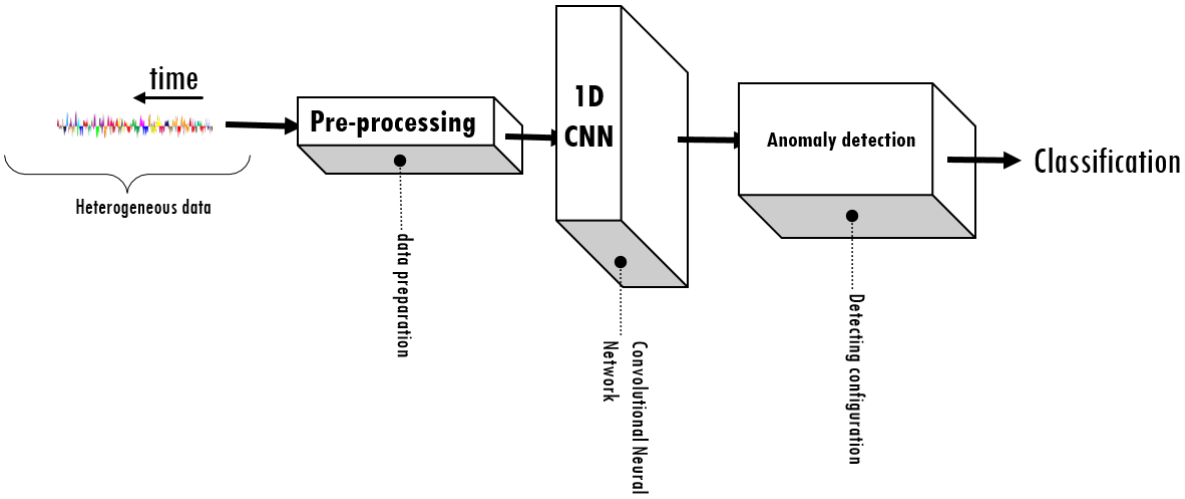


Figure 6.3: 1D-CNN architecture and related data processing pipeline used to create data slices. The CNN block learns the representations of normal and anomalous data. The last block relies on an adjustable threshold that defines the boundary between normal and abnormal class output.

one or several sensors to a 2D heat-map representation, which is a histogram of the values of the data within a window. In this representation, the x-axis represents the window size (time) and the y-axis is the number of slices in each image. For instance, a camera input that records the traffic of passing-by people, can have an x-axis of 24 hours and y-axis of 7. In this example, each image represents one week worth of data. Optionally, a frequency transformation box applies a Fourier transformation on the raw data. We perform this transformation optionally, in order to compare the performance of the histogram and the Fourier transformed images. A feature map creates rectangular boundaries that surround the anomalies. The last block is an adjustable threshold that defines the boundary between normal and abnormal classifications. The main advantage of the windowing approach combined with 2D-CNNs is the ability to localize the pattern in time.

Figure 6.6 and 6.7 show the performance obtained by 2D-CNNs. We discover that, despite our expectations, 2D-CNNs are not performing as well as 1D-CNNs, even though they go one step further and localize the anomalies. We believe that the reason for the lower performance of 2D-CNNs in detecting patterns and anomalies might relate to the nature of the anomalies residing in time series, which are scattered and sparsely located. As a result, 2D-CNNs, which exploit rigid boundaries to surround the anomalies, must create rectangles that are too large to allow the method to pinpoint the individual anomalies. This prevents the approach from generalizing well in problems of this nature, that present a very large class imbalance between normal patterns and anomalies.

Although 2D-CNNs seemed an intuitive solution to solve the shortcomings of 1D-CNNs, our results confirm that major issues arise when applying 2D-CNNs to sequential data, because: *1)* the method can only accept inputs of a fixed size; *2)* 2D-CNNs can only detect one type of target (anomaly or pattern) per scan in each slice; and *3)* in the case of sparse targets, a common property of anomaly detection in time series data, they create a large

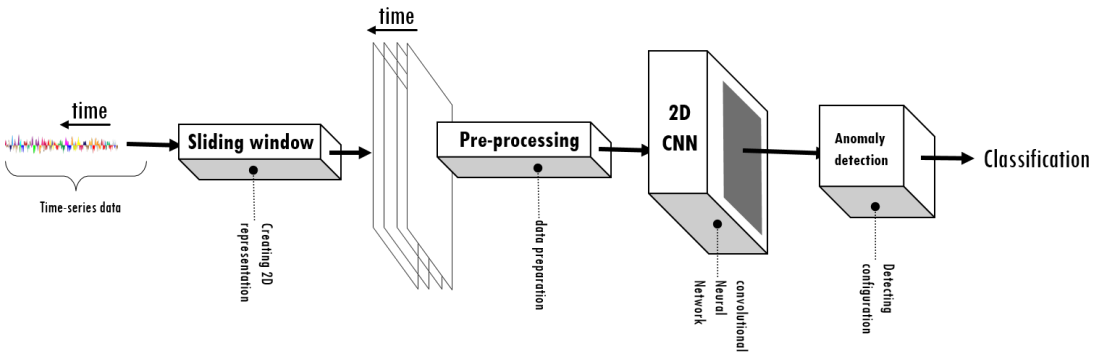


Figure 6.4: The pre-processing block transfers streaming data (e.g. continuous time series of measurements) into pieces using a sliding window approach. For training, windowed sequences are labeled. The second (optional) box applies a Fourier transform to the data. A two dimensional CNN learns the normal and anomalous data. A feature map creates rectangular boundaries that surrounds anomalies. The last block is an adjustable threshold that defines the boundary between normal and abnormal classifications.

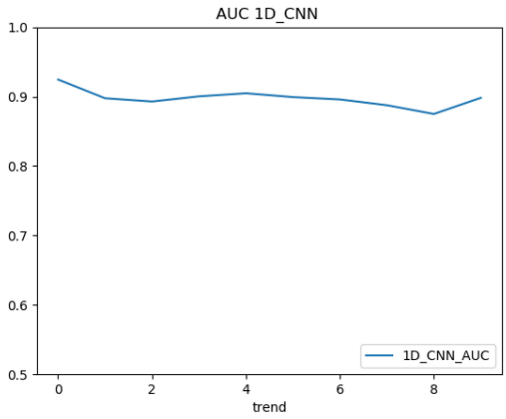


Figure 6.5: Anomaly detection performance of 1D-CNNs during concept drifts. We test each model using 10 test-sets. The x-axis represents the percentage of trend in the test data increasing from 0% to 10% of the length of the observations, while the y-axis represents the AUC score.

bound that includes everything that falls between the targets. This behavior introduces noise and decreases the detection performance by producing high numbers of false alarms.

To overcome the limitations of 2D-CNNs, we transform time series and feed the model with the image representation. Experimental results show that this transformation dramatically improves the generalization capability. However, the performance is still limited due to large feature maps and sparse targets.

Table 6.2 displays the results obtained by the different methods considered in this study. The SMCNN method achieved the highest anomaly detection performance (95.89%) using heat-map input and simultaneously defined boundaries of every individual anomaly without explicit training labels. The advantage of using heat-maps compared to Fourier

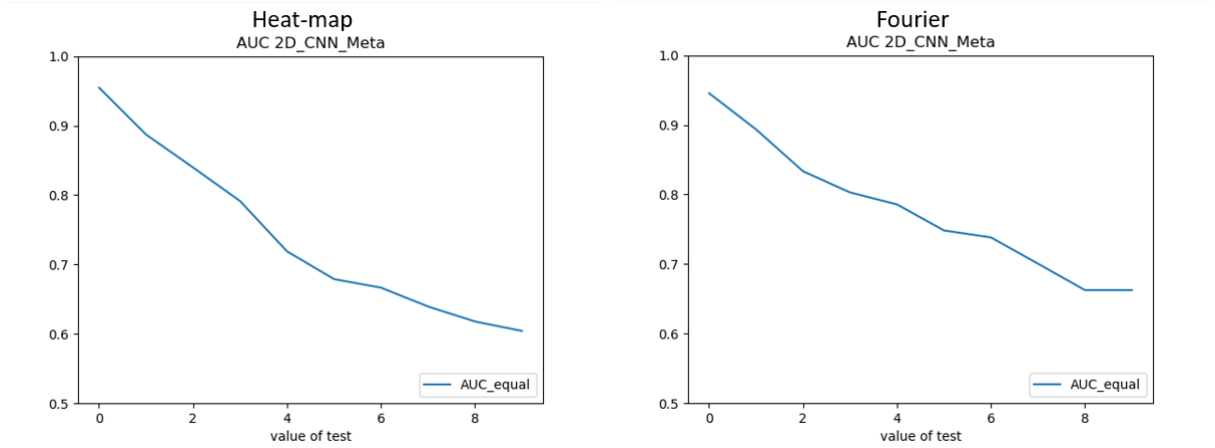


Figure 6.6: Anomaly detection performance of 2D-CNNs using a heat-map representation of time series data (top). Anomaly detection performance of 2D-CNNs using a Fourier transformed representation of time series data (bottom). The x-axis represents the percentage of trends in the test data, while the y-axis represents the AUC score.

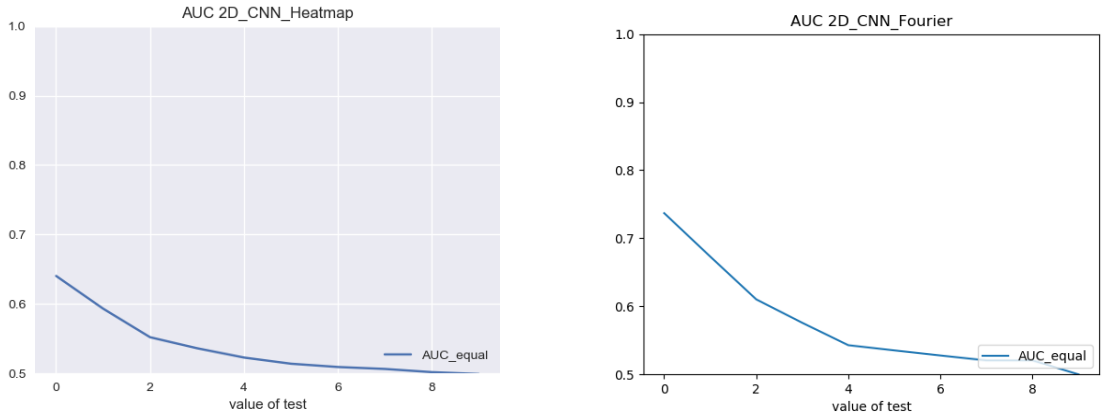
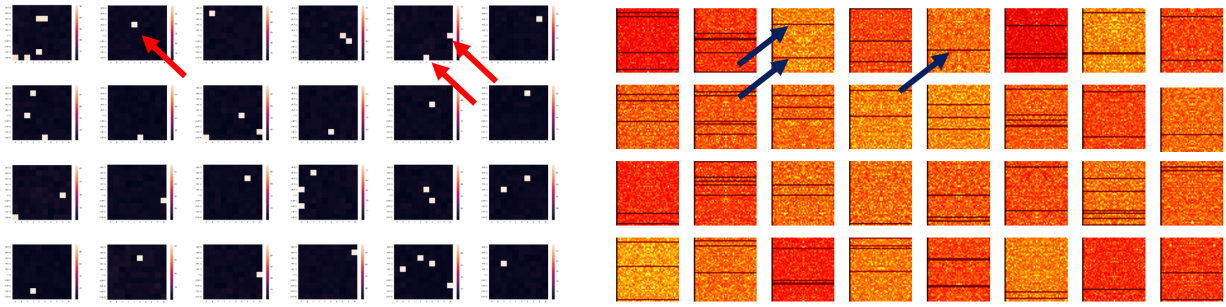


Figure 6.7: The rigid boundaries identified by traditional 2D-CNNs results in a high amount of false positive detections, and reduce the overall AUC. The performance of 2D-CNNs adopting a heat-map representation of data with various trends (top). The performance of 2D-CNNs adopting a Fourier transform representation of data with various trends (bottom). The x-axis represents the percentage of trend in the test data, while the y-axis represents the AUC score.

transformed images, despite less processing, is that the image can be directly mapped back to the input data representation and localize the object in the raw data.

We attribute the success of our method to the fact that it pinpoints the exact boundaries of multiple target objects on the image representation. Furthermore, our approach greatly reduces the requirement for labels to just one label for several window slices that are aligned on y-axis, instead of one label for each window slice. Benefiting from the transformations through layers of deep learning has made CNNs completely location and



(a) (b)

Figure 6.8: The masking CNN architecture creates flexible bounds that can exactly surround anomalies. Anomalies pointedly detected and localized on a heat-map representation of time series data (a). The masking CNN applied on a Fourier transformed representation of the data. The anomalies are detected as horizontal boundaries (b).

Table 6.1: Comparison of supervised and unsupervised models, as well as the effect of various introduced transformation on performance and generalization of GANs.

Method	AUC (Maximum)	AUC (Accumulated)
SMCNN Heatmap	95.89%	71.06%
SMCNN Fourier	95.09%	78.45%
CNN with APMS	96.01%	91.55%
CNN with FAPMS	98.21%	94.69%

representation invariant, resulting in the ability to detect objects despite the order or location of their occurrences. Specifically, CNNs produce high-level features by automatically learning filters values. As Figure 6.8 shows, SMCNN also provides an intuitive solution that can be visualized and it is easy to interpret by a human or an AI-based method.

6.5 Conclusion

Our study in this Chapter exploits a deep learning approach to detect and localize anomalies in sequential data. Experimental results showed that 1D-CNNs are unable to detect the localization of patterns. We have also shown that, since anomalies are often sparse, generic 2D-CNNs are not able to pinpoint the locations of the objects and also perform poorly in the presence of concept drift. Our method creates a two-dimensional representation of

Table 6.2: Comparison of the anomaly detection performance of different methods.

Algorithm	AUC (max)	Time	Segmentation	Strength	Vulnerability
MLP	82.72%	8620 sec	No	Robust (Vs. conventional ML)	Variation in input, fix dimensionality
LSTM	92.96%	5112 sec	No	Inputs sequential data format	Sensitive to the order and location of input
1D-CNNs	93.09%	50 sec	No	Target location invariance	Detects only if a slice contains targets
2D-CNNs (Heat-map)	63.89%	123 sec	Weakly	Target location invariance	Poor performance of localization
2D-CNNs (Fourier)	73.04%	97 sec	Weakly	Target location invariance	Poor performance of localization
SMCNN (Heat-map)	**95.89%**	343 sec	Yes	Location invariance and flexible segmentation	N/A
SMCNN (Fourier)	95.09%	298 sec	Yes	Location invariance and flexible segmentation	N/A

time series data that not only improves the performance, but also explicitly segments data to multiple target objects, and generalizes to all type of sequential data, once inputs have become images made of pixels. Our method was shown to significantly outperform prior endeavors, and to generalize to a wide variety of sequential data.

The deep learning approach based on feature masking performs best on data that has not been extensively transformed, and it is extensible to pattern and anomaly detection across various sequential data structures. The solution provides intuitive results that explicitly find multiple targets and segments them according to their subcategories. Finally, the segmentation power of the proposed Sequential Mask Convolutional Neural Network (SMCNN) architecture exceeds our expectations, since it is able to provide accurate detections and isolate multiple target elements in the data, even if the provided input representation contains only one data label for each image. In our future work, we will validate our method with real-world datasets in different domains, and investigate the effect of our data transformation approach for unsupervised anomaly detection using Generative Adversarial Networks [52].

Chapter 7

Unsupervised Anomaly Detection in Drifting Paradigms

7.1 Introduction

Learning models that are able to capture markers relevant to latent patterns and anomalies in data is a challenging task in real-world applications. A large number of labelled examples are typically required by supervised methods in order to train millions of parameters of the models. Providing high number of quality labels appears to be impractical in many real-world problems. The generalization becomes more problematic when the characteristics of the data varies in different sensors or for multiple data sources over time.

Generative Adversarial Networks (GANs) provide an architecture that allows to perform unsupervised learning with a more robust performance than other methods. In this chapter, we propose a novel approach that enables a discriminator model of a GAN architecture to perform anomaly detection tasks in temporal data. Our approach, based on one-class learning introduced by Japkowicz et al [65], only requires normal data for training and is able to detect novel patterns as anomalies.

7.2 Background of the Problem

Generative Adversarial Networks (GANs) [52] have been successfully applied to model complex and high dimensional data. More recently, GANs have also been successfully exploited for detecting novel objects [100], [34], [89]. GANs models the normal behavior using the adversarial training process and detect the anomalies with an anomaly score. GANs learn a generator that maps samples from an arbitrary latent distribution (noise prior) to data and a discriminator distinguishes between real and generated samples. The BiGAN extended GANs architecture providing an inverse mapping mechanism from the data back to the latent representation. This property of BiGAN helps the model to learn with higher speed compared to basic GAN architecture.

Schlegl et al. [100], first proposed and anomaly detection method based on GANs which identifies anomalies by discriminating between random data (in this context anomalies), and real data. The advantage of this approach is the ability of detecting all unseen

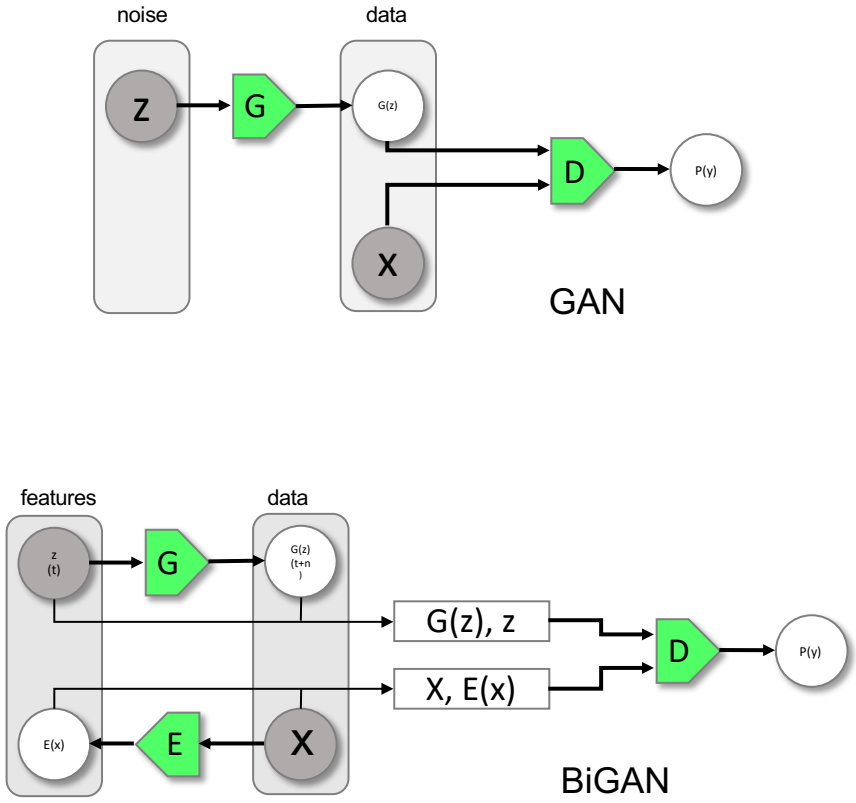


Figure 7.1: (top)Generic GAN architecture. (bottom)BiGAN architecture.

data based on unsupervised training of a model on normal data. Results demonstrate good sensitivity and the capability to segment anomalies (finely identifying instances that result in anomalies). However, Schlegl et al., propose anomaly detection based on deep generative adversarial networks but generally speaking GANs are not successful when the characteristics of the anomalous data change over time. GANs are natively designed to learn generative models that generate detailed realistic images [34], [33]. Radford et al. [89] showed that GANs are capable of capturing semantic image content enabling vector arithmetic (standard vector operations for 3D graphics, physics and animation) for visual concepts. Schlegl at al. [100] proposed AnoGAN, a deep convolutional GAN, that learns a manifold of normal anatomical variability that can serve as imaging biomarker candidates. This characteristic is accompanied by a novel anomaly scoring scheme based on a mapping from an image space to a latent space.

A GAN model trained to fit the distribution of normal samples is able to reconstruct samples from a certain latent representation and discriminate whether the samples belong from the true data distribution or not. However, as GANs implicitly model the data distribution, using them for anomaly detection requires a costly optimization procedure to recover the latent representation of a given large datasets for real-time applications. Zenati et al. [129] leveraged simultaneous encoder learning to develop an efficient anomaly detection using the BiGAN-based [34] method.

Anomaly detection methods that use GAN use a standard model, trained only on

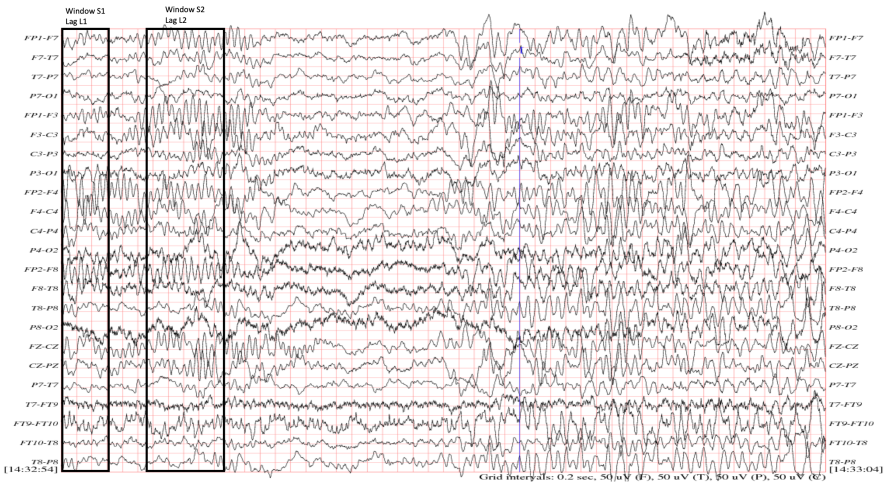


Figure 7.2: Example of anomaly detection in electroencephalogram (EEG) data. Each row represents a time series of an electrode located on the skull of a patient. Providing labels for patterns in each signal or correlative patterns i multiple time-series is cumbersome. It is more feasible to obtain training data for a GAN method from healthy groups. The model would be able to detect not only known anomalies but also any deviation from healthy patterns that is not categorized yet. The size of rolling window has a significant effect on accuracy of supervised and unsupervised models, as well as position of the window and intensity of concept drift.

positive samples, to learn a mapping from the latent space representation to the realistic sample and use this learned representation to map new, unseen, samples back to the latent space. Training a GAN on normal samples only, makes the generator learn the manifold of normal samples. Given that the generator learns how to generate normal samples, when an anomalous image is encoded, its reconstruction will be non-anomalous; hence the difference between the input and the reconstructed image will highlight the anomalies. This approach however, becomes problematic when a concept drift occurs in the normal data. Under this circumstance, the model can not differentiate between normal drift or anomalies, thus producing a high number of False Positives.

7.3 Methodology

It seems reasonable to wonder whether an unsupervised method can actually achieve the performance of supervised methods both, in terms of pattern and anomaly detection capabilities as well as boundary localization of target objects. Although the development of an optimal model might involve complex steps which impose a higher computational complexity (as in deep learning methods compared to traditional machine learning methods) it might positively affect the performance of the model.

Multiple variants of GANs have been applied to generate image [89], text [90], and video [111], or other image manipulation applications. The generator component of the architecture is the model used for these usages. In contrast, the discriminator component

[96] can be used for novelty detection.

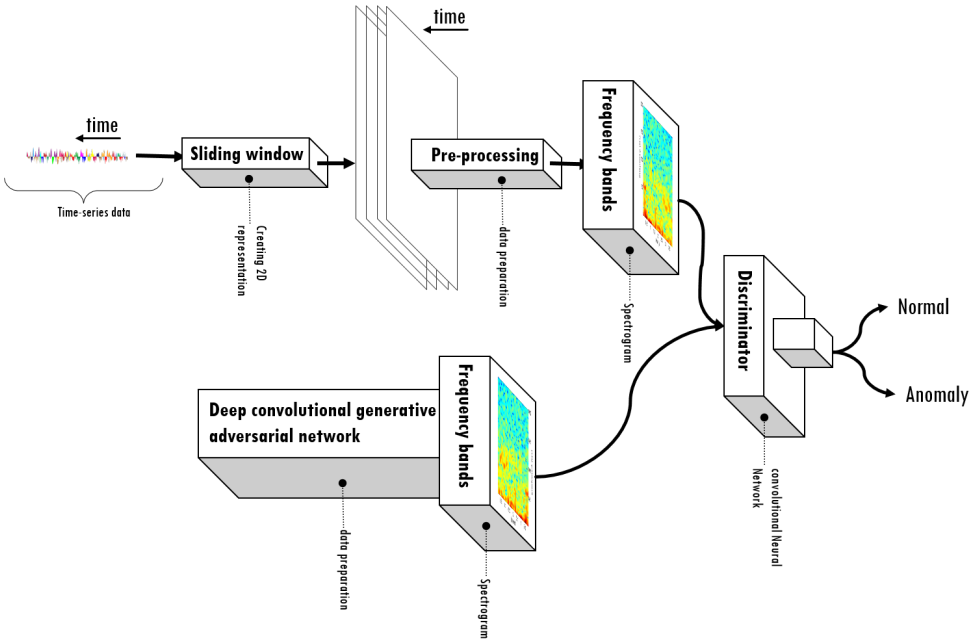


Figure 7.3: The proposed architecture for a GAN-based novel pattern and anomaly detector using transformed data. At the top-part of the architecture, the data preprocessing layers are shown that prepare the one dimensional data to a two dimensional representation. The next layer is an optional box that transforms the data from time-domain (heat-map) to frequency-domain. The Automatic Pattern Matching and Sliding method then remove concept drift and sends the preprocessed data to the Discriminator model. In the bottom section of the architecture a Generator model create fake data.

We use the data generation and transformation methods discussed in chapter 3 for training the model. We use normal data to train the GAN and evaluate the ability detecting novel anomalous patterns in unseen data.

However, the DriftGAN discriminator model has higher performance of anomaly detection, but it produces high levels of False Positives when concept drift increases which result in low AUC. The model detects any novelty even legitimate concept drift as anomaly which increase False Positive dramatically.

7.4 Results

In order to increase the performance of GANs, after transforming sequential data to the image domain, one of the important preprocessing method we adopt is a First-difference estimator. The first difference of a time series is the series of changes from one period to the next. The first difference of Y at period t is equal to $Y_t - Y_{t-1}$. The first-difference (FD) estimator is an approach by running a pooled regression of $\Delta\ y_{it}$ on $\Delta\ x_{it}$. The FD

estimator avoids bias due to some omitted, time-invariant variable c_i using the repeated observations over time. While:

$$y_{it} = x_{it}\beta + c_i + u_{it}, t = 1, ...T \tag{7.1}$$

and,

$$y_{it-1} = x_{it-1}\beta + c_i + u_{it-1}, t = 2, ...T \tag{7.2}$$

Differencing both equations:

$$\Delta y_{it} = y_{it} - y_{it-1} = \Delta x_{it}\beta + \Delta u_{it}, t = 2, ...T \tag{7.3}$$

The method randomly generate sliding windows size and the point of start of the slide, as well as number of slices in the image. Subsequently, it performs first-difference estimation on images and selects the hyper parameters that produce the darkest images in total (total values are closer to zero). In order to remove uninformative normal repetition and concept drift effects in data, the method starts with generating various sliding window sizes and starting points. Then it randomly selects a subset of window sizes and transforms those slices to images. After performing first-difference estimation on images, the method selects hyperparameters that produce darkest images in general. The hyperparameters that fits the best to the repetitive patterns are then chosen for the training model. Fig 7.4 illustrates the process.

As shows in Figs 7.5, 7.8, 7.12, there is a significant improvement in generalization of models using the Automatic Pattern Matching and Sliding method when data are characterized by concept drifts.

7.4.1 Removing concept drift significantly improved performance

In many applications, the repetition of the data patterns are not exactly aligned in the same slice of time. A travel from point A to B might have similar characteristics to a flight from C to D, but with various time lengths. Lantao Yu et al. [126], propose a sequence generation framework, that is able to handle sequences and generate new sequences. Zenati et al. [129], introduce an efficient anomaly detection using GAN. Our sequential to image transformation technique, enables a deep convolutional GAN model to learn the distribution of normal data and perform pixel-wise segmentation on anomalies. One of the biggest advantage of our method over all previous methods is the ability to define various size sliding sizes in one data while keeping the repetition points at relativity similar points. For better understanding let us explain with an example. Lets assume we would like to track a person's EEG over time. The person might enter various stages of action that affects the measurements . For example, high values in the time series might be normal if accompanied by high brain activity in certain time frames (e.g. due to watching an interesting scene in a movie) or anomalous. As it would be much more challenging to observe all other activities that might effect our measurements, the more reliable and feasible method is to only consider the differentiation between each two brainwave. However, the repetitions might not aligned equally. A slight modification in the Automatic Pattern Matching and Sliding algorithm provides flexible window sizes. In this variation of our original APMS

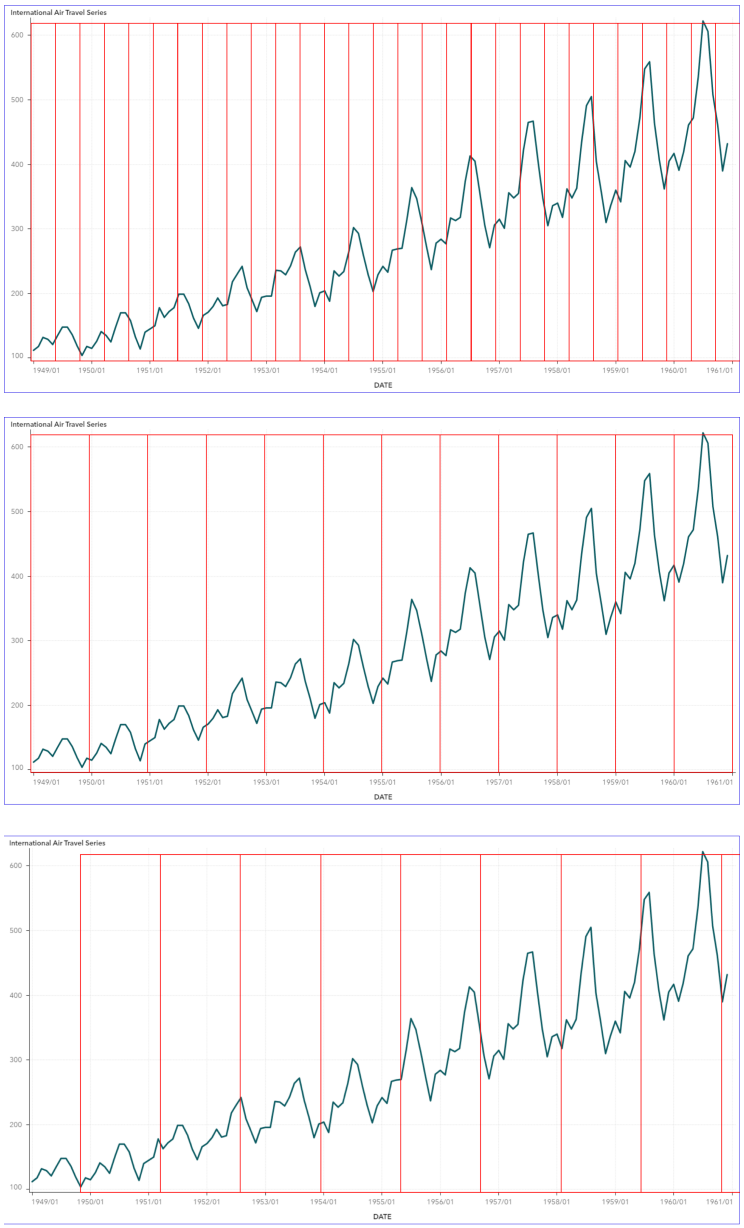


Figure 7.4: Time series are subject to a random slicing procedure to identify the optimal sliding window size. This process is required to find the optimal start and end points of the windows, in order to match with the periodicity of the time series.

algorithm window slices can be unequal and our pixel-wise method provides a solution that can deal with the inequality of the sliding windows.

The method can automatically find the repetitive patterns and fit the sliding window to each repetition and then our pixel based input model deals with the variation in the size of slices and result in better performance. As shown in Fig 7.11, Flexible Automatic Pattern Matching and Sliding produce precise window slices that fits the periodicity of patterns and improve the model detection performance when the data contains concept

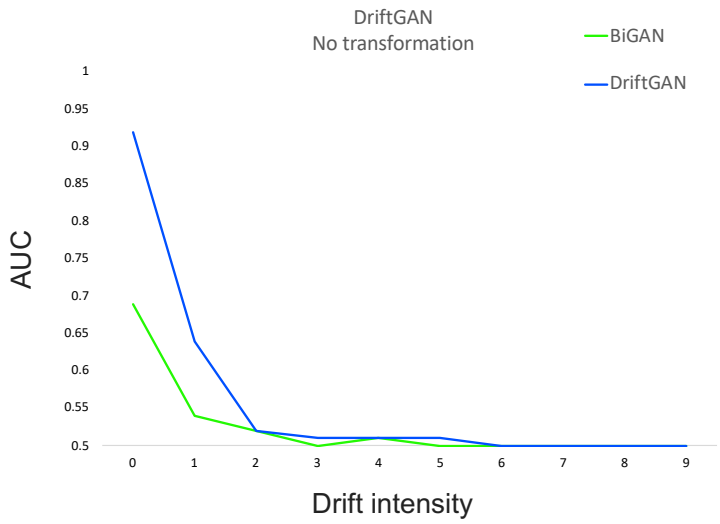


Figure 7.5: GAN-based algorithms exhibit a low performance in the presence of concept drift. The regular GAN architecture is not able to differentiate between normal drift over time and an anomalous increase.

Table 7.1: Comparison of supervised and unsupervised models, as well as the effect of various introduced transformation on performance and generalization of GANs.

Method	AUC (Maximum)	AUC (Accumulated)
GAN	68.89%	52.10%
GAN with Image Transformation	92.10%	64.15%
GAN with APMS	92.10%	83.87%
GAN with FAPMS	95.01%	88.89%

drifts, see Fig 7.12.

The proposed framework enables GANs to identify anomalies is data based on unsupervised training of a model on healthy data when concept drifts. Results demonstrates higher sensitivity and the capability to segment anomalies using our proposed extension method for transforming data and removing normal concept drifts. We think the methodology should improve both robustness to noise, and long pattern anomaly detection performance. The algorithm provided in below.

7.5 Discussion

Fig 7.3 shows the architecture of our proposed GAN model that detects novel patterns and anomalies in unseen data. Similarly to the model used in previous chapters for transforming the data as a two dimensional representation and detecting anomalous patterns. The model finds novel patterns that were not present in the training set. Similar to One Class classifiers, the model requires clean and accurate data for the training and then would be able to detect multiple novel patterns simultaneously. We compared our proposed Drift-

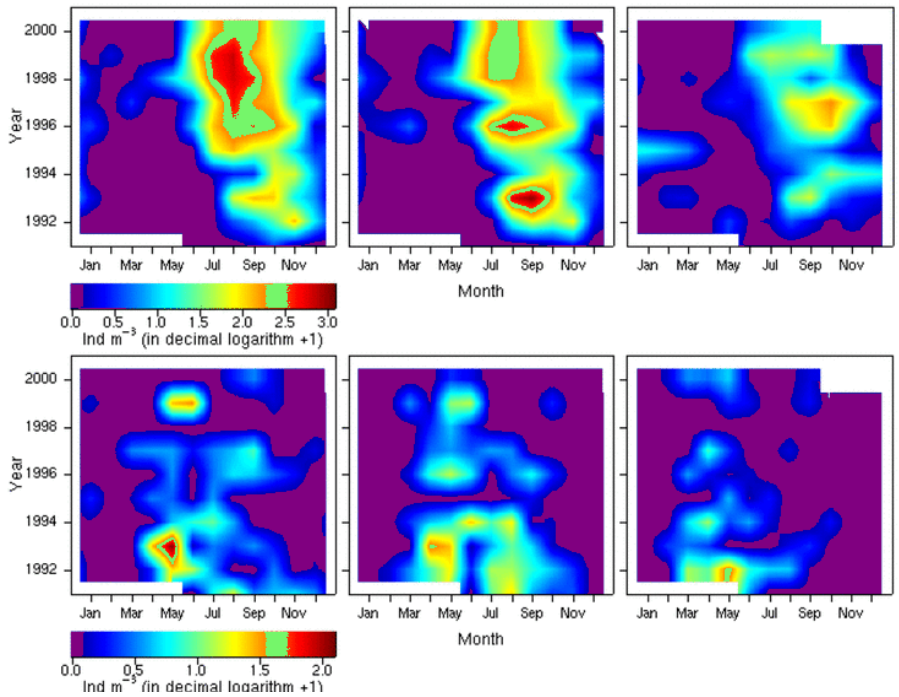


Figure 7.6: Transforming the sliding window to a two dimensional representation. The method enables CNNs models to be able to capture the notion of time. The transformation can also provide a standard input to CNN from various size data slices and time durations. For some data scenarios, the total duration of the optimal time granularity for the 2D representation is clear (e.g. one year in this example), whereas in other data scenarios, the height of the 2D representation corresponding to the sliding window size, needs to be adjusted.

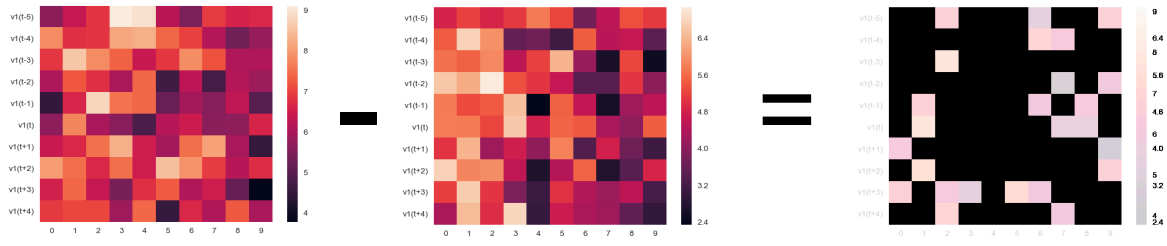


Figure 7.7: A differentiation approach allows to discard regularities in consecutive sliding windows of data, highlighting the patterns of interest.

GAN architecture and method with AnoGAN [100] model in a BiGAN [129] architecture and tested on the data while slowly changing the intensity of concept drifts. The AUC results show that regular GAN methods produce dramatically low detection performance which is motivated by the high number of False Positives, while many anomalous patterns are left undetected.

Our solution provides a unified approach for pattern detection in any type stream

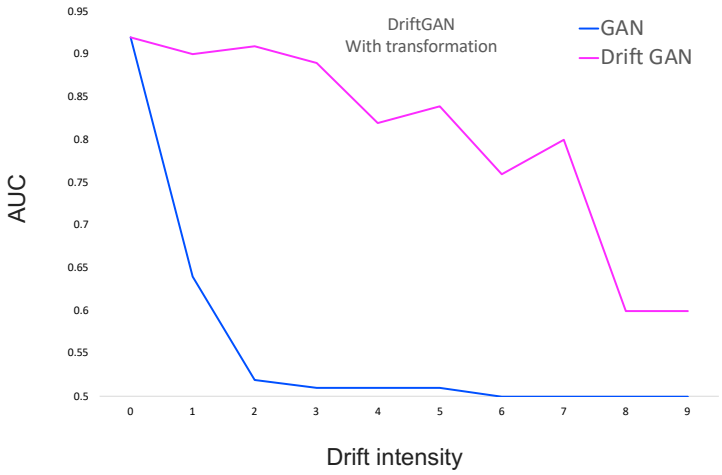


Figure 7.8: Our data transformation and differentiation approach, allow GANs to improve their anomaly detection performance significantly, when data are characterized by concept drift.

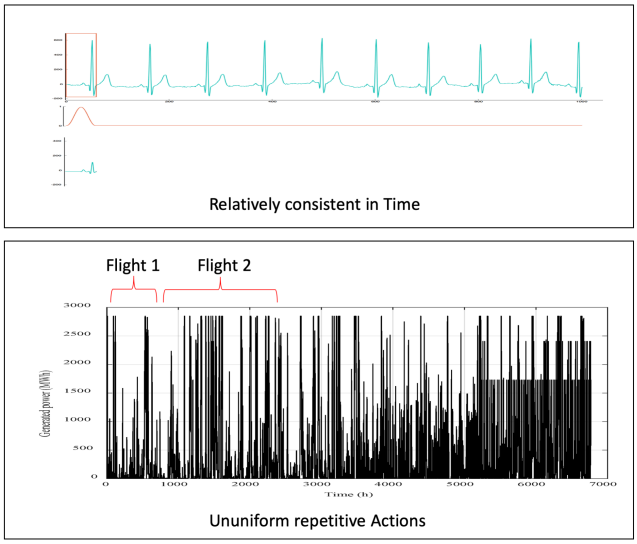


Figure 7.9: (top) ECG time series can be relatively stable, but a concept drift can result in a significant change. (bottom) Selecting an efficient window size for sequential data is crucial for high performing models. The data requires to cover a full episode of action to capture all meaningful patterns. Common approaches to select the optimal window size generally require prior knowledge about the data and the domain or application at hand. Also the size of the window can be fixed or vary over different inputs and actions. In air travel data the input can be equal to the duration of the flight that can be vary from several minutes to several hours.

or sequential data. Our sequential to image transformation technique, enables a deep convolutional GAN model to learn the distribution of normal data and perform pixel-wise segmentation on anomalies. We propose DriftGAN, an unsupervised framework that is

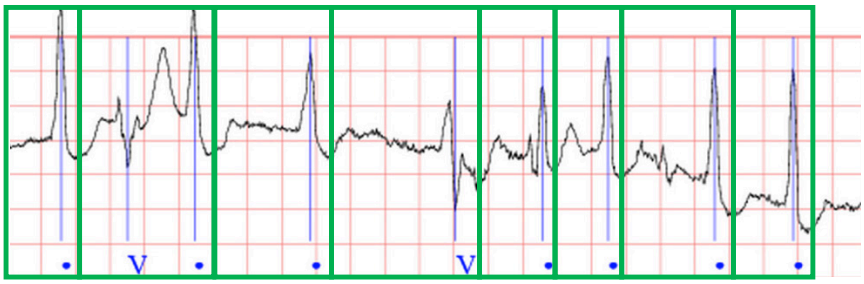


Figure 7.10: Flexible Automatic Pattern Matching and Sliding result on an EEG time series.

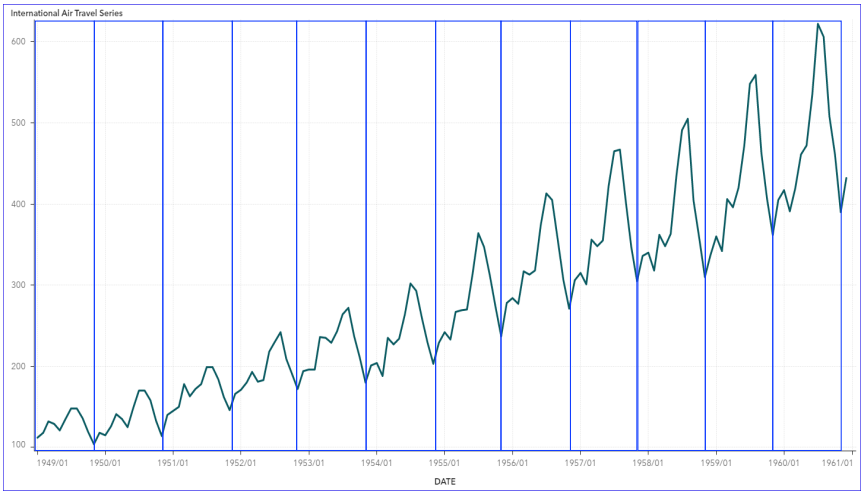


Figure 7.11: Our flexible automatic pattern matching and sliding approach leads to a further improvement in GANs anomaly detection performance with respect to random window transformation.

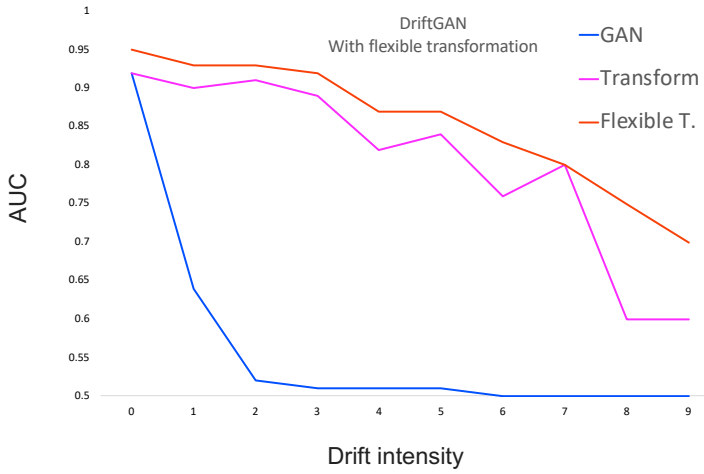


Figure 7.12: Flexible Automatic Pattern Matching and Sliding.

able to locate and isolate novel patterns and anomalies. We evaluated the performance of the model by comparing the detection and segmentation results with the performance obtained form the supervised methods. The benefit of the model is the ability of detecting and localizing novel anomalies with a training models only using normal data.

GAN's discriminative model is able to differentiate between newly seen patterns and the normal distribution of previously trained data. However, the model is not able to differentiate between novel patterns or anomalies or normal concept drift. This problem of GANs produce a high number of False Positives which reduce their accuracy in the anomaly detection task. On the other hand selecting a one dimensional model for the discriminator results in detecting only one target label per slice of data which can not identify the exact incident that lead to an anomalous slice of data.

We propose a transformation method that creates images from one dimensional data (we tested for supervised models in previous chapter) that enables GANs to detect multiple target patterns and anomalies and create bounding boundaries around each object to segment them. Our Automatic Pattern Matching and Sliding method then automatically finds optimal sliding window size, location of the start of the slices and the number of slices in each image to maximize the detection and in addition remove the concept drift that improves the overall accuracy. The method improves both robustness to noise, and long pattern anomaly detection performance. This pixel-wise input could be applied to study the improvements the generative adversarial networks [52] provides when dealing with sequential data.

7.6 Concluding Remarks

Today, the idea of systems that can create novel paradigms has morphed into an active research area inspired by the recent deep learning breakthroughs, such as Generative Adversarial Networks (GANs). GANs have shown that solve the central problem of conventional one-class classifications by using an architecture that simultaneously trains two models, namely generative and discriminative components. Nonetheless, GANs applicability has been limited to the image data. Leveraging the data transformation approach introduced in this thesis, we propose a technique for mapping from any sequential data (e.g., stream, time-series, sensor data) to and image space and then aromatically remove the normal concept drifts and make it suitable for applying GAN models to the resulting data representation. We also improve the generic GAN model to be able to segment anomalies similar to resulting data representation that was achieved using our SMCNN technique introduced in Chapter 5. An automatic flexible window size generation with pixel input models we developed before could produce highly generalizable models in concept drift. Our model performs relatively similar to the performance achieved by our supervised method with no requirements for labeled data, which makes the model applicability extend to large variety of data without labels.

Chapter 8

Conclusion

The main objectives of this thesis were to develop transformation methods that enable deep learning models to tackle sequential or stream data, and to propose analytical frameworks that can generalize to various data and concept drift. Specifically, we aimed to adopt state-of-the-art CNN segmentation and GAN discriminative methods for patterns and anomalies of sequential data, to mitigate the drawbacks of conventional machine learning algorithms. Further, by introducing APMS and FAMPS algorithms we found the optimal configuration for preprocessing data with concept drift, theses method fit the pattern repetition and remove the drift to improve anomaly detection performance. By introducing SMC-NNs and DriftGANs, we created robust systems that detect and isolate complex patterns and anomalies in various types of sequential and stream data. The system improved the computational limitations, dynamics and unpredictability, and other well-known and well-investigated challenges of sequential data involving concept drift and complex systems. Overall, this thesis provided a general discussion about applying machine learning mechanisms to create a unified system that can learn complex and constantly changing data.

The main purposes of the thesis were:

- To propose a Sequence to Image Transformation Technique to overcome the challenges mentioned earlier.
- To adapt this technique to the Supervised Setting.
- To adapt this technique to the Unsupervised Setting.
- To test the resulting approaches in a variety of situations.

Conventional machine learning methods for sequential data provide simple ways to represent a mapping between sliding window input and the label. The sliding window size remains constant throughout the training procedure and is estimated before by using trial and error along with some heuristics but require the injection of human expertise. In the majority of models, however, it may be the case that these fixed windows sizes do not capture the actual temporal locations of the time dependencies. On the other hand, the deep learning based models for sequential data are often based on Recurrent Neural Network (RNN) structures with a "memory" feature that can capture this information by learning these dependencies. The problem with RNNs is that they are impractical to use when trained with traditional techniques (e.g., Backpropagation through time) for learning long term dependencies. This problem arises from the so-called "vanishing/exploding" gradient

which basically means that as we propagate the error signals backwards through the networks structure they tend to vanish or explode. More advanced recurrent structures (e.g., LSTM) have properties that mitigate this problem and can learn long term dependencies and are particularly suited for learning sequential data with long term dependencies. However, the problem with LSTM and its prior structure RNN is that they are sensitive to the representation variance of the pattern in the time dependencies. The change of order, magnitude, size and shape of the pattern can negatively affect their performance. A 1-D CNN with simple hyperparameters (e.g., stride = 1) can perform the same by making the 1-D convolution window wide enough to include the longest delay inside the window and setting all weights for inputs neurons that do not have delay taps into zero with a certain window size. However, the problem with 1-D CNN similar to other sliding window methods is the limited precision of pattern detection. The label can only indicate if a pattern or anomaly exists in a slice of time or not but does not provide enough granularity to explicitly identify the target instances. The dilemma is that larger sliding window can fit a long-term pattern while negatively affects the granularity of the detection and smaller sliding windows sizes with higher precision do not capture the actual temporal locations of the time dependencies. A 2-D CNN can localize the target pattern but are bot compatible with sequential data. In this thesis we propose a transformation technique that enables 2-D CNNs to handle sequential data. We propose APMS and FAPMS algorithms to optimize the image transformation hyperparameters. The problem with the regular 2-D CNN object detection method is their low performance in detecting anomalies with scarce footprint. We introduce SMCNN that can mitigate this problem with a flexible masking mechanism. Further. We also introduce DriftGAN that performs unsupervised anomaly detection is sequential data with long-term time dependencies.

Overfitting and underfitting are common problems when the complexity of data and models do not match, or they change drastically after fitting. To boost the performance of models for higher levels of domain coverage (e.g. IoT devices, phones, self-learning system) with label limitations, we proposed a unified unsupervised architecture that uses Generative Adversarial Network (GAN) for detecting and localizing anomalies with no prior labels. The design provides stacks of CNNs, and a generator and discriminator that feeds networks based on the outputs of previous CNNs. Each CNN is specialized for one type of data and creates an analytical framework that might significantly increase the generalization of methods to shape a complex learning system that can deal with data from complex environments. To summarize, the contribution of this thesis consists of the following points:

- Pattern and anomaly detection: We study the performance of meta-learning and deep learning and using empirical evidence.
- Anomaly detection via segmentation for time series data: We investigate the performance of deep learning models (e.g. LSTM, and 1D and 2D CNNs).
- Pattern and anomaly segmentation: We introduce Sequential Mask CNNs (SMCNNs).
- Generative Adversarial Learning: We propose DriftGAN for unsupervised pattern and anomaly detection in the presence of concept drift in data.

Our primary objective was to generalize a model that could be used for analyzing all

types of sequential data. For this we invented a series of new approaches for transforming sequential data to visual imagery and automatically optimize the representations to remove normal pattern repetitions. We developed a mathematical formula that provides plausible descriptions of the raw data in the form of two dimensional image tensors. We also introduced adapted versions of state-of-the-art methods in object detection using CNNs and GANs to create extra powerful models for detecting patterns and anomalies. The frameworks can bring innovative studies in image and video to the domain of sequential, stream and temporal data analysis. Our methods can provide unsupervised models to perform anomaly detection with outstanding accuracy impoundments in the presence of concept drift. Our unsupervised method constructs tensors that incorporate the notion of time to the data and remove concept drift and our adapted state-of-the-art GAN technique overcomes the issue of segmenting and localizing objects, patterns and anomalies without labels.

8.1 Summary of Findings

The following are key points of our findings that each led to the development of the next step of the research.

- consistent with NFL theorem, conventional machine learning algorithms are subject to performance limitations while the characteristics of the data vary,
- when performing data science on complex and drifting data using conventional models, an automatic mechanism is required to select the optimal models without prior biases and only based on previous scores,
- deep learning outperforms the upper-bound performance of conventional models,
- to expand the applicability of deep learning models, we introduce a representation transformation to two dimensional image, which results in two gains, first the pixel-wise approach enables the method to provide off the shelf solutions to broad types of data, and second enables pattern and anomaly detection with high precision using fewer labels for data,
- for detecting and isolating patterns and anomalies in a sequence, we propose Sequential Masking CNN, with the ability of localizing boundaries of search objects,
- for detecting and isolating patterns and anomalies in a sequence without labels, we propose DriftGAN, with the ability of unsupervised localizing boundaries of search objects,

8.2 Conclusion

The study showed the necessity of a model-changing mechanism (e.g., meta learning) according to changes in characteristics of the data when using classic machine-learning algorithms. Although a trained model with certain data should illustrate its maximum performance when tested with similar data characteristics, the results suggested otherwise. Furthermore, Deep Learning could achieve better AUC and more stable results than meta-learning. Deep learning-based pattern detection is much simpler and quicker to train and

has better detection performance than classical approaches.

Our empirical result, on the tested domains, suggests that a category of algorithms, deep learning, exists that outperforms other conventional algorithms in all tasks we evaluated. This might be due to the higher bias in the structures of conventional algorithms compared to the flexibility of CNN and GAN architectures, and the parallel and auto-adoptable layers of the CNNs. CNNs, automatically produce a representation of data within the limited number of abstraction layers; the second level of fully-connected neural networks becomes superior to MLP that had stable but always average performance when concept drifts. Deep learning, particularly CNN, seems a generalizable approach for all similar problems because it has better overall AUC and stable results. 1D-CNN has the highest generalization in detecting anomalies, although the maximum resolution of detected anomalies is equal to concatenated slices. We continue to two dimensional representation in order to perform target localization.

One-dimensional CNNs achieve good generalization performance. However, we have determined that their main limitation is the low detection precision because the models are unable to address the position of target instances in a slice of the sequence. While they are able to indicate if a slice of the sequence contains anomalies, they cannot localize the anomalous instance inside a sequence. One-dimensional CNNs also suffer from the inability to detect and categorize multiple targets in one slice of the sequence. Although an intuitive solution seems to be a 2D CNN, our results confirm that major issues arise when applying 2D CNNs to sequential data because (1) the method can only accept inputs of a fixed size; (2) a 2D CNN can only detect one type of target (anomaly or pattern) per scan in each slice; and (3) in cases of sparse targets, a common property of anomaly detection in sequential data, they create a large bound that includes everything that falls between the targets, which introduces noise and decreases the detection performance by producing high numbers of false alarms. To overcome the limitations of 2D CNNs, we transform the sequential data into an image representation and feed the model with pixels of the image. This transformation dramatically modifies the generalization capability and the need for labeled instances. However, the performance is still limited due to large feature maps and sparse targets. To overcome this issue, we propose the SMCNN technique by adapting state-of-the-art feature masking CNN for image segmentation for sequential and time-series data. The solution was found to significantly outperform prior endeavors and generalize to a wide variety of sequential data. We attribute the success of our method to the fact that it pinpoints the exact boundaries of multiple target objects on the image representation. Furthermore, our approach greatly reduces the requirement for labels; in our method, labels are required only for each image that contains several instances as opposed to a label for every individual instance.

This study used deep learning to detect and localize anomalies in sequential data. The maximum precision of 1D-CNN for detecting anomalies was equal to concatenated slices of the input data. As anomalies are often sparse, generic 2D-CNNs were not able to pinpoint the locations of the objects and performed poorly in the presence of concept drift. This study created a two-dimensional representation of time-series data that not only improved the performance but also specifically segmented data to multiple target objects and generalized to all types of sequential data as the inputs became image pixels. This solution significantly outperformed prior endeavors in the area of pattern and anomaly

detection and generalized to a wide variety of sequential data.

The feature masking deep learning approach performs best on data that have not been extensively transformed and are extensible to pattern and anomaly detection across various sequential data structures. This study's solution provided intuitive results that found multiple targets and segmented them according to their subcategories. In conclusion, the power of segmentation of our sequential mask convolutional neural networks (SMCNN) exceeded expectations by providing a method that required labels for a large section of data. Minimalizing the labeled data is a very significant aspect of the study's method that was especially important for pattern and anomaly detection in large scale complex data. Detection and isolation of multiple target elements of the data was possible only through providing one label per image, not at the level of sequence or individual instances.

With using our transformation introduced in this thesis, we propose a technique for mapping from any sequential data (e.g., stream, time-series, sensor data) to and image space and then aromatically remove the normal concept drifts and make it ready for applying GAN models to the results. We also improve the generic GAN model to be able to segment anomalies similar to what was achieved using our SMCNN technique. An automatic flexible window size generation with pixel input models we developed before could produce highly generalizable models in concept drift. Our model performs relatively similar to the performance achieved by our supervised method with no requirements for labeled data, which makes the model applicability extend to a large variety of data without labels.

Although it is generally accepted that a model performs better when the characteristics of the test data are similar to those used to train the model, the findings of this study suggested otherwise. While the similarity between training and test data sets effected the performance positively, the maximum performance was not achieved; the characteristics of the two data sets were identical but not the instances. For instance, a model trained with a slightly higher trend in the training data than the test set performed maximally. Similar results were obtained by altering other properties of the data such as seasonality and amplitude of anomalies. These results illustrated the importance of using an unbiased mechanism for model selection that relied only on the previous performances of the models.

8.3 Limitations of research and suggestions for future research

This study investigated the optimal supervised methods for detecting patterns and anomalies in the data. However, while working with complex and dynamic data, the main limitation was the access to high quality labeled data with gradual changes in concept drift, which is almost always impractical in real-world applications. Therefore, we basically modelled data from real-world engineered documentation and added real-world anomalies as well as the concept drift artificially. The implementation requires testing deep learning on much more domains and applications to completely understand the performance of deep learning models compared to conventional models.

We performed an experiment to compare the performance of proposed models in detecting patterns in the three stages of the patterns appearance in the data. We used real

cyber-attack long-term anomalous patterns as a benchmark and generated artificial concept drift and characteristic changes to evaluate the generalization capabilities of the models. However, more future research can use know benchmark datasets to test our trained models and investigate if the performance remains high enough compared to state-of-the-art.

A suggestion for future research can be using drift detection models such as no-change-detection and use it in combination with our proposed methods to investigate if there is any further improvement possible with the fusion. Another suggestion could be combining our methods with few-shot learning method as these models proved improvements for data abstraction and higher performance because of its ability to quick configuration tuning [37], [39], [38]. However, most current methods use a highly increased learning rate for few-shots that are prone to overfit the noise. To overcome the stated issues of generating meta-data and selecting models on the fly, an automatic and effective abstraction system that creates representation without removing any important features of data is crucial. SMCNN and DriftGAN could potentially help to provide the input for the few-shot learner.

To extend and further test our methods we suggest using spatiotemporal approaches with our transformation technique. Instead of overlapping sliding windows to form the images, an idea could be using recently introduces 3D convolutional neural networks. 3D-CNN models have shown to achieve high performance in human action recognition [79], and in combination with GAN [101]. The extraction of features from both the spatial and the temporal dimensions by performing 3D convolutions, captures the motion information encoded in multiple adjacent frames and combines the information from all channels which might improve the performance and robustness of models.

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] Emin Aleskerov, Bernd Freisleben, and Bharat Rao. 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)*, pages 220–226. IEEE, 1997.

[3] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015.

[4] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[5] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, pages 4331–4339, 2016.

[6] UABUA Bakar, Hemant Ghayvat, SF Hasanm, and Subhas Chandra Mukhopadhyay. Activity and anomaly detection in smart home: A survey. In *Next Generation Sensors and Systems*, pages 191–220. Springer, 2016.

[7] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

[8] Gregory Bateson. Social planning and the concept of deutero-learning. In *Science, philosophy and religion, second symposium*, volume 2, pages 81–97, 1942.

[9] Gregory Bateson. *Steps to an ecology of mind: Collected essays in anthropology, psychiatry, evolution, and epistemology*. University of Chicago Press, 1972.

[10] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[11] Paula Branco, Luis Torgo, and Rita Ribeiro. A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*, 2015.

[12] Pavel Brazdil, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[13] Pavel Brazdil, João Gama, and Bob Henery. Characterizing the applicability of classification algorithms using meta-level learning. In *European conference on machine learning*, pages 83–102. Springer, 1994.

[14] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[15] Carla E Brodley. Automatic selection of split criterion during tree growing based on node location. In *Machine Learning Proceedings 1995*, pages 73–80. Elsevier, 1995.

[16] Suratna Budalakoti, Ashok N Srivastava, and Matthew E Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):101–113, 2009.

[17] Diego Carrera, Giacomo Boracchi, Alessandro Foi, and Brendt Wohlberg. Detecting anomalous structures by convolutional sparse models. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[18] Rodolfo C Cavalcante, Leandro L Minku, and Adriano LI Oliveira. Fedd: Feature extraction for explicit concept drift detection in time series. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 740–747. IEEE, 2016.

[19] Michelangelo Ceci, Roberto Corizzo, Donato Malerba, and Aleksandra Rashkovska. Spatial autocorrelation and entropy for renewable energy forecasting. *Data Mining and Knowledge Discovery*, 33(3):698–729, 2019.

[20] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

[22] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 34–37. IEEE, 2001.

[23] John S Chipman. Efficiency of least-squares estimation of linear trend when residuals are autocorrelated. *Econometrica: Journal of the Econometric Society*, pages 115–128, 1979.

[24] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[27] Roberto Corizzo, Michelangelo Ceci, and Nathalie Japkowicz. Anomaly detection and repair for accurate predictions in geo-distributed big data. *Big Data Research*, 16C:18–35, 2019.

[28] Roberto Corizzo, Michelangelo Ceci, and Nathalie Japkowicz. Anomaly detection and repair for accurate predictions in geo-distributed big data. *Big Data Research*, 16:18–35, 2019.

[29] Joseph C Culberson. On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation*, 6(2):109–127, 1998.

[30] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[31] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 international joint conference on Neural networks (IJCNN)*, pages 1–8. IEEE, 2015.

[32] Rosario Delgado and Xavier-Andoni Tibau. Why cohen's kappa should be avoided as performance measure in classification. *PloS one*, 14(9):e0222916, 2019.

[33] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks: Supplementary material.

[34] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[35] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[36] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[37] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[38] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

[39] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[40] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.

[41] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. 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*, pages 401–410. ACM, 2005.

[42] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[43] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.

[44] Dileep George. Hierarchical temporal memory: Theory and applications. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6643–6643. IEEE, 2006.

[45] Anup K Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *USENIX security symposium*, volume 99, page 12, 1999.

[46] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[47] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[48] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[49] David Gómez and Alfonso Rojas. An empirical overview of the no free lunch theorem and its effect on real-world machine learning classification. *Neural computation*, 28(1):216–228, 2016.

[50] Fabio Gonzalez and Dipankar Dasgupta. Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In *First International Conference on Artificial Immune Systems*, pages 203–211, 2002.

[51] Fabio A González and Dipankar Dasgupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, 2003.

[52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

[54] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[55] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[56] Robert Gwadera, Mikhail J Atallah, and Wojciech Szpankowski. Reliable detection of episodes in event sequences. *Knowledge and Information Systems*, 7(4):415–437, 2005.

[57] Michael Harries and Kim Horn. Detecting concept drift in financial time series prediction using symbolic machine learning. In *AI-CONFERENCE-*, pages 91–98. Citeseer, 1995.

[58] Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[59] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[60] Paul Helman and Jessie Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4):449–466, 1997.

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

[62] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[63] Charles C Holt. Planning production, inventories, and work force., 1960.

[64] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[65] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523, 1995.

[66] Hiroyuki Kawano, Shojiro Nishio, Jiawei Han, and Toshiharu Hasegawa. How does knowledge discovery cooperate with active database techniques in controlling dynamic environment? In *International Conference on Database and Expert Systems Applications*, pages 370–379. Springer, 1994.

[67] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *null*, pages 226–233. Ieee, 2005.

[68] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[69] Santosh Kumar, Anish Arora, and Ten-Hwang Lai. On the lifetime analysis of always-on wireless sensor network applications. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, pages 3–pp. IEEE, 2005.

[70] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies, 2004.

[71] Terran Lane and Carla E Brodley. Approaches to online learning and concept drift for user identification in computer security. In *KDD*, pages 259–263, 1998.

[72] Terran Lane, Carla E Brodley, et al. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49. Providence, Rhode Island, 1997.

[73] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[74] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44. IEEE, 2015.

[75] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[76] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

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

[78] Will E Leland, Murad S Taqqu, Walter Willinger, and Daniel V Wilson. On the self-similar nature of ethernet traffic, 1993.

[79] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2012.

[80] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[81] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[82] Davide Maltoni. Pattern recognition by hierarchical temporal memory. *Available at SSRN 3076121*, 2011.

[83] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[84] JANA M McPHERSON, Walter Jetz, and David J Rogers. The effects of species' range sizes on the accuracy of distribution models: ecological phenomenon or statistical artefact? *Journal of applied ecology*, 41(5):811–823, 2004.

[85] Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey, 1980.

[86] Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2003.

[87] Joshua Oldmeadow, Siddarth Ravinutala, and Christopher Leckie. Adaptive clustering for network intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 255–259. Springer, 2004.

[88] Mykola Pechenizkiy, Jorn Bakker, I Žliobaitė, Andriy Ivannikov, and Tommi Kärkkäinen. Online mass flow prediction in cfb boilers with explicit detection of sudden concept drift. *ACM SIGKDD Explorations Newsletter*, 11(2):109–116, 2010.

[89] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[90] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[91] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[92] Lior Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific, 2010.

[93] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring internet backbone traffic variability: Models metrics, measurements and meaning. In *Teletraffic Science and Engineering*, volume 5, pages 379–388, 2003.

[94] Sid Ryan. https://github.com/sidryan, 2018.

[95] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[96] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[97] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[98] Cullen Schaffer. A conservation law for generalization performance. In *Machine Learning Proceedings 1994*, pages 259–265. Elsevier, 1994.

[99] Louis L Scharf and Cédric Demeure. *Statistical signal processing: detection, estimation, and time series analysis*, volume 63. Addison-Wesley Reading, MA, 1991.

[100] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[101] Wonsup Shin, Seok-Jun Bu, and Sung-Bae Cho. 3d-convolutional neural network with generative adversarial network and autoencoder for robust anomaly detection in video surveillance. *International Journal of Neural Systems*, page 2050034, 2020.

[102] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2003.

[103] Xiuyao Song, Mingxi Wu, Christopher Jermaine, Sanjay Ranka, et al. Conditional anomaly detection. *IEEE Trans. Knowl. Data Eng.*, 19(5):631–645, 2007.

[104] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA 2001)*, pages 3–10. IEEE, 2001.

[105] Gregor Stiglic and Peter Kokol. Interpretability of sudden concept drift in medical informatics domain. In *2011 IEEE 11th international conference on data mining workshops*, pages 609–613. IEEE, 2011.

[106] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[107] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. Frequency-based anomaly detection for the automotive can bus. In *WCICSS*, pages 45–49, 2015.

[108] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 130–139. IEEE, 2016.

[109] Yun-Cheng Tsai, Jun-Hao Chen, and Chun-Chieh Wang. Encoding candlesticks as images for patterns classification using convolutional neural networks. *arXiv preprint arXiv:1901.05237*, 2019.

[110] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[111] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[112] Eric A Wan. Time series prediction by using a connectionist network with internal delay lines. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 15, pages 195–195. Addison-Wesley publishing co, 1993.

[113] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. *arXiv preprint arXiv:1506.00327*, 2015.

[114] Zhiguang Wang and Tim Oates. Spatially encoding temporal correlations to classify temporal data using convolutional neural networks. *arXiv preprint arXiv:1509.07481*, 2015.

[115] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1578–1585. IEEE, 2017.

[116] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[117] Peter Whitle. *Hypothesis testing in time series analysis*, volume 4. Almqvist & Wiksells, 1951.

[118] Gerhard Widmer and Miroslav Kubat. Effective learning in dynamic environments by explicit context tracking. In *European Conference on Machine Learning*, pages 227–243. Springer, 1993.

[119] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[120] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.

[121] Billy Williams, Priya Durvasula, and Donald Brown. Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models. *Journal of the Transportation Research Board*, 1644:132–141, 1998.

[122] Harry F. Wolcott. The anthropology of learning. *Anthropology and Education Quarterly*, 13(2):83–108, 1982.

[123] David H Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002.

[124] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[125] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001, 2015.

[126] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[127] Stefano Zanero and Sergio M Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 412–419, 2004.

[128] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

[129] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[130] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*, 2016.

[131] Fuzhi Zhang and Quanqiang Zhou. A meta-learning-based approach for detecting profile injection attacks in collaborative recommender systems. *J. Comput*, 7(1):226–234, 2012.

[132] Jiong Zhang and Mohammad Zulkernine. Anomaly based network intrusion detection with unsupervised outlier detection. In *2006 IEEE International Conference on Communications*, volume 5, pages 2388–2393. IEEE, 2006.

[133] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[134] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[135] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

[136] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

[137] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big data analysis: new algorithms for a new society*, pages 91–114. Springer, 2016.

[138] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *Sixth International Conference on Learning Representations*, 2018.