8-4-2021 10:45 AM

An Anomaly Detection System for Smart Manufacturing Using Deep Learning

Tareq Tayeh, *The University of Western Ontario*

Supervisor: Shami, Abdallah, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

The smart manufacturing evolution enables financial and operational improvements across the manufacturing industry. However, smart manufacturing encompasses complex, interconnected systems which can fail at any time. To address this challenge, a novel, two-part anomaly detection system for robotic processes, with an application focus on robotic surface finishing, is presented. The first part proposes an unsupervised Attention-based Convolutional Long Short-Term Memory Autoencoder with Dynamic Thresholding (ACLAE-DT) framework for anomaly detection and diagnosis in multivariate time series of robotic surface finishing components. The second part proposes a deep residual Convolutional Neural Network-based triplet model for anomaly detection in the produced robotic surface finishes, where defective training samples are synthesized exclusively from non-defective samples via random erasing data augmentation. Evaluation results demonstrate the performance strength of ACLAE-DT over state-of-the-art time series methods and the performance strength of the proposed triplet model in detecting anomalies for known and novel surfaces.

Keywords: Smart Manufacturing, Industrial Internet of Things, Anomaly Detection, Artificial Intelligence, Machine Learning, Deep Learning, Unsupervised Learning, Time Series, Computer Vision.

Summary for Lay Audience

Manufacturing is a critical part of any industry and is considered the essence of the secondary sector of the economy. To further increase revenue and reduce operational risks in manufacturing, smart manufacturing was born. Smart manufacturing aims to add intelligence and autonomy to manufacturing by utilizing collaborative robotics, interconnected sensors, and a range of advanced technologies in an effort to optimize the entire manufacturing process. However, the complex network of systems utilized in smart manufacturing increase the possibility of system failures, which can cause disruptive difficulties for the manufacturer. As a result, detecting anomalies, which are irregular observations that differ from the norm of the data, accurately and rapidly across manufacturing processes is vital to enabling the operators in solving the underlying issues.

In this work, a novel, two-part anomaly detection system for robotic processes, with an application focus on robotic surface finishing, is presented. The entire system leverages Deep Learning (DL), which is a collection of theories and methods that enable computers to learn from data and make predictions without being explicitly programmed. More specifically, DL allows complex features to be learned from the data autonomously without the need for manual feature development, thus creating an end-to-end learning structure with minimal human interference. The first part of the presented system proposes a DL-based framework for anomaly detection and diagnosis in multivariate time series of robotic surface finishing components. Moreover, the second part proposes a DL-based computer vision framework for anomaly detection in the produced robotic surface finishes. The entire system eliminates the use of any real-life anomalous samples during training, as there is often an abundance of non-anomalous data and a relatively small or no amount of anomalous data in well-optimized processes. Evaluation results, conducted on real-life manufacturing data sets, demonstrate the performance strengths of the proposed system over existing methods in detecting anomalies in robotic time series processes and in industrial surface images.

Co-Authorship Statement

This thesis has been structured according to the integrated-article format following the guidelines and regulations of the School of Graduate and Postdoctoral Studies (SGPS) at Western University. Chapter 2 of the Thesis has been published in arXiv, an open-access repository of electronic preprints. Chapter 3 of the Thesis has been submitted for publication to a peer-reviewed technical journal. Chapter 4 of the Thesis has been published in the 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). All domain investigation, data analysis, framework development, and draft manuscript preparation for publications were carried out by the candidate under the direct supervision of Professor Abdallah Shami. All other co-authors contributed with their technical expertise in the field and their perspective.

- **T. Tayeh** and A. Shami, "Anomaly Detection in Smart Manufacturing with an Application Focus on Robotic Finishing Systems: A Review," *arXiv preprint arXiv:2107.05053,* 2021.

- **T. Tayeh**, S. Aburakhia, R. Myers and A. Shami, "An Attention-based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series," submitted to *IEEE Internet of Things Journal*.

- **T. Tayeh**, S. Aburakhia, R. Myers and A. Shami, "Distance-Based Anomaly Detection for Industrial Surfaces Using Triplet Networks," *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, Nov 2020, pp. 0372-0377.

Dedication

I would like to dedicate this thesis to:

My Father, Ala Tayeh

My Mother, Naheel Kayyali

My Sister, Toleen Tayeh

Acknowledgements

First and foremost, I would like to thank God the Almighty for providing me with the patience, strength and ability to complete this thesis.

I would like to express my sincere gratitude to my supervisor Prof. Abdallah Shami for providing me with this research opportunity and for his continuous support and assistance throughout the entire program. I will forever be grateful for that.

Many thanks go to the amazing faculty and staff here at Western University and Western Engineering for their dedication and hard work, making achievements like this possible.

I would like to thank National Research Council Canada and Ryan Myers for partially supporting my research. Ryan's knowledge and guidance helped to push this work to the standard it is at now.

I would also like to extend my deepest appreciation to my dear friend and research partner Sulaiman Aburakhia for his continuous encouragement, assistance, and recommendations throughout this research project. I will always cherish our friendship.

Special thanks go to all my friends and colleagues that have been there for me through thick and thin throughout this journey. My experience would not have been the same without your support and encouragement. I wish you all unlimited success in the future.

Finally, I would like to extend my deepest and sincere gratitude to every member in my amazing family for their unparalleled love and support. To my parents and sister, I haven't seen you in over a year and half due to the travel restrictions imposed around the world, but you have been there for me unconditionally every single day through every medium possible. Thank you for all the sacrifices you have made in your lives to get me to where I am at today. Thank you for all the prayers, empathy, love and support. Thank you for giving me the opportunities and

experiences that have made me who I am. Thank you for everything you have done for me and still do. Words can't describe how much I love you all, you truly are my biggest blessing in this world. I dedicate all my achievements to you and will forever be in your debt.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	*Artificial Intelligence*
ACLAE-DT	*Attention-based Convolutional Long Short-Term Memory Autoencoder with Dynamic Thresholding*
AR	*Augmented Reality*
ARIMA	*Auto Regressive Integrated Moving Average*
AUC	*Area Under the Curve*
CAGR	*Compound Annual Growth Rate*
CNC	*Computer Numerical Control*
CNN	*Convolutional Neural Network*
ConvLSTM	*Convolutional Long Short-Term Memory*
DL	*Deep Learning*
ELU	*Exponential Linear Units*
GDP	*Gross Domestic Product*
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things*

ILSCVRC2014	*ImageNet Large Scale Visual Recognition Challenge 2014*
K-NN	*K-Nearest Neighbor*
LSTM	*Long Short-Term Memory*
MAE	*Mean Absolute Error*
ML	*Machine Learning*
MSE	*Mean Squared Error*
MVTec AD	*MVTec Anomaly Detection Dataset*
ReLU	*Rectified Linear Units*
ROC	*Receiver-Operating Characteristic*
RMSE	*Root Mean Squared Error*
RNN	*Recurrent Neural Network*
SELU	*Scaled Exponential Linear Units*
SGD	*Stochastic Gradient Descent*
SMART	*System-level Manufacturing and Automation Research Testbed*
SVM	*Support Vector Machine*
UAV	*Unmanned Aerial Vehicle*
VR	*Virtual Reality*

Chapter 1

Introduction

1.1 Background

Manufacturing is an essential and critical part of any industry, as it operates to produce finished, usable products from extracted raw materials and commodities. Manufacturing is considered the essence of the secondary sector of the economy, where manufactured goods are crucial for trade and the service industry, and each manufacturing job creates three other jobs in the wider economy through ”the multiplier effect” [1]. According to Statistics Canada, manufacturing was ranked as the second most contributing sector to Canada’s Gross Domestic Product (GDP) in 2020, with sales exceeding USD $510 billion for the year [2]. The manufacturing industry categories include the apparel, chemical, electronic equipment, food, fabricated metal, furniture, petroleum refining, transportation equipment industries, alongside many others.

Due to the essence of manufacturing worldwide, methods and techniques of optimizing the entire manufacturing process are required to further increase revenue and reduce operational risks. As a result, smart manufacturing was born, which aims to add intelligence and autonomy across the entire manufacturing process. More specifically, smart manufacturing aims to utilize

advanced collaborative robotics, interconnected sensors, big data analytics, real-time processing, embedded systems, computer vision technologies, unmanned trucks, Unmanned Aerial Vehicles (UAVs), Augmented Reality (AR), Virtual Reality (VR), Artificial Intelligence (AI), and various other advanced technologies for smart autonomous operations and management of supply chains. Figure 1.1 visualizes an example of a smart manufacturing setting in today's world. The global smart manufacturing market size is currently valued at USD $249.46 billion in 2021, and is expected to reach USD $576.21 billion in 2028, a Compound Annual Growth Rate (CAGR) of 12.7% [3]. These values demonstrate a demand and need for efficient smart manufacturing processes.

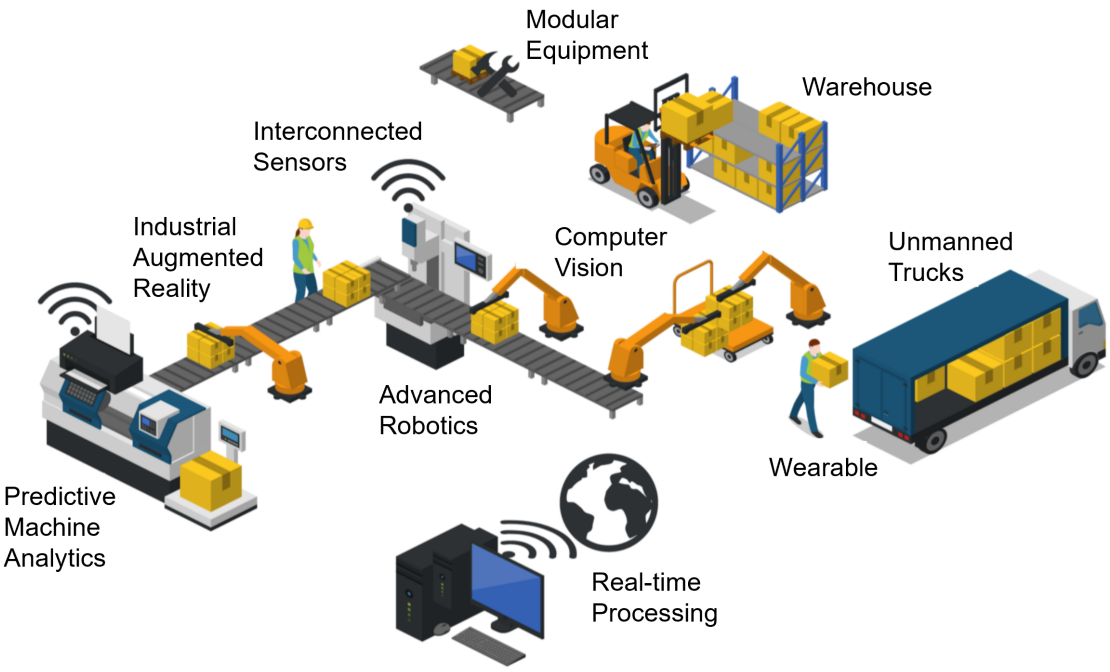

Figure 1.1: A Smart Manufacturing Setting Example

However, due to the complex network of systems in smart manufacturing, potential system or production failures might occur, imposing detrimental operational and financial difficulties for the manufacturer [4]. It is estimated that unplanned breakdowns anywhere in the production line costs industrial manufacturers USD $50 billion annually [5]. As a result, detecting anomalies accurately and rapidly across the manufacturing processes is vital to enabling the

operators to solve the underlying issues. Anomalies within a sequence of data are broadly defined as unusual observations that signify irregular behavior. Furthermore, anomalies have a very low probability of occurring, meaning that manual detection processes are a meticulous assignment that often requires more manpower than is generally available, raising the need for intelligent and automated anomaly detection systems.

One way of adding intelligence to the automated anomaly detection systems in smart manufacturing is to leverage Deep Learning (DL) architectures and algorithms, due to their superior performance strengths in the literature in recent years. DL is a subset of Machine Learning (ML), which is a data-driven AI technique that enables applications to comprehend and analyze the features and structures in the data, improving their accuracy over time without being explicitly programmed to do so. However, ML approaches often require human input for feature extraction, where they start to fail in dealing with the dimensionality and variety of the data as data volume and velocity increases [6]. DL solves the aforementioned challenge, where it transforms data into abstract representations, which allows hierarchical discriminative features to be learned and eliminates any manual feature development by domain experts. Moreover, DL integrates feature learning and model construction into a single model and trains the model's parameters jointly, creating an end-to-end learning structure with minimal human interference. The aforementioned features make DL one of the main drivers to the exponential growth of smart manufacturing, as it reduces manufacturing downtime and operating costs while gaining more value and better visibility from the operations [6]. Figure 1.2 compares DL's performance against traditional algorithms as the amount of data increases.

Various DL architectures exist in the literature, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) neural networks, Convolutional LSTM (ConvLSTM) [8] neural networks and autoencoders. CNNs are most commonly applied to the analysis of visual imagery, as it reduces the number of model parameters very efficiently without losing out on the quality of models. LSTMs, a special kind of Recurrent Neural Networks (RNNs),

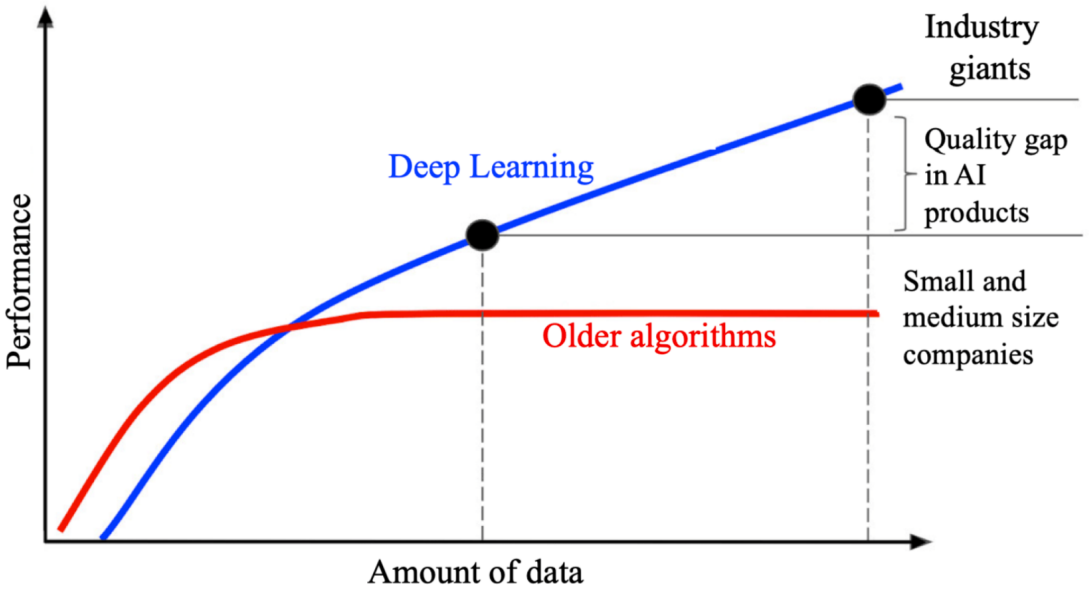

Figure 1.2: Comparison of DL with Traditional Algorithms in Smart Manurfacturing [7]

accurately models temporal behavior by employing a feedback loop for learning and using an extra parameter metric for connections between time steps. Leveraging the complementary strengths of CNNs and LSTMs, ConvLSTM neural networks emerged to accurately model the spatio-temporal information by having convolutional structures in both the input-to-state and state-to-state transitions. Finally, autoencoders impose a bottleneck in the network, forcing a compressed knowledge representation of the original input.

1.2 Thesis Goal and Objectives

In this thesis, the application focus will be on robotic surface finishing. Robotic surface finishing is one of the most important applications in smart manufacturing, as all manufactured products undergo at least a single surface finishing session during their production before they can be used. The goal of robotic surface finishing is to manufacture products without any defects and with the adequate amount of surface roughness, in order to ensure smooth and

productive operations. The goal in this thesis is to develop a novel, intelligent, and automated anomaly detection system to detect faults in robotic surface finishing machines and in the produced robotic surface finishes, while ensuring a generalization of the system to allow its extendibility into other collaborative robotic processes in smart manufacturing. Moreover, the system should operate without the use of any real-life defective samples during training, as there is often an abundance of non-anomalous data and a relatively small or no amount of anomalous data in well-optimized processes.

The work in this thesis aims to achieve three objectives:

1. Investigate the domain of anomaly detection in smart manufacturing and in robotic finishing systems.
2. Monitor and analyze robotic surface finishing machine components via sensor readings to mitigate potential surface finish anomalies.
3. Analyze the surface quality of surface finished products in an effort to detect surface defects.

1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2 aims to achieve the first objective outlined in 1.2, by providing an overview of the components, benefits, challenges, methods, and open problems of anomaly detection in smart manufacturing and in robotic finishing systems.

Chapter 3 aims to achieve the second objective outlined in 1.2, by presenting a novel, unsupervised Attention-based ConvLSTM [8] Autoencoder with Dynamic Thresholding (ACLAE-

DT) framework for anomaly detection and diagnosis in smart manufacturing multivariate time series. Furthermore, the chapter aims to achieve part of the first objective outlined in 1.2 by investigating the various anomaly detection techniques for multivariate time series in the literature.

Chapter 4 aims to achieve the third objective outlined in 1.2, by presenting a novel framework that utilizes a deep residual CNN-based triplet model trained on surface texture patches with a distance-based anomaly detection objective, where defective training samples are synthesized exclusively from non-defective samples via random erasing techniques. Furthermore, the chapter aims to achieve part of the first objective outlined in 1.2 by investigating the various anomaly detection techniques for surface finishes in the literature.

Finally, Chapter 5 summarizes the research outcomes and conclusions, along with providing recommendations for future research.

1.4 Thesis Contributions

The major contributions of the thesis are summarized as follows.

1.4.1 Chapter 2 Contributions

1. Outline the major challenges encountered when applying anomaly detection techniques to smart manufacturing data.

2. Discuss the general anomaly detection methods utilized in smart manufacturing under the time series analysis and computer vision domains.

3. Explore open problems and potential directions of anomaly detection methods in smart manufacturing.

1.4.2 Chapter 3 Contributions

1. Develop a novel framework that consists of pre-processing and enriching the multivariate time series, constructing feature images, an attention-based ConvLSTM network autoencoder to reconstruct the feature images input, and a dynamic thresholding mechanism to detect anomalies and identify anomalies' root cause in multivariate time series.

2. Implement a generic, unsupervised learning framework that utilizes state-of-the-art DL algorithms and can be applied for various different multivariate time series use cases in smart manufacturing.

3. Design an attention-based, time-distributed ConvLSTM encoder-decoder model that is capable of sustaining a constant performance as the rate of input time-series sequences from the manufacturing operations increase.

4. Construct a nonparametric and dynamic thresholding mechanism for evaluating reconstruction errors that addresses non-stationarity, noise and diversity issues in the data.

5. Evaluate the robust framework on a real-life public manufacturing data set, where results demonstrate its performance strengths over state-of-the-art methods under various experimental settings.

1.4.3 Chapter 4 Contributions

1. Artificially synthesize defective images exclusively from non-defective samples using lightweight random erasing data augmentation that do not require any extra memory consumption or parameter learning.

2. Implement a deep, residual-based CNN architecture with very small convolution kernels, allowing the network to learn more interesting features while having a small number of trainable parameters.

3. Maximize the margin of separation between the defective and non-defective images during evaluation by utilizing a hard-margin Support Vector Machine (SVM) against both the mean and maximum feature space distance values collected.

4. Evaluate the framework on a real-world data set with a focus on industrial surfaces, where results demonstrate its performance strength in detecting different types of anomalies for known surfaces that are part of the training data and unseen novel surfaces.

5. Analyze the individual defect types to gain a deeper insight into the anomalies' perceptibility when using the proposed framework.

References

[1] The Global Teach-In. Why is manufacturing important, https://www.globalteachin.com/articles/why-is-manufacturing-important, 2020.

[2] Statistics Canada. Manufacturers' sales, inventories, orders and inventory to sales ratios, by industry (dollars unless otherwise noted), https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1610004701, 2021.

[3] Fortune Business Insights. Smart manufacturing market size, share & covid-19 impact analysis, https://www.fortunebusinessinsights.com/smart-manufacturing-market-103594, 2021.

[4] Ivan Russo, Ilenia Confente, David M Gligor, and Nicola Cobelli. The combined effect of product returns experience and switching costs on B2B customer re-purchase intent. *J. Bus. Ind. Mark.*, 32(5):664–676, 2017.

[5] Wall Street Journal. How manufacturers achieve top quartile performance, https://partners.wsj.com/emerson/unlocking-performance/how-manufacturers-can-achieve-top-quartile-performance/, 2017.

[6] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.*, 48:144–156, 2018.

[7] Ruhul Amin Khalil, Nasir Saeed, Mudassir Masood, Yasaman Moradi Fard, Mohamed-Slim Alouini, and Tareq Y Al-Naffouri. Deep learning in the industrial internet of things: Potentials, challenges, and emerging applications. *IEEE Internet of Things Journal*, 2021.

[8] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Chapter 2

Literature Review: Anomaly Detection in Smart Manufacturing with an Application Focus on Robotic Finishing Systems

2.1 Introduction

Internet of Things (IoT) has boomed in recent years, due to the massive increase in the number of physical objects embedded with sensors, which allowed the creation of larger and more interconnected ”smart” networks [1]. IoT is defined as a computing paradigm that allows interrelated devices to transfer data over a network and communicate with each other without requiring human-to-human or human-to-computer interaction [2]. By the end of 2020, there was 6.6 billion active and connected IoT devices worldwide [3], and it is estimated that the potential economical impact of IoT applications will be up to USD $11.1 trillion per year by 2025 [4].

IoT is used extensively in a range of applications, such as healthcare, agriculture, trans-

portation, and retail. In particular, the industrial segment was ranked as the top IoT application area in 2020 with a global share of 22% [5], where it is often referred to as Industrial IoT (IIoT). IIoT entails interconnected sensors and devices that are networked together within industrial applications, which includes manufacturing. The connected network enables the collection, exchange, and analysis of data, facilitating improvements in industrial efficiency and overall productivity, which in turn provides cost reduction, shorter time-to-market, mass customization, sustainable production, and improved safety [6]. Accenture estimates that IIoT could add up to USD $14.2 trillion to the global economy by 2030 [7].

As IIoT systems are becoming increasingly complex and produce diverse data at an ever-increasing rate, the incorporation of computer intelligence is needed to capture the value that IIoT applications generate, resulting in the emergence of smart manufacturing. Smart manufacturing utilizes intelligent computer-controlled and internet-connected machinery to harness sensor data and automate operations to enhance the manufacturing supply chain and performance. In essence, smart manufacturing integrates IIoT with predictive analytics, big data, and Artificial Intelligence (AI), as visualized in Figure 2.1, enabling a more intelligent insight into manufacturing processes.

Statistics show that 82% of the companies that utilize smart manufacturing technologies have experienced an increase in their manufacturing efficiency and 45% have experienced an increase in customer satisfaction [8]. However, due to the complex nature of the automated, interconnected systems in smart manufacturing; potential production failures might occur. This raises the need to detect anomalies efficiently across the entire manufacturing process in order for the operators to solve the underlying issues and achieve the required product quality. In this chapter, the background, benefits, challenges, methods, and open problems of anomaly detection in smart manufacturing, with an application focus on robotic finishing systems, are discussed.

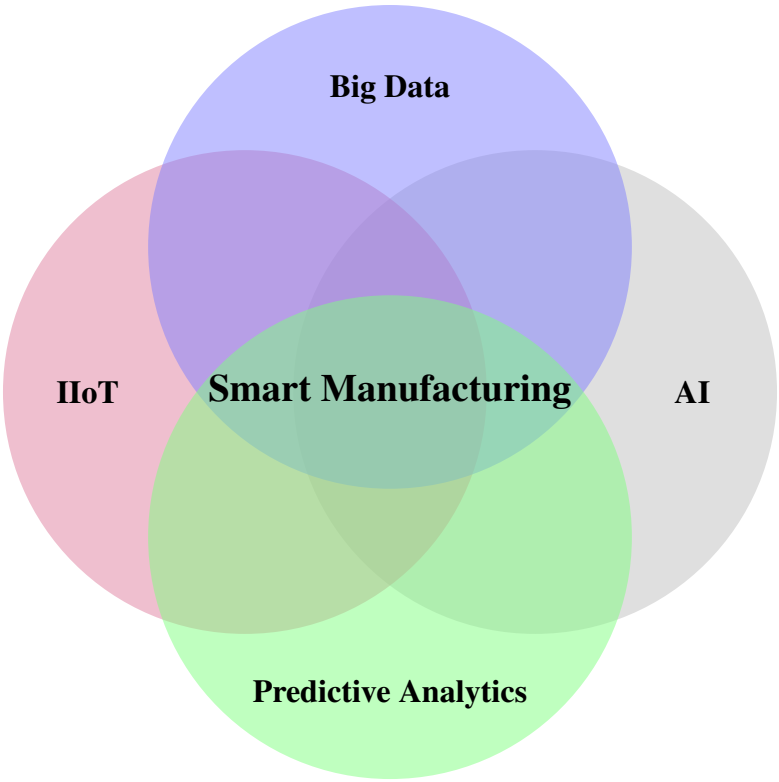

Figure 2.1: Venn Diagram of Smart Manufacturing

2.2 Background

In this section, robotic finishing systems and anomaly detection are introduced and discussed.

2.2.1 Robotic Finishing Systems

All manufactured products undergo at least a single finishing session during their production before they can be used. A finishing session aims to alter the surface of a manufactured part in order to achieve a particular characteristic [9]. Traditionally, finishing sessions have been conducted manually by human experts, particularly in small and medium volume productions where the finishing tasks are non-repetitive in nature. However, manual finishing can be labor-intensive and impose health and safety risks to the human workers. Manual finishing can add significant costs to the overall manufacturing process, as well as reducing the product manufac-

turing rate, especially when all other stages of the manufacturing process are automated [10]. As a result, automated mechanisms, and in particular, robotic-based processes, have emerged for finishing tasks instead [10].

Robotic finishing systems usually indicate either equipping a standard robotic arm with material removal devices, such as abrasive stones and grinding tools, or using the standard robotic arm to present the product to fixed finishing tools. The systems can be used for light-duty tasks such as buffing, lapping, or polishing, and for heavier operations such as weld bead finishing or burr removal. Furthermore, the introduction of Computer Numeric Control (CNC) machines allow more complex parts to be processed, as it provides the ability for program computer devices to control machine tools, rapidly advancing productivity by automating the highly technical and labor-intensive processes. Moreover, CNC machines allow the operator to call a program from memory, change parameters, and resume production in a matter of seconds or minutes.

Robotic finishing systems include [11, 12]:

1. Grinding Robots: Robots that use an abrasive wheel as the cutting tool for grinding. Each abrasive part on the wheel's surface cuts a small chip off the manufactured product workpiece via shear deformation.

2. Buffing Robots: Robots that enhance metal surfaces by removing its dull, smudged, or scratched outlook and smoothing it to give a new, refined, shiny or appealing look.

3. Polishing Robots: Similar to buffing robots, as both procedures include changing the workpiece surface to become more appealing, however, polishing involves the use of abrasive products to even out imperfections.

4. Sanding Robots: Robots that utilize a moving abrasive surface or sandpaper to smooth and remove workpiece surfaces to prepare it for painting or repairment.

5. Cutting Robots: Robots that cut or carve the workpiece into a particular shape and size.

6. Deburring Robots: Robots that remove small imperfections known as burrs from grinded, drilled, stamped, or machined sharp edges, and from corners on machined parts, molded components, forgings, castings, and welded assemblies.

7. Deflashing Robots: Robots that remove dross, flash, or parting lines on castings, forgings, molded components and welded assemblies via tumbling, breaking, grinding, or cutting. Could be considered as a subset of deburring robots in some aspects.

8. Routing Robots: Robots that rout an area in hard material via a rotating blade.

9. Drilling Robots: Robots that cut into a surface vertically by digging holes using up and down motions.

10. Milling Robots: Robots that cut into a surface vertically and horizontally with the side of the bit, digging holes using up, down, left, and right motions. Similar to drilling robots but with a wider range of use and bulkier machinery.

Utilizing these various robotic finishing systems enables efficient material removal, reduced waste production, reduced tool wear, longer tool life, increased health and wellbeing, improved energy use, and reduced harmful emissions [13].

Moreover, smart manufacturing provides the required visibility into shop floor and field operations to enable product quality control and anomaly detection, in a process called condition monitoring. Condition monitoring can be conducted by two methods: by inspecting products as it moves through the production cycle, or by monitoring the condition of the machinery and tools which manufacture the products [14]. The former method provides more accurate results by uncovering minor defects, however, it can only be applicable to discrete manufacturing and such inspections are performed manually, making it a costly, laborious and time-sensitive

operation. The latter, on the other hand, helps to detect bottlenecks in the manufacturing operation by identifying badly tuned or underperforming machines when defected products are manufactured, making it a more suitable method of use in majority of the cases.

However, even with these advantages and advances in smart manufacturing and in automated robotic finishing systems, it is still not guaranteed that a product will be manufactured free of defects, raising the need for efficient anomaly detection methods.

2.2.2 Anomaly Detection

Anomaly detection, also called novelty detection, outlier detection, or event detection, is the process of identifying unusual or novel observations or sequences within the data captured. These anomalous sequences are often referred to as anomalies, outliers, discordant observations, exceptions, faults, defects, aberrations, noise, errors, damage, surprise, novelty, peculiarities or contaminants in different application domains [15]. Furthermore, there have been various attempts to define the nature of anomalous data. Hawkins defined an anomaly as ”An observation which deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism.” [16]. Alternatively, Barnett and Lewis defined an anomaly as ”An outlier is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.” [17].

Anomaly detection is an important concept in the data analysis domain and has been widely researched in a variety of domains such as fraud detection, fault detection, and intrusion detection in diverse application domains such as health care, tax, insurance, finance, cyber security, traffic management, energy management, automated industrial processes, and many others. Anomalous data can indicate significant and critical incidents that can require urgent attention. In smart manufacturing, an anomaly is considered as an unanticipated change in the status or behavior of the IIoT system which deviates from the norm. The nature of IIoT data follows

two important observations [18]:

1. The majority of the data captured by an IIoT system is considered non-anomalous, representing the normal operating status of the system.
2. The normal operating behavior of the system can change at any time for many reasons.

Moreover, there are three broad categories of time series anomalies [18]:

1. Point Anomaly: A single data point that deviates significantly from other observations before the time series returns to its previous normal state. Point anomalies could be the result of statistical noise or faulty sensors.
2. Contextual Anomalies: Sequence of observations which deviate from the expected time series patterns in the same context. However, if these observations were taken individually, their values may be within the expected range of values for the time series. Contextual anomalies could be the result of seasonal patterns.
3. Collective Anomalies: Collection of observation which deviate from the entire data set, but the values of the individual data points are not considered anomalies contextually. Collective anomalies could be the result of combining time series data from two sensors that combined indicate an anomaly.

Many anomaly detection methods require substantial human effort to facilitate a smart manufacturing system, including data interpretation and analysis. It is easier for an expert to manually identify trends and patterns in a small subset of data representing the system, however, as the number of interconnected devices and machines increase, the complexity of the system and data analysis increases. This, in turn, increases the need for developing automated and intelligent anomaly detection approaches to enable experts to focus solely on the most important

observations. One of the main purposes of utilizing intelligent anomaly detection approaches is to provide a deeper insight into the operational state of these devices, improving the overall process efficiency. In addition, as the prices of IIoT devices fall, the likelihood of replacing older industrial equipment with new retrofitted monitoring devices increases.

Financially, it is estimated that a 1% productivity improvement across the manufacturing industry can result in an annual savings of USD $500 million, whereas a breakdown in the production line can cost up to USD $50 thousand per hour [19]. Furthermore, predicting anomalies on time can decrease the number of breakdowns by up to 70%, maintenance costs by up to 30%, and over-scheduled repairs by up to 12% [19].

2.3 Challenges

Maximizing machine availability and reducing the number of defected products is challenging. Any abrupt machine failure would result in an undesirable loss of quality and low assembly line uptime would result in loss of productivity [19]. Anomaly detection helps ease and reduce these challenges, however, there are a number of factors which make anomaly detection itself very challenging in the smart manufacturing domain. The main factors include [18]:

1. Historical Data: It is challenging to have sufficient historical data to model and define all the correct normal and anomalous states of a system when deploying an anomaly detection approach into a poorly known or novel system [20]. Moreover, there is often an abundance of non-anomalous data and a relatively small or no amount of anomalous data in well-optimized processes, making supervised learning-based approaches infeasible. In order to tackle this challenge, a number of methods exist such as manipulating the imbalanced data sets, introducing copies of known anomalies or synthetic anomalies, or reducing non-anomalous instances. However, all these approaches can damage the

temporal context between the time series, as important trends may be eliminated. Unsupervised or semi-supervised approaches tackle this lack of knowledge by training the system using only the normal data collected about the system's state so that when data falls outside of the system state's boundary it is considered as anomalous. Unsupervised or semi-supervised approaches allow the discovery of novel anomalies or anomaly modes in a new environment, but at a cost of having detailed information about the discovered anomalies.

2. Data Dimensionality: Refers to the number of separate data attributes represented in each observation [21]. Some anomaly detection approaches are unsuitable for higher dimensional data, which incur higher computational costs than lower dimensional data. IIoT data is produced in two main categories:

   (a) Univariate Data: A single sensor producing a sequence of observations or aggregated data from multiple sensors combined into a single sequence of observations is considered as univariate data. Univariate data are often in the form of key-value pair, where the key is the timestamp of the observation and the value is the corresponding sensor's reading. For example, univariate data could just indicate timestamps and the corresponding robotic finishing machine's feedrate.

   (b) Multivariate Data: Multiple sensors each producing a sequence of observations, which can be viewed as a collection of temporally correlated univariate data that provide a more thorough and rich view of the system, are considered as multivariate data. Multivariate data are often in the form key-vector pair, where the key is the timestamp of the observation and the vector is the readings associated with the different sensors. Furthermore, multivariate data requires to analyze the relationship between the various univariate data at given time steps to enrich the anomaly detection algorithm. For example, multivariate data could indicate timestamps and the corresponding readings from the output current, position, and velocity in a robotic

finishing machine.

3. Stationarity: A stationary time series is one where the mean, variance, and autocorrelation does not depend on the time at which the series is observed. A system needs to display stationarity in order to effectively utilize many smart manufacturing anomaly detection approaches. Furthermore, it is important for anomaly detection approaches to adapt to the changes in data structures for longer-term deployments, as historic anomalous data points might not be anomalous anymore given the current state of the system. Non-stationary elements include:

   (a) Seasonality: Cyclic patterns that repeat regularly over time.

   (b) Trends: The linear increasing or decreasing behavior of the series over time.

   (c) Concept Drift: Change in the statistical properties of the data flow over time in unforeseen ways. [22].

   (d) Change Points: Local or global permanent changes in the normal state of the system [23]. These changes are more sudden and rapid than those in concept drift, and the system rapidly adapts to a new state. Change points can be expected when machine or machine component upgrades occur, or an unexpected increase in usage of a particular component.

4. Noise: Represents fluctuations and unwanted random disturbance in the data caused by unrelated events within the sensor's surroundings, transmission errors in the data management system, or by slight differences in the sensitivity of the detector. Although noisy data do not affect the overall structure of the data, it is important to understand its nature and the reasons behind it as it may represent a major event within the system that requires the use of complex noise reduction techniques.

5. Contextual Information: The abundance of data available enables the inclusion of contextual information into the anomaly detection method [24]. The inclusion of contextual

information enriches the anomaly detection algorithm to correctly identify anomalous observations or sequences that do not conform to the norm, but increases the complexity of the process and introduces some challenges that must be overcome, such as:

(a) Temporal Context: It is implied that a temporal correlation between observations exist in the generated IIoT time series data [25]. For example, there exists temporal correlation between the output current, position, and velocity in a robotic finishing machine.

(b) Spatial Context: It is implied that a spatial context exists when there are multiple sensors deployed monitoring a system, such as sensor's location and activity, time of day, and sensor's proximity to other devices [25]. Spatial context becomes more challenging as it increases in size or when sensors are made mobile. For example, if sensors on the robotic finishing system were placed at different elevations or angles, observations that were previously normal at specific time steps may be anomalous when observed as the robotic finishing system conducts its task.

(c) External Context: A subset of spatial context that monitors the external conditions around the system. For example, external context could be monitoring the temperature and humidity of the manufacturing facility in which the robotic finishing system is placed in, by mounting external sensors on the system.

6. Time and Resource Constraints: The majority of IIoT devices have limited computational resources, making it challenging to process any data on these devices. Therefore recently, models are being trained and processed at a centralized location, usually in the cloud or at a datacenter, where higher computational resources are being leveraged. However, centralized processing introduces latency due to the round-trip delays and resource scheduling [26]. In many cases, having a bit of latency does not raise issues, however, in some cases, it is a requirement to process the data rapidly and be able to generate reports as soon as the data is generated [27]. The use of Fog and Edge devices

aim to perform the data processing closer to the location of the data, however, these devices have lower power than cloud services or datacenters, highlighting the importance of realizing the computational cost of any operation performed on the data during deployment [27]. Moreover, managing and storing data coming from multiple sensors can cause a concern due to its volume and velocity. More specifically, storing an entire data set collected by an IIoT network in a format easily interpreted or retrieved by the anomaly detection approach may be impractical. As a workaround, analysis and computations can be performed before further data is collected in techniques such as:

(a) Sliding Windows: Indicates creating a specified window size and performing calculations on the data within this window, before sliding or rolling to the next window based on the step size specified and so on, till the entire data has been covered in at least a single window. Sliding windows help reduce the storage requirements on devices, however, some attributes and features may not be discovered when analysis is only conducted on particular windows. Hence, anomaly detection models would need a way of retaining past trends and patterns without having access to the entire data history.

(b) Incremental Processing: Indicates processing the most recent observation only, where each data point is analyzed exactly once by the anomaly detection model. Using incremental processing requires all past trends and patterns to be retained within the model.

7. Reporting Anomalies: There are two main ways to report anomalies [28]:

(a) Anomaly Score: Usually a value between 0 and 100 that indicates the degree of deviation of a data point from the expected value, where a high deviation would translate to a high anomaly score. This method is useful when anomalies are to be analyzed further or when investigating certain types of anomalies within a particular time period. For example, the use of anomaly score would be ideal when different

types of anomalies exist in robotic finishing machines and these anomalies need to be analyzed further.

(b) Labels: A binary label indicating whether a data point observation is anomalous or non-anomalous. The assigned label is often calculated by setting a threshold and any value that crosses the set threshold is labeled as anomalous. This method is useful when immediate reporting is required or when having binary labels is sufficient for the efficiency of the anomaly detection approach. For example, the use of labels would be ideal when the finished component part is to be visually inspected for any defects.

The discussed factors and their challenges are also summarized in Table 2.1.

Factor	**Challenge**
Historical Data	- Challenging to have sufficient historical data to model all the correct normal and anomalous states of a system. - Challenging to obtain anomalous data in well-optimized processes.
Data Dimensionality	- Challenging to deal with high dimensonal data, where many sensors are each producing a sequence of observations.
Stationarity	- Challenging to deal with a time series where its mean, variance, and autocorrelation are time-dependent.
Noise	- Challenging to deal with major fluctuations and unwanted random disturbances in the data.
Contextual Information	- Challenging to deal with temporal, spatial, and external contextual information to enrich anomaly detectors.
Time and Resource Constraints	- Challenging to process data on IIoT devices due to their limited computational resources. - Challenging to store the entire data produced in a practical format for the anomaly detectors.
Reporting Anomalies	- Challenging to calculate the correct anomaly score for each obtained observation during deployment. - Challenging to set the most appropriate anomaly threshold for observations collected during deployment.

Table 2.1: Summary of Anomaly Detection Challenges in Smart Manufacturing

2.4 Methods

Effective anomaly detection in robotic finishing systems can be split into two main domains: time series analysis and computer vision. Firstly, to monitor and analyze robotic machine components via sensor readings to mitigate anomalous outcomes, time series analysis-based anomaly detection methods were developed. Secondly, to analyze the visual quality of finished products and mitigate finishing defects, automated computer vision-based anomaly detection methods were developed. In this section, each domain and its methods will be discussed in detail.

2.4.1 Time Series Analysis

Time series is defined as an ordered sequence of data points at equally spaced time intervals. Time series analysis is a statistical technique for analyzing this time series data, which enables time series models to be built to obtain an understanding of the underlying nature of the time series data or fit a model for forecasting or monitoring. An important application of time series analysis and modelling is anomaly detection, where novel or unusual states within a system are identified.

To examine the quality of the manufacturing process, components such as equipment calibration, machine conditions, and environmental conditions are monitored to ensure that their status are within their normal range and identify when they exceed beyond the normal thresholds. Ideally, if sensor readings are approaching thresholds that can lead to potential product defects, a monitoring solution should trigger an alert and pinpoint the source of the problem in order for the operators to be able to solve the underlying problems.

The general methods for detecting anomalies in time series data can be divided into the following seven groups:

1. Statistical and Probabilistic: Methods that utilize historical data to model the normal behavior of the system, where if a new observation received by the system does not fit the model, the observation is labelled as an anomaly [29, 30].

2. Predictive: Methods that generate a regression model based on historic system trends, where if a new observation received by the system vary greatly from the predicted values, the observation is labelled as an anomaly [31].

3. Reconstructive: Methods that embed data into a lower dimensional subspace and then reconstructs them, where if a new observation received by the system vary greatly from the expected values, the observation is labelled as an anomaly.

4. Pattern Matching: Methods that use direct modeling of the time series, where if a new observation received by the system follows known characteristics of anomalous subsequences based on a database of labelled anomalies or historic patterns within normal data, the observation is labelled as an anomaly.

5. Distance-Based: Methods that utilize distance metrics to model relationships between observations, where if a new observation possess a large enough distance of separation with the normal data, the observation is labelled as an anomaly [21].

6. Clustering: Methods that project the data into a multi-dimensional space to create clusters, where if a new observation received by the system is further away or does not belong to the dense clusters, the observation is labelled as an anomaly [21].

7. Ensemble: Methods that utilize different algorithms to observe data points, where a voting mechanism is employed over the results of each algorithm to produce the resulting ensemble method output [32].

Table 2.2 discusses pros and cons of the aforementioned general methods. Furthermore, Chapter 3.2 discusses various state-of-the-art methods in the literature that utilize time series

analysis to detect anomalies in time series data. As the work in this thesis takes into account many different manufacturing components and conditions, only methods that detect anomalies in multivariate time series data were surveyed.

Method	**Pros**	**Cons**
Statistical and Probabilistic	Often easier to detect outliers as real-life data often follows some statistical distribution.	Often challenging with high dimensional data.
Predictive	Useful for real-time prediction of anomalies.	Often requires large volume of training data.
Reconstructive	Able to detect novel anomalies.	Often requires large volume of training data.
Pattern Matching	Robust in detecting anomalies.	Often requires large volume of training data.
Distance-Based	Works well when normal data is highly concentrated around a region.	Highly dependent on the distance measures and anomaly scoring mechanisms used.
Clustering	Unsupervised method.	Can mistake data points in less dense clusters as anomalies.
Ensemble	Often improves the anomaly detection performance.	Increases computation time and complexity.

Table 2.2: Pros and Cons of the General Anomaly Detection Methods

2.4.2 Computer Vision

Computer vision is a field concerned with enabling computers to gain high-level understanding from digital images or videos, by identifying and processing images in the same way that human vision does. This interdisciplinary field aims to simulate and automate elements of human vision by using sensors and AI algorithms, emulating human intelligence and instincts in a computer. Anomaly detection is one of the most important applications of computer vision, where novel or unusual images or parts of an image within a system are identified.

Moreover, with the increasing demand for maintaining an almost perfect product quality finish in smart manufacturing, surface finishes are being monitored by cameras installed on various machines to ensure the surface finish roughness and appearance are within the ac-

ceptable normal visual bounds. Ideally, if a surface finish does not meet the required product standard and falls out of the acceptable normal bounds, then the manufacturing process will be flagged and will request the intervention of system operators for necessary further examination.

There are four primary computer vision techniques for detecting anomalies in images:

1. Image Classification: Technique that aims to comprehend an image as a whole to classify the entire image as normal or anomalous.

2. Object Detection: Technique that aims to comprehend an entire image to locate and classify potential anomalous objects within an image.

3. Semantic Segmentation: Technique that aims to segment an entire image and label each pixel to detect potential anomalous objects within an image. Treats multiple objects of the same class as a single entity.

4. Instance Segmentation: Technique that aims to segment an entire image and label each pixel to detect potential anomalous objects within an image. Treats multiple objects of the same class as distinct individual objects.

Moreover, the general methods mentioned in 2.4.1 can also be classified as the general methods for anomaly detection in the computer vision domain, but they aim to achieve one of the aforementioned four primary computer vision techniques for anomaly detection [33]. Chapter 4.2 discusses various state-of-the-art methods in the literature that utilize computer vision to detect anomalies in surface finishes.

2.5 Open Problems

Although anomaly detection methods are becoming increasingly accurate and efficient, there are still research challenges that require further exploration. The following are some of the open problems faced by anomaly detection methods in smart manufacturing [18, 34, 35, 36, 37]:

1. Real-Time Processing: In smart manufacturing, detecting anomalies in real-time or near real-time is of utmost importance due to the nature of the manufacturing process. Defective products and machine components need to be detected instantly in order for the operators to fix the underlying issues. Otherwise, if the anomaly detector takes a long time to process observations to flag anomalies, the system will fail, imposing financial and operational problems.

2. Reducing Storage Requirements: It would be very costly to hold all the data generated in smart manufacturing for processing and detecting anomalies. Techniques such as sliding windows and incremental processing help reduce the storage and memory requirements for the processing system, but more novel methods could be explored in the future to allow a reduction of storage requirements while processing a larger number of observations.

3. Change in Data Structures: As data may change during longer-term deployments, an anomaly detection method should be able to identify this data change and still be able to efficiently detect anomalies. However, completely retraining the method with the new data structures is costly, raising the need for online adaptive learning methods to be developed, in order to adapt to novel behavior in the data. Offline methods could be used prior to the the deployment of the online adaptive method for initial method training on the historical data collected from the system.

4. Insufficient Labeled Anomaly Data: In the real-world, manufacturing processing are

often well-optimized, meaning that there will be an abundance of non-anomalous data but a relatively small amount of anomalous data. The lack of anomalous data could impose a limitation on supervised learning approaches, due to the data imbalance and that these scarce anomalous data will often not represent the whole range of potential anomalies that could occur in the system. As a result, efficient unsupervised and semi-supervised approaches need to be developed to ensure the anomaly detection methods are able to detect novel anomalies during deployment.

5. Utilizing More Data Sources: Utilizing different types of data from various data sources, such as the inclusion of contextual information or incorporating more sensors around the manufacturing systems, can enable the anomaly detection algorithm to learn enriched data behaviors and structures that can assist with detecting hidden anomalies. Therefore, it is important for anomaly detection algorithms to be able to process multivariate data successfully.

6. Developing a Generalized Anomaly Detection Method: The development of generalized anomaly detection algorithms than can be applied in multiple different areas in smart manufacturing or different domains enables the reusability of methods and eases their deployments in new environments.

7. Interpretability of Detected Anomalies: Many anomaly detection methods are proficient at detecting anomalies, however, most also fail in providing clear and concise reasons behind the anomaly detection, particularly in rare anomalies which can lead to algorithmic bias against particular sensors or machines. To mitigate this risk, anomaly detection algorithms should be able to provide straightforward cues about why certain data points are flagged as anomalous, enabling human experts to understand the system better and correct the bias if needed.

2.6 Conclusion

Deploying anomaly detection algorithms in smart manufacturing reduces many financial and operational risks, while reducing the monitoring burden on system operators. However, these algorithms are associated with many challenges which need to be taken into account. These challenges are dictated by the system's specific application, objective, and environment, and need to be addressed when designing and developing the algorithms. Computer vision methods and time series analysis methods are the two most common domain of anomaly detection methods in smart manufacturing, where the former aims to detect anomalies in sensor data and the latter aims to detect anomalies in the produced surface finishes. Although anomaly detection methods are constantly improving with time, many open problems still exist, meaning anomaly detection algorithms still have a significant amount of room to evolve and become more efficient, accurate, and generalized.

References

[1] Hassan Hawilo, Manar Jammal, and Abdallah Shami. Network function virtualization-aware orchestrator for service function chaining placement in the cloud. *IEEE Journal on Selected Areas in Communications*, 37(3):643–655, 2019.

[2] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[3] ABI Research. Tomorrow's smart connected products require smarter connectivity services today, https://iotbusinessnews.com/download/white-papers/ABI-RESEARCH-Tomorrows-Smart-Connected-Products.pdf, 2020.

[4] McKinsey&Company. The internet of things: Mapping the value beyond the hype, https://www.mckinsey.com/media/OurInsights/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.pdf, 2015.

[5] Preddio Technologies. Understanding iot, https://iotbusinessnews.com/download/white-papers/PREDDIO-TECHNOLOGIES-Understanding-IoT.pdf, 2021.

[6] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1 – 12, 2018.

[7] Accenture. The secret to maximizing the industry iot, https://www.accenture.com/sg-en/insights/consulting/secret-maximizing-industrial-iot, 2019.

[8] Smart Manufacturing Coalition. Manufacturing growth continues despite uncertain economy, https://smlconsortium.org/sites/default/files/12.16.13_manufacturing_outlook_survey.pdf, 2013.

[9] 3DExperience. Introduction to finishing processes, https://make.3dexperience.3ds.com/processes/introduction-to-finishing-processes, 2021.

[10] J Norberto Pires and Robert Bogue. Finishing robots: a review of technologies and applications. *Industrial Robot: An International Journal*, 2009.

[11] Automated robotic finishing systems: Mesh automation, inc., https://meshautomationinc.com/robotic-machining-finishing/, 2020.

[12] Robotic cutting, deburring, polishing, grinding, buffing; more, https://www.fanucamerica.com/solutions/applications/material-removal.

[13] Alex Owen-Hill. The 10 Sustainability Benefits to Choosing Robot Finishing, https://blog.robotiq.com/the-10-sustainability-benefits-to-choosing-robot-finishing, 2020.

[14] ScienceSoft. Iot in manufacturing: The ultimate guide, https://www.scnsoft.com/blog/iot-in-manufacturing, 2018.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 14:15, 2007.

[16] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[17] Vic Barnett and Toby Lewis. Outliers in statistical data. *osd*, 1984.

[18] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019.

[19] Progress. Anomaly Detection & Prediction Decoded: 6 Industries, Copious Challenges, Extraordinary Impact, https://www.progress.com/docs/default-source/datarpm /progress_datarpm_cadp_ebook_anomaly_detection_in_6_industries.pdf?sfvrsn=82a183de _2, 2017.

[20] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[21] V Chandola and A Banerjee. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 2009.

[22] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[23] Michele Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice hall Englewood Cliffs, 1993.

[24] Michael A Hayes and Miriam AM Capretz. Contextual anomaly detection in big sensor data. In *2014 IEEE International Congress on Big Data*, pages 64–71. IEEE, 2014.

[25] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet of Things Journal*, 5(1):1–27, 2017.

[26] Sumon Kumar Bose, Bapi Kar, Mohendra Roy, Pradeep Kumar Gopalakrishnan, and Arindam Basu. Adepos: Anomaly detection based power saving for predictive maintenance using edge computing. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 597–602, 2019.

[27] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhán Clarke. Middleware for internet of things: a survey. *IEEE Internet of things journal*, 3(1):70–95, 2015.

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

[29] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[30] Anas Saci, Arafat Al-Dweik, and Abdallah Shami. Autocorrelation integrated gaussian based anomaly detection using sensory data in industrial manufacturing. *IEEE Sensors Journal*, 21(7):9231–9241, 2021.

[31] Federico Giannoni, Marco Mancini, and Federico Marinelli. Anomaly detection models for iot time series data. *arXiv preprint arXiv:1812.00890*, 2018.

[32] Li Yang, Abdallah Moubayed, and Abdallah Shami. Mth-ids: A multi-tiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[33] Kelathodi Kumaran Santhosh, Debi Prosad Dogra, and Partha Pratim Roy. Anomaly detection in road traffic using visual surveillance: A survey. *ACM Computing Surveys (CSUR)*, 53(6):1–26, 2020.

[34] Tareq Tayeh, Sulaiman Aburakhia, Ryan Myers, and Abdallah Shami. Distance-based anomaly detection for industrial surfaces using triplet networks. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0372–0377.

[35] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *arXiv preprint arXiv:2007.02500*, 2020.

[36] Li Yang and Abdallah Shami. A lightweight concept drift detection and adaptation framework for iot data streams. *IEEE Internet of Things Magazine*, pages 1–6, 2021.

[37] Sulaiman Aburakhia, Tareq Tayeh, Ryan Myers, and Abdallah Shami. A transfer learning framework for anomaly detection using model of normality. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0055–0061.

Chapter 3

ACLAE-DT: An Attention-based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series

3.1 Introduction

Manufacturing has advanced substantially in recent years and has become more computerized, complex, and automated, driving the emergence of smart manufacturing [1]. Smart manufacturing is a technology-driven approach that integrates Artificial Intelligence (AI), predictive analytics, big data, and Industrial Internet of Things (IIoT) to harness sensor data and automation to improve manufacturing performance. As quality control is an indispensable part of the production process in all manufacturing industries around the globe, smart manufacturing enables higher quality products to be manufactured, while reducing costs and improving safety

[2].

Smart manufacturing encompasses complex systems of interconnected sensors and computer components with diverse types that generate a substantial amount of multivariate time series data. As a result, potential system or production failures might occur, deeming anomaly detection at certain time periods as a vital task in order for the operators to solve the underlying issues. Anomalies within a sequence of data are broadly defined as observations that are unusual and signify irregular behavior. These irregular and unusual events have a very low probability of occurring, meaning that manual detection processes are a meticulous assignment that often requires more manpower than is generally available. Broken, cracked, and other imperfect products may result in costly returns, imposing operational and financial difficulties [3]. It is estimated that a 1% productivity improvement across the manufacturing industry can result in an annual savings of USD $500 million, whereas a breakdown in the production line can cost up to USD $50 thousand per hour [4]. Furthermore, predicting anomalies on time can decrease the number of breakdowns by up to 70%, maintenance costs by up to 30%, and over scheduled repairs by up to 12% [4]. As a result, Machine Learning (ML) has been utilized in recent years to detect anomalies [5, 6, 7, 8], and in particular, Deep Learning (DL) algorithms [9, 10, 11, 12].

Automated anomaly detection algorithms leverage ML due to the latter's data-driven nature, efficiency, and ability to perform data analysis without explicitly programming the application [13]. Moreover, DL in particular, is able to learn higher level features from data in a hierarchical fashion while continuously improving the system's accuracy and automated decision making even as the dimensionality and variety of the data increases [14]. The aforementioned features make DL one of the main contributors to the fast growth of smart manufacturing, as it allows operating costs and downtimes to be reduced while gaining more value and better visibility from the operations [14].

A basic requirement for building a supervised learning-based automated system to detect

anomalies on a classification objective is the accessibility of a sufficient amount of training data for each class [15]. However, efficient supervised learning methods are often infeasible, as with well-optimized processes, there is often an abundance of non-anomalous data and a relatively small or no amount of anomalous data. To address this data imbalance challenge, a substantial amount of unsupervised anomaly detection methods have been developed in recent years, as these methods are trained on unlabeled input data with no output variables. An additional advantage of this approach is potentially detecting anomalies in novel classes that are not part of the training data set, providing a general solution to the surface quality manufacturing process task [16].

Some previous approaches in the literature use defective samples for training, not solving the anomaly detection task described in this work. Other approaches utilize classical methods, such as probabilistic, distance-based, clustering, ensemble, and predictive approaches to detect anomalies in an unsupervised fashion, but all fail to capture complex structures in the data without the input of domain experts [16]. Unsupervised ML approaches were then proposed to capture these complex structures, however, they start failing to deal with the high dimensionality of the data feature space and the varying data aggregation as the data volume increases, requiring human expertise for feature extraction [14]. Various DL architectures, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) neural networks, Convolutional LSTM (ConvLSTM) [17] neural networks and autoencoders have emerged in recent literature to solve the aforementioned challenges. Furthermore, when a production failure is occurring, it is important to localize the anomaly root cause to plan adequate countermeasures and fix the production system. This is done by pinpointing the sensors with anomalous readings, assisting the system operators to perform the system diagnosis and repair the system accordingly.

In this chapter, the ACLAE-DT framework is proposed, which is an unsupervised attention-based ConvLSTM autoencoder with dynamic thresholding to detect anomalies and identify anomalies' root cause in a manufacturing process multivariate timeseries. ACLAE-DT is a

DL-based framework that is able to capture both the temporal and contextual dependencies across different time steps in the multivariate time series data, while being robust to noise. The framework starts by normalizing the input data via Min-Max scaling to scale the values to a fixed range. Post pre-processing, the data is then enriched using sliding windows to be able to capture more temporal behavior via the lagged values, followed by the incorporation of contextual manufacturing process data via an embedding layer to capture the contextual information. Afterwards, ACLAE-DT constructs feature images based on the processed and enriched data, which are matrices of inner-products between a pair of time series within the sliding window segments. Feature images characterizes the system statuses across different time steps by capturing shape similarities and inter-correlations between two time series across different time steps, while being robust to noise. Subsequently, an attention-based ConvLSTM autoencoder is employed to encode the constructed feature images and capture the temporal behavior, followed by decoding the compressed knowledge representation to reconstruct the feature images input. The structures that exist in the data are learned and consequently leveraged when forcing the input through the autoencoder's bottleneck. Attention is added to the autoencoder to sustain a constant performance as the input time-series sequences increase, reducing model errors [18]. Furthermore, several hyperparameters are optimized via random search in order to enhance the model's performance [19]. A dynamic thresholding mechanism is then utilized against the computed reconstruction errors to detect and diagnose the anomalies, where the threshold is dynamically updated based on statistical derivations from the reconstructed normal data errors. The intuition is that ACLAE-DT will not be able to reconstruct the feature images well if it never observed similar system statuses before, resulting in anomalous processes to be flagged as it crosses the computed threshold. ACLAE-DT underwent rigorous empirical analysis, where results demonstrated the superior performance of the proposed approach over state-of-the-art methods.

The work presented in this chapter is able to capture both the temporal and contextual dependencies across different time steps in the multivariate time series data in a manufacturing

process to detect anomalies and identify the anomalies' root cause. The main contributions of this chapter include:

- A novel framework that consists of pre-processing and enriching the multivariate time series, constructing feature images, an attention-based ConvLSTM network autoencoder to reconstruct the feature images input, and a dynamic thresholding mechanism to detect anomalies and identify anomalies' root cause in multivariate time series.
- A generic, unsupervised learning framework that utilizes state-of-the-art DL algorithms and can be applied for various different multivariate time series use cases in smart manufacturing.
- An attention-based, time-distributed ConvLSTM encoder-decoder model that is capable of sustaining a constant performance as the rate of input time-series sequences from the manufacturing operations increase.
- A nonparametric and dynamic thresholding mechanism for evaluating reconstruction errors that addresses non-stationarity, noise and diversity issues in the data.
- A robust framework evaluated on a real-life public manufacturing data set, where results demonstrated its performance strengths over state-of-the-art methods under various experimental settings.

The remainder of this chapter is structured as follows. Section 3.2 presents the motivations behind the use of DL in smart manufacturing and explores related work. Section 3.3 discusses the technical preliminaries of the key concepts used in this chapter. Section 3.4 details the methodology and implementation of the ACLAE-DT framework. Section 3.5 describes the data set used, the different experimental setups, and the comparison benchmarks. Section 3.6 discusses the obtained results and performance evaluation. Finally, Section 3.7 concludes the chapter and discusses opportunities for future work.

3.2 Motivation and Related Work

Robotic surface finishing is one of the most important applications in smart manufacturing. The goal of robotic surface finishing is to manufacture products without any defects and with the adequate amount of surface roughness, in order to ensure smooth functional and financial operations. However, potential system or production failures might occur at any point in time, demonstrating the importance of having an efficient anomaly detection algorithm in place overseeing different parts of the system to detect and mitigate any manufacturing malfunctions.

Let us consider a robotic surface finishing system designed and calibrated to produce a finished metal car part. Multiple sensors will be mounted on the system at different locations in order to gather different types of readings, such as the feed rate, spindle rate, and torque. As the robotic machine is surface finishing the car part, all the sensor readings are being processed in order to monitor the system's performance. In the case of an imperfect finished car part or an anomalous system behavior, the anomaly detection algorithm will trigger an alarm and would ideally locate the anomalous sensor readings or system behavior, in an effort to notify the system operators and enable them to solve the underlying issues.

Many classical methods have been used to detect anomalies, such as probabilistic approaches [20], distance-based approaches [21], clustering approaches [22], and predictive approaches [23]. These approaches may be computationally incomplex, however, their performance is sub-optimal as they fail to capture complex structures in the data without the input of domain experts [16]. Furthermore, as the data volume increases, traditional approaches can fail to scale up as required to maintain their anomaly detection performances [16].

Moreover, ML algorithms were proposed to resolve the limitations in classical methods for anomaly detection. ML algorithms include K-Nearest Neighbors (K-NNs) [24], Support Vector Machines (SVMs) [25], and neural networks [26]. These algorithms are all supervised learning-based, meaning they rely on labeled normal and anomalous historical data for training

[27]. However, collecting labeled anomalous data is often infeasible, as with well-optimized processes, there is a often a high imbalance in the training data due to the abundance of non-anomalous samples and a relatively small or no amount of anomalous samples. Furthermore, ML approaches often require human input for feature extraction, where they start to fail in dealing with the dimensionality and variety of the data as data volume and velocity increases [14].

DL techniques have emerged in recent literature to solve the aforementioned challenges. Although ML is a data-driven AI technique to model input and output relationships, DL has distinctive advantages over ML in terms of feature learning, model construction, and model training. DL transforms data into abstract representations, allowing hierarchical discriminative features to be learned, which eliminates any manual feature development by domain experts. Moreover, DL integrates model construction and feature learning into a single model and trains the model's parameters jointly, creating an end-to-end learning structure with minimal human interference.

There has been an increase in available types of sensory data collected from various distinct aspects across the operational system in smart manufacturing. As a result, data modelling and analysis are vital tasks in order to handle the large data volume increase and the real-time data processing to detect any system anomalies; tasks which DL excels in [28]. Different DL architectures were used in the literature to detect anomalies in smart manufacturing, such as CNNs [29], LSTM neural networks [30], and autoencoders [31]. A CNN uses convolution in place of general matrix multiplication in at least one of their layers, and reduces the number of parameters very efficiently without losing out on the quality of models. This makes a CNN the prime choice for analyzing visual imagery, however, it faces difficulty in learning the high-dimensional features of the input time series as it operates in vector space. However, LSTMs, a special kind of Recurrent Neural Networks (RNNs), excel in modelling temporal behavior, as they use a feedback loop for learning and an extra parameter metric for connections between

time-steps. Leveraging the complementary strengths of CNNs and LSTMs, ConvLSTMs [17] emerged to accurately model the spatio-temporal information by having convolutional structures in both the input-to-state and state-to-state transitions.

3.3 Technical Preliminaries

3.3.1 Convolutional Neural Network (CNN)

A CNN is a feedforward deep neural network that is most commonly applied to analyzing visual imagery [32], having a wide range of applications such as image and video recognition, image classification, image segmentation, and recommender systems. A CNN uses convolution in place of general matrix multiplication in at least one of their layers, and reduces the number of parameters very efficiently without losing out on the quality of models. Figure 3.1 visualizes the structure of a simple CNN.

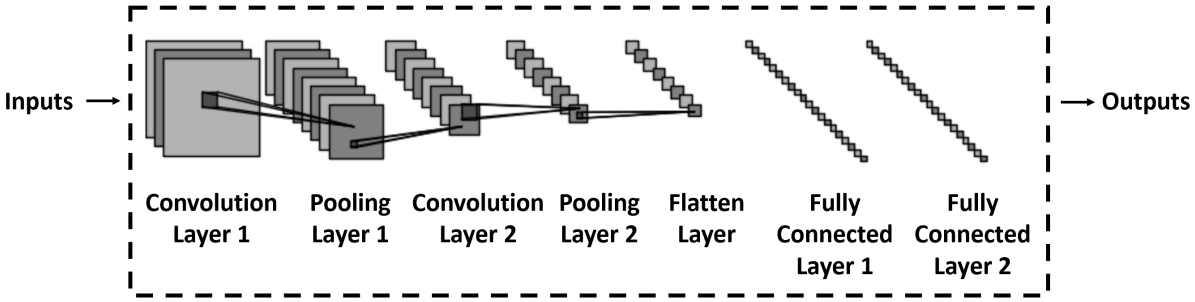

Figure 3.1: Structure of a Simple CNN

The network keeps on learning new higher dimensionality and more complex features with every layer. The input data layer takes an input vector with shape ($H \times V \times D$), where H is the height, V is the width, and D is the number of channels. The input is then passed to the convolutional layer, which convolves the input and abstracts it to a feature map based on a set of weights called the kernel. In the convolutional layer, each neuron receives input from a

restricted area of the previous layer called the neuron's receptive field. The receptive field is typically a square matrix of weights with sizes that are smaller than the input. The convolution operation conducted by the receptive field along the input area is described as:

$$y_{ij} = \sigma\Big(\sum_{r=1}^{F} \sum_{k=1}^{F} w_{rk} x_{(r+i\times S)(k+j\times S)} + b \Big) \tag{3.1}$$

$$0 \le i \le \frac{H-F}{S}, 0 \le j \le \frac{V-F}{S} \tag{3.2}$$

where y_{ij} is the feature map output of the node, σ is the nonlinear activation function applied, F is the height and width size of the receptive field, r and k are the receptive field's width and height step respectively, w is the weight, x is the input data, S is the stride length, and b is the bias vector.

The convolutional layer applies a filter that scans the whole image a couple of pixels at a time, reducing the single input image to produce a feature map of size $((\frac{H-F+2P}{S+1}) \times (\frac{V-F+2P}{S+1}) \times K)$, where K indicates the number of filters and P indicates the padding value. The pooling layer scales down the information produced from the convolution layer by reducing the spatial dimension, yet is still able to maintain the most essential information and control overfitting. Different types of pooling include max pooling, where the maximum pixel value of the batch is selected, min pooling, where the minimum pixel value of the batch is selected, and average pooling, where the average value of all the pixels in the batch is selected. The fully connected layer connects every neuron in one layer to every neuron in another layer.

3.3.2 Long Short-Term Memory (LSTM)

An LSTM deep neural network is a special variant of RNN that excel in modelling temporal behavior, such as time-series, language, audio, and text, due to the feedback loops used for learning and the extra parameter metric available for connections between time-steps. An LSTM's main components are the memory cells and the input, forget, and output gates. These components allow the LSTM network to have connections from previous time steps and layers, where every output is influenced by the input as well as the historical inputs. Usually, there are multiple LSTM layers, where each layer is comprised of many LSTM units, and each unit comprise of input, forget, and output gates. The input gate protects the unit from irrelevant events, the forget gate helps the unit forget previous memory contents, and the output gates exposes the contents of the memory cell at the output of the LSTM unit. The equations for the input, forget, and output gates, as well as the candidate cell state, the cell state, and the LSTM cell output are described as, respectively:

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i) \tag{3.3}$$

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f) \tag{3.4}$$

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t + b_o) \tag{3.5}$$

$$\tilde{C}_t = tanh(W_{hC}h_{t-1} + W_{xC}x_t + b_C) \tag{3.6}$$

$$C_t = f_tC_{t-1} + (1 - f_t)\tilde{C}_t \tag{3.7}$$

$$h_t = o_t tanh(C_t) \tag{3.8}$$

where i is the input gate, f is the forget gate, o is the output gate, $\tilde{C}_t$ is the candidate cell state, C is the cell state, h is the hidden state and cell output, σ denotes a standard logistic sigmoid function, W is the weight matrix, and b is the bias vector. Figure 3.2 visualizes the

structure of an LSTM memory cell.

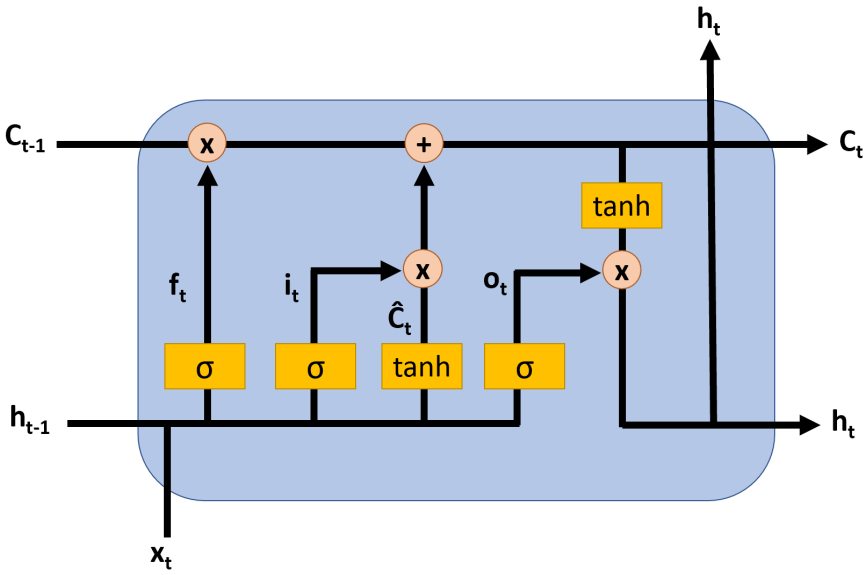

Figure 3.2: Structure of an LSTM Memory Cell

3.3.3 Convolutional LSTM (ConvLSTM)

ConvLSTMs [17] are a special kind of LSTMs that emerged to accurately model the spatio-temporal information, by leveraging the strengths of a CNN and an LSTM. Similar to the LSTM, the ConvLSTM is able to decide what information to discard or retain from the previous cell state in its present cell state. However, convolutional structures are utilized in both the input-to-state and state-to-state transitions, which basically exchanges the internal matrix multiplications with convolution operations. The input vector to the ConvLSTM is fed as a series of 2-D or 3-D images, as the convolution operations allows the data that flows through the ConvLSTM cells to keep their input dimension instead of being a 1D vector with features. Furthermore, the transition between the states is analogous to the movement between the frames in an ConvLSTM. The equations that describe the ConvLSTM operations are as follows:

$$i_t = \sigma(W_{Ci} \circ C_{t-1} + W_{hi} * h_{t-1} + W_{xi} * x_t + b_i) \tag{3.9}$$

$$f_t = \sigma(W_{Cf} \circ C_{t-1} + W_{hf} * h_{t-1} + W_{xf} * x_t + b_f) \tag{3.10}$$

$$o_t = \sigma(W_{Co} \circ C_t + W_{ho} * h_{t-1} + W_{xo} * x_t + b_o) \tag{3.11}$$

$$\tilde{C}_t = tanh(W_{hC} * h_{t-1} + W_{xC} * x_t + b_C) \tag{3.12}$$

$$C_t = f_t \circ C_{t-1} + (1 - f_t) \circ \tilde{C}_t \tag{3.13}$$

$$h_t = o_t \circ tanh(C_t) \tag{3.14}$$

where $\circ$ represents the Hadamard product, $*$ represents the convolutional operator, W_{Ci}, W_{hi}, W_{xi}, W_{Cf}, W_{hf}, W_{xf}, W_{Co}, W_{ho}, W_{xo}, W_{hC}, $W_{xC} \in \mathbb{R}^{n \times T}$ represent the convolutional kernels within the model and b_i, b_f, b_o, b_C are the bias parameters. Figure 3.3 visualizes the structure of a ConvLSTM, where the red lines indicate the peephole connections found in a conventional ConvLSTM cell but not in a conventional LSTM cell.

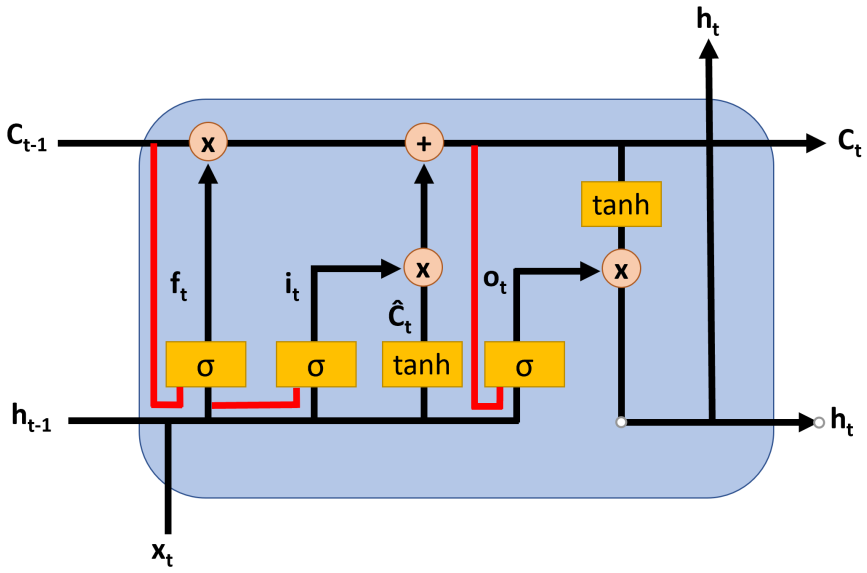

Figure 3.3: Structure of a ConvLSTM Memory Cell

3.3.4 Autoencoder

An autoencoder is an unsupervised feedforward neural network that imposes a bottleneck in the network, forcing a compressed knowledge representation of the original input. More specifically, an autoencoder learns how to efficiently compress and encode data, before learning how to reconstruct the data back from the reduced encoded representation to an output representation that is as close to the original input as possible. An autoencoder consists of 3 main parts: the encoder, which learns how to reduce the input dimensions and compress the input data into an encoded compressed representation, the compressed representation itself, and the decoder, which learns how to reconstruct the compressed representation to be as close to the original input as possible. The network is trained to minimize the reconstruction error, $L(x, \hat{x})$, which measures the differences between the original input and the consequent reconstruction. By design, autoencoders reduce data dimensionality by learning how to ignore the noise in the data. Figure 3.4 visualizes an autoencoder example, where the original input image is encoded to a compressed representation via the encoder then the compressed representation is decoded via the decoder to produce the reconstructed input.

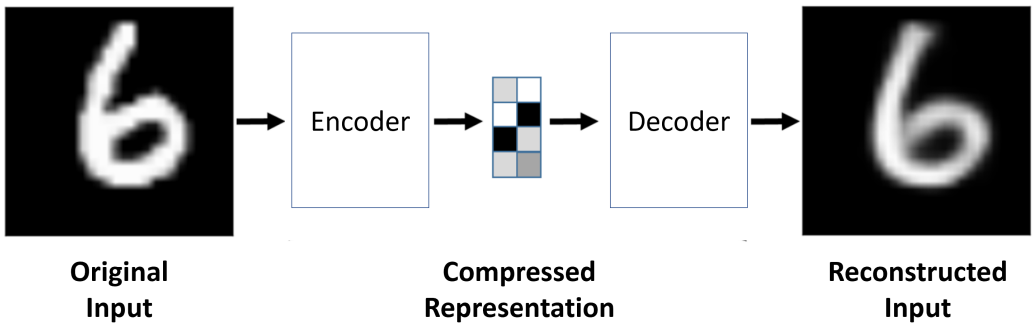

Figure 3.4: An Autoencoder Example

3.4 ACLAE-DT Framework

The following section details the ACLAE-DT framework's design, methodology, and implementation. First of all, the problem statement addressed in this work is discussed, before detailing each module in ACLAE-DT. The framework starts off by pre-processing and enriching the input raw time series, before constructing feature images across the different time steps. Afterwards, the attention-based ConvLSTM autoencoder aims to reconstruct the feature images input by minimizing the reconstruction errors. Hyperparameter optimization is conducted to improve the model's performance. Lastly, a dynamic thresholding mechanism is applied against the computed reconstruction errors for anomaly detection and diagnosis. Figure 3.5 visualizes a flowchart of the ACLAE-DT framework.

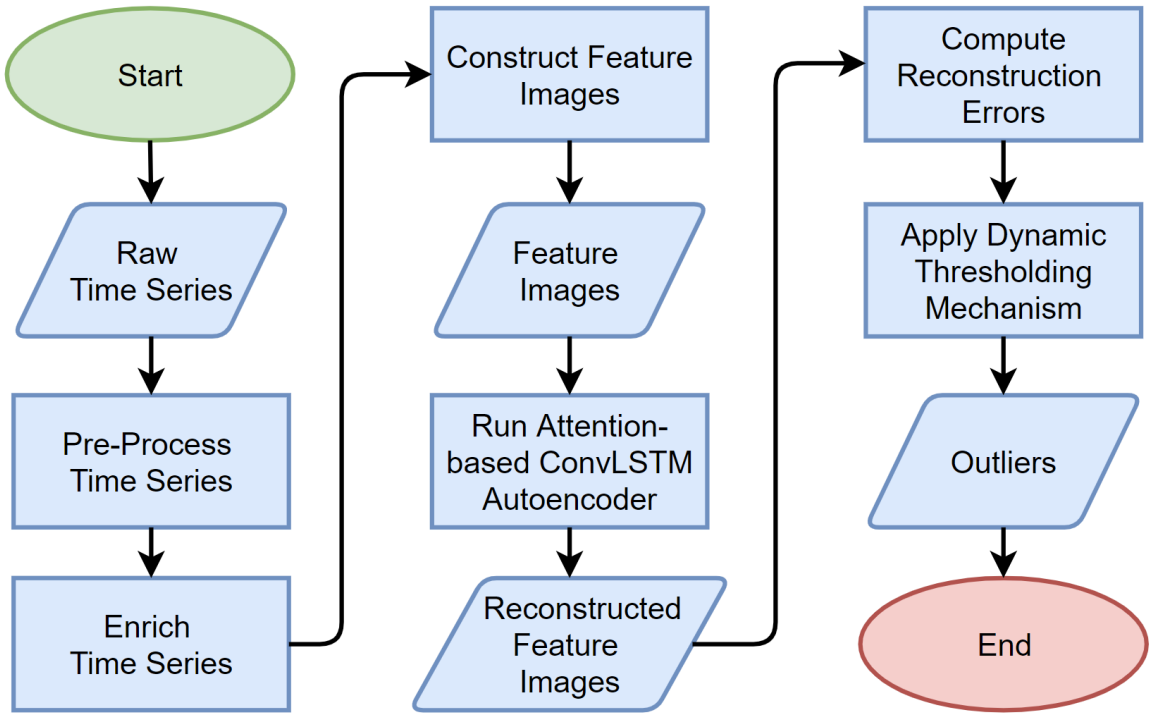

Figure 3.5: ACLAE-DT Framework Flowchart

3.4.1 Problem Statement

The historical raw multivariate time series is represented as,

$$X = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^{n \times T} \tag{3.15}$$

where x_i is a single time series, n is the number of time series, T is the length of the time series, and X is the entire time series. Assuming there are no anomalies in the training data, ACLAE-DT aims to detect anomalies by computing an anomaly score for each time step in x_i after T, such that a score outside of the threshold boundaries is flagged as an anomaly, indicating an anomalous time step. Moreover, given the anomaly detection results, ACLAE-DT aims to identify the anomalies' root cause by quantitatively and qualitatively analyzing the time series that are most likely to be the causes of the flagged anomalous time step.

3.4.2 Pre-Process Time Series

Each input raw multivariate time series is normalized via Min-Max scaling to rescale their values between 0 and 1. Min-Max scaling normalization can be implemented as:

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{3.16}$$

where x_i' is the normalized input time series value. As each time series is considered a feature, feature scaling eliminates large scaled features to be dominating, while preserving all relationships in the data [33]. Furthermore, it allows gradient descent to converge much faster when training the attention-based ConvLSTM autoencoder model [33].

3.4.3 Enrich Time Series

Utilizing Sliding Windows

Each pre-processed time series is then converted to a larger, more enriched time series of multivariate subsequences through the use of a sliding window. Sliding window indicates the creation of a specified window size and performing calculations on the data within this window, before sliding or rolling to the next window based on the step size specified and so on, till the entire data has been covered in at least a single window. This means data overlaps can occur across the different sliding windows when creating these extra sub periods and incorporating the lagged values, which can assist the attention-based ConvLSTM autoencoder to extract richer features from the constructed feature images [34]. A more formal definition of a sliding window is specified as: given a time series x_i of length T and a user-defined subsequence length of d, all possible subsequences can be extracted by a sliding window of size d across x_i and considering each subsequence s as $t - d$ to t, where t indicates a particular time position.

Embedding Contextual Information

Changes in a time series may be due to contextual changes. For example, a surface finished material in a manufacturing process could end up being scratched or bent due to the clamp pressure used to hold the workpiece in the vise rather than the x, y and z axis sensor values or the spindles. Therefore, taking contextual changes into consideration when detecting anomalies can enhance the anomaly detection performance [35].

Contextual information are usually represented as text or categorical variables. However, DL-based models are only able to process numerical values. One method to achieve the required numerical conversion includes the use of ordinal encoding, where each unique category

or text value is assigned an integer value. Nonetheless, this assumes that integer values have a natural ordered relationship between each other, which is often not the case. To resolve this shortcoming, one-hot encoding can be applied to remove the integer encoded variable and replace it with a one new binary variable for each unique integer value in the variable. However, one-hot encoding does not scale well with respect to the number of categories, as the computation cost increase significantly as the categories increase, and it does not capture similarities between the categories.

Entity embeddings [36] resolve the aforementioned limitations by using an additional mapping operation that transforms and represents each category to a low-dimensional space. The entity embedding vector or layer is a matrix of weights represented as $W_{embedding} \in \mathbb{R}^{q \times p}$, where q indicates the number of categories and p indicates the number of dimensions in the low dimensional space. In this work, given a particular category v, a one-hot representation method $onehot(v) \in \mathbb{R}^{q \times 1}$ is used. Afterwards, an embedded representation method $embedded(v) = onehot(v) \times W_{embedding}$ is used for each category. It is important to set $p < q$ to ensure that as the number of categories increase, the dimensional value set limits the computational cost from increasing. As a result, the embedded representation is smaller than the one-hot representation and is able to capture similarities between the categories. The contextual entity embeddings are attached to each time series as extra features before constructing the feature images.

3.4.4 Construct Feature Images

In order to characterize the manufacturing system status accurately, it is critical to calculate and pinpoint the correlations between the different pairs of time series [37]. Acting as an extension to the work in [38], feature images are constructed in this chapter to represent the inter-correlations between the different pairs of sensor values and contextual entity embeddings time series in a multivariate time series. More specifically, an $(n + p) \times (n + p)$ feature image

M^t is constructed for each sliding window segment s based upon the pairwise inner-product of two time series within this segment. The inter-correlation between two time series in a single segment is calculated as:

$$m_{ij}^t = \frac{\sum_{\delta=0}^{w} x_i^{t-\delta} x_j^{t-\delta}}{s} \in M^t \tag{3.17}$$

where i and j indicate two time series features, and δ indicates every single step in the segment. A feature images matrix is produced for each experiment or multivariate time series of length T, which consists of a feature image M^t for each segment s. In addition to representing the inter-correlations and shape similarities between the pairs of time series, feature images are robust to input noise, as instabilities at certain time steps have a small consequence on the feature images. Figure 3.6 visualizes a single feature image example M^t, where the x axis indicates features i and the y axis indicates features j.

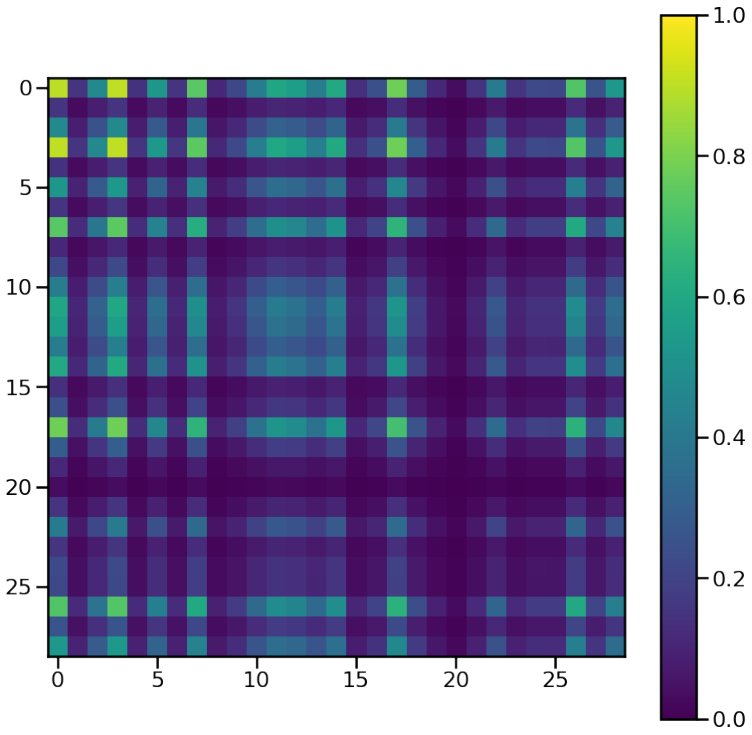

Figure 3.6: Feature Image Visualization

3.4.5 Attention-based ConvLSTM Autoencoder Model

Once all feature images has been constructed, they are input into the ConvLSTM autoencoder for reconstruction. More specifically, the autoencoder first encodes the constructed feature images and captures the temporal behavior via three alternating ConvLSTM encoding layers and MaxPool layers, followed by decoding the compressed knowledge representation to reconstruct the original feature images input via three alternating ConvLSTM decoding layers and UpSample layers. All MaxPool and UpSample layers have a size of (2x2), whereas all the ConvLSTM layers have a kernel size of (3x3) and 'same' padding. Moreover, all the ConvLSTM layers have 64 filters, except for the last ConvLSTM encoder layer and the first ConvLSTM decoder layer, which have 32 filters. To describe the different ConvLSTM layers in the model, equations (3.9) - (3.14) are rewritten as:

$$i^{t,l} = \sigma(W^l_{Ci} \circ C^{t-1,l} + W^l_{hi} * h^{t-1,l} + W^l_{xi} * x^{t,l} + b^l_i) \tag{3.18}$$

$$f^{t,l} = \sigma(W^l_{Cf} \circ C^{t-1,l} + W^l_{hf} * h^{t-1,l} + W^l_{xf} * x^{t,l} + b^l_f) \tag{3.19}$$

$$o^{t,l} = \sigma(W^l_{Co} \circ C^{t,l} + W^l_{ho} * h^{t-1,l} + W^l_{xo} * x^{t,l} + b^l_o) \tag{3.20}$$

$$\tilde{C}^{t,l} = tanh(W^l_{hC} * h^{t-1,l} + W^l_{xC} * x^{t,l} + b^l_C) \tag{3.21}$$

$$C^{t,l} = f^{t,l} \circ C^{t-1,l} + (1 - f^{t,l}) \circ \tilde{C}^{t,l} \tag{3.22}$$

$$h^{t,l} = o^{t,l} \circ tanh(C^{t,l}) \tag{3.23}$$

where l represents the ConvLSTM layer. Figure 3.7 visualizes the employed ConvLSTM autoencoder architecture.

Although ConvLSTMs have been developed to accurately model the spatio-temporal information in a sequence, its performance may deteriorate as the sequence length increases.

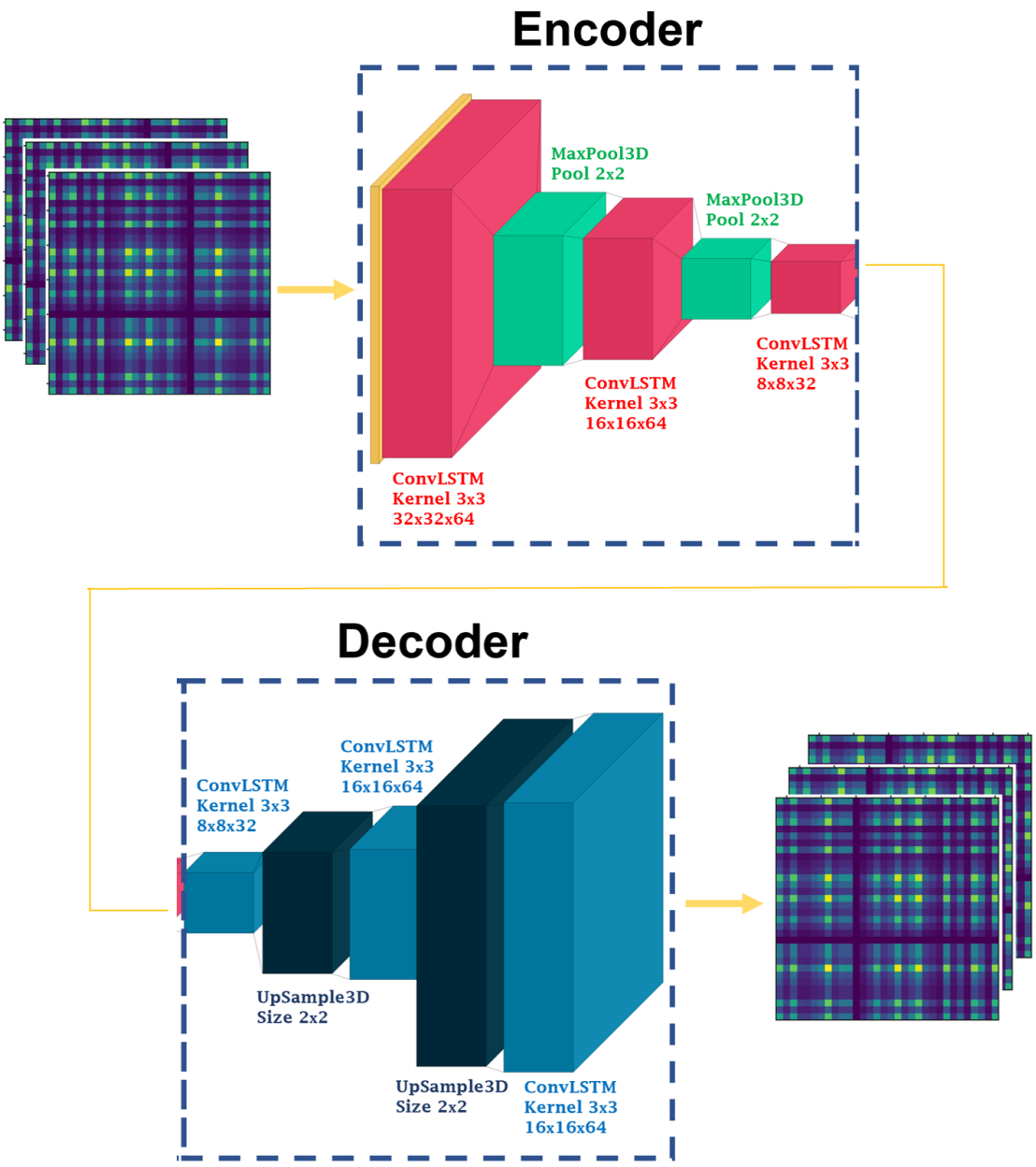

Figure 3.7: The Proposed Attention-based ConvLSTM Autoencoder Model

To overcome this issue, the Bahdanau attention [18] is added to the model, which can adaptively select relevant hidden states across different time steps and aggregate the representations of the informative feature maps to form a refined output of feature maps. A constant model performance is achieved and model errors are reduced as the input time-series sequences increase. The Bahdanau attention is the first type of attention introduced for RNNs, becoming the predominant concept in neural network literature [39]. The additive Bahdanau attention is described as:

$$c^t = \sum_{t'=1}^{T_x} \alpha^{t,t'} h^{t'} \tag{3.24}$$

$$\alpha^{t,t'} = \frac{exp(u^{t,t'})}{\sum_{k=1}^{T_x} exp(u^{t,k})} \tag{3.25}$$

$$u^{t,t'} = a(s^{t-1}, h^{t'}) \tag{3.26}$$

where c is the context vector for the sequence of hidden state annotations, α denotes the weights of each annotation, and u is the alignment model and output score of the feedforward neural network described by function a that attempts to capture the alignment between the attention-based ConvLSTM hidden state s of the previous time step $t-1$ and the t'-th annotation from the hidden state h of the input sequence.

3.4.6 Model's Hyperparameter Optimization

In order to enhance the performance of the model, several hyperparameters need to be tuned and optimized [19]. Model hyperparameters are used in processes to help estimate the model parameters, which are learned and estimated during the training process of minimizing an objective loss function. Several hyperparameter optimization methods exist, such as grid search,

random search, and gradient-based optimization. Grid search is an exhaustive search through a set of manually specified set of hyperparameter values, which is time-consuming and impacted by the curse of dimensionality [40]. Gradient-based optimization utilizes the gradient descent algorithm to compute the gradient with respect to hyperparameters, but they only support continuous hyperparameters and can only detect a local optimum for non-convex hyperparameter optimization problems rather than a global optimum [41]. Finally, random search randomly searches the grid space and supports all types of hyperparameters, allowing a larger and more diverse grid space to be explored. Hence, random search is used as the optimization method in this work as it is more computationally efficient than grid search and gradient-based optimization [19].

Five hyperparameters are optimized in this work, as presented in Table 3.1 alongside all the values of consideration and in which layer they lie, if applicable. The selected hyperparameters are considered to be the most influential five hyperparameters on the model's performance, based on initial experiments. All hyperparameter value options are based off initial experiments as well and off the most common used values in the DL domain [19]. Activation function values considered include Rectified Linear Units (ReLU) [42], Leaky ReLU [43], Exponential Linear Units (ELU) [44], and Scaled Exponential Linear Unit (SELU) [45]. Optimization algorithms considered include Adam [46], RMSProp [47], AdaDelta [48], and Stochastic Gradient Descent (SGD) [49]. Finally, the loss functions considered include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Layer	**Hyperparameter**	**Values**
ConvLSTM	Activation Function	ReLU, Leaky RELU, ELU, SELU
N/A	Learning Rate	1e-2, 1e-3, 1e-4, 1e-5, 1e-6
N/A	Batch Size	16, 32, 64, 128, 256
N/A	Optimizer	Adam, RMSProp, ADADelta, SGD
N/A	Loss Function	MAE, MSE, RMSE

Table 3.1: Model Hyperparameters

3.4.7 Compute Reconstruction Errors

The model's loss objective in this work is to minimize the reconstruction errors of the normal feature images during training, in order to accurately reconstruct the normal feature images and inaccurately reconstruct the anomalous feature images during the testing phase. The loss function used is dependent on the random search hyperparameter optimization method output for the loss function hyperparameter, with options being MAE, MSE, and RMSE. In general terms, the model is trained to minimize the reconstruction error, $L(x_i, \hat{x}_i)$, which measures the differences between the original input and the consequent reconstruction.

3.4.8 Dynamic Thresholding Mechanism

At this stage, an efficient anomaly thresholding mechanism is needed in order to detect anomalous reconstructed feature images. Often, the anomaly threshold is learned with supervised methods, however, as the nature of the data in the manufacturing domain is continuously changing and there is insufficient labeled data for each class, supervised methods would not be optimal for use here [50]. Hence, a nonparametric and dynamic anomaly thresholding mechanism is proposed in this work, which calculates a different threshold for each time series pair based on statistical derivations, achieving high anomaly detection performance with low overhead. More specifically, a single threshold is set against every single time series pair in the feature image, based on the mean and standard deviation of that specific normal time series pair reconstruction errors. Any time series pair value that surpasses the threshold at any time step during testing will be flagged as anomalous and will flag the entire process at that time step as anomalous. Mathematically, the method is described as:

$$\epsilon_{ij} = \left(\mu(e_{ij}) + z\sigma(e_{ij})\right)^T \in \epsilon \tag{3.27}$$

where ϵ_{ij} indicates the threshold value for the i and j time series features pair across the entire time series, ϵ is the $(n+p) \times (n+p)$ threshold matrix, e_{ij} is the set of reconstruction errors for the normal i and j time series features pair, μ is the mean, σ is the standard deviation, and z is an ordered set of positive values representing the number of standard deviations. Values for z depend on context, with a range of two to five found to produce the most accurate experimental results in this work. The presented dynamic anomaly detection thresholding mechanism detects outliers as well as localizes the anomaly root cause, by pinpointing the sensors or components that are causing the detected outlier.

3.5 Experiments

The data set used in this chapter is the Computer Numerical Control (CNC) Mill Tool Wear data set provided by the University of Michigan [51] and found on Kaggle [52]. The data set consisted of a series of machining experiments run on 2" x 2" x 1.5" wax blocks in a CNC milling machine in the System-level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan. Machining data was collected from a CNC machine for variations of feed rate, tool condition, and clamping pressure, where each experiment produced a finished wax part with an "S" shape as visualized in Figure 3.8. 44 time series readings from the 4 motors in the CNC machine, the X-axis, Y-axis, Z-axis, and spindle (S-axis), were collected at a sampling rate of 10 Hz. The data set contained a total of 25,286 time series measurements from 18 experiments conducted, where 4 of these experiments failed the visual inspection check.

In this chapter, every measurement is taken as an independent observation within a sliding window to identify normal or anomalous behavior and to pinpoint the sensors that flagged windows as anomalous. Data from 10 normal experiments were used for training, whereas data from the remaining 4 normal experiments and from the 4 anomalous experiments were used for testing.

Figure 3.8: Data Set Test Artifact [51]

In order to evaluate ACLAE-DT's anomaly detection performance, the attention-based ConvLSTM autoencoder model is compared with seven baseline methods. The baseline methods comprise of an ML classification method, a classical forecasting method, three state-of-the-art DL methods, and two variants of ACLAE-DT. The classical and ML methods are evaluated to demonstrate the effectiveness of using a DL model, the baseline DL methods are evaluated to demonstrate the effectiveness of a ConvLSTM autoencoder, and the two variants of ACLAE-DT are evaluated to demonstrate the effectiveness of each component within the model. The same number of layers, hyperparameters, and components are used for each DL method, if applicable. The baseline methods are:

1. SVM: An ML method that classifies whether a test data point is an anomaly or not based on the learned decision function from the training data.

2. Auto Regressive Integrated Moving Average (ARIMA): A classical prediction model that captures the temporal dependencies in the training data to forecast the predicted values of the testing data.

3. LSTM Autoencoder: A DL method that utilizes LSTM networks in both the encoder and decoder.

4. ConvLSTM-LSTM Autoencoder: A DL method that utilizes ConvLSTM networks in the encoder and LSTM networks in the decoder.

5. CNN-LSTM Autoencoder: A DL method that utilizes CNN-LSTM networks in both the encoder and decoder.

6. ACLAE-DT Shallow: An ACLAE-DT variant that utilizes ACLAE-DT's model without the last MaxPool3D and ConvLSTM encoder components and without the first UpSample3D and ConvLSTM decoder components.

7. ACLAE-DT No-Attention: An ACLAE-DT variant that utilizes ACLAE-DT's model without attention.

To empirically examine the models, three different experiments are conducted. The models are tested using a window size of 10 with a step size of 2 in Experiment 1, a window size of 30 with a step size of 5 in Experiment 2, and a window size of 60 with a step size of 10 in Experiment 3. These are common window sizes used in the literature [38]. All DL-based models are trained for 250 epochs. Moreover, comparison metrics are employed to evaluate the models used and compare their anomaly detection performances. In order to fully capture the values of the true and false positives and negatives for each model, the precision, recall, and F1 score metrics are utilized, as well as the time taken to train each model. An anomalous window is defined as a poorly reconstructed feature image with a value that surpasses the corresponding threshold. True positives in this work indicate anomalous windows correctly classified as anomalous and true negatives indicate non-anomalous windows correctly classified as non-anomalous. All experiments are repeated five times and the average results are computed for performance comparison.

Afterwards, ACLAE-DT's results are analyzed to pinpoint the readings that flagged windows as anomalous. It is important to localize the anomaly root cause during a production failure in order to plan adequate countermeasures and fix the production system. All networks

are built and implemented in Python 3.7.9, using the Tensorflow [53] and Keras [54] libraries. All work is run on a machine which comprises of an NVIDIA GeForce GTX 1650 4GB, a 16GB DDR4 2666MHz RAM, and a 9th Generation Intel Core i7-9750H Processor.

3.6 Performance Evaluation

3.6.1 Anomaly Detection Results

The anomaly detection performance for each model under the three different experimental settings are illustrated in Tables 3.2, 3.3 and 3.4, respectively, and visualized in Figure 3.9. Note that the results for ARIMA and SVM are the same across all three experiments, as they do not consider window sizes and step sizes in their algorithmic calculations. Table 3.2 demonstrates the performance evaluation of all eight models in Experiment 1. It can be observed that ARIMA detected anomalies better than SVM, indicating that the data set had a temporal dependency feature that cannot be captured by the classification method. However, all the DL-based methods performed better than ARIMA, indicating DL's strength in capturing more complex structures and model a finer multivariate temporal dependency and correlation from the data set. Furthermore, it can be observed that all variants of ACLAE-DT performed greater than the three baseline DL models based on every single evaluation metric used, while taking less time to train. The full ACLAE-DT model performed at least 4.8% better in every single evaluation metric, while taking at least 22.9% less time to train than the three baseline DL models. Moreover, the full ACLAE-DT model performed either similarly or better than the two variant baseline models, while taking 6.7% less time to train than the shallow model but 3.1% more time than the no-attention model.

Similar observations can be drawn from Tables 3.3 and 3.4, which demonstrate the performance evaluation of all eight models in Experiment 2 and Experiment 3, respectively. As the

Method	**Precision**	**Recall**	**F1-Score**	**Train Time (s)**
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.83	0.80	0.82	13,468
ConvLSTM-LSTM Autoencoder	0.80	0.84	0.82	11,914
CNN-LSTM Autoencoder	0.84	0.84	0.84	10,136
ACLAE-DT Shallow	0.94	0.87	0.90	8,372
ACLAE-DT No Attention	0.95	0.87	0.91	7,574
ACLAE-DT Full	0.95	0.88	0.92	7,812

Table 3.2: Anomaly Detection Results Using a Window Size of 10 with a Step Size of 2

Method	**Precision**	**Recall**	**F1-Score**	**Train Time (s)**
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.82	0.83	0.83	15,932
ConvLSTM-LSTM Autoencoder	0.79	0.84	0.81	8,274
CNN-LSTM Autoencoder	0.83	0.85	0.84	5,362
ACLAE-DT Shallow	0.91	0.89	0.90	3,388
ACLAE-DT No Attention	0.95	0.88	0.90	3,122
ACLAE-DT Full	0.96	0.90	0.93	3,234

Table 3.3: Anomaly Detection Results Using a Window Size of 30 with a Step Size of 5

Method	Precision	Recall	F1-Score	Train Time (s)
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.79	0.83	0.82	7,462
ConvLSTM-LSTM Autoencoder	0.77	0.82	0.79	13,496
CNN-LSTM Autoencoder	0.84	0.85	0.85	5,152
ACLAE-DT Shallow	0.96	0.99	0.97	2,814
ACLAE-DT No Attention	0.97	0.99	0.98	1,638
ACLAE-DT Full	0.99	1.00	1.00	1,736

Table 3.4: Anomaly Detection Results Using a Window Size of 60 with a Step Size of 10

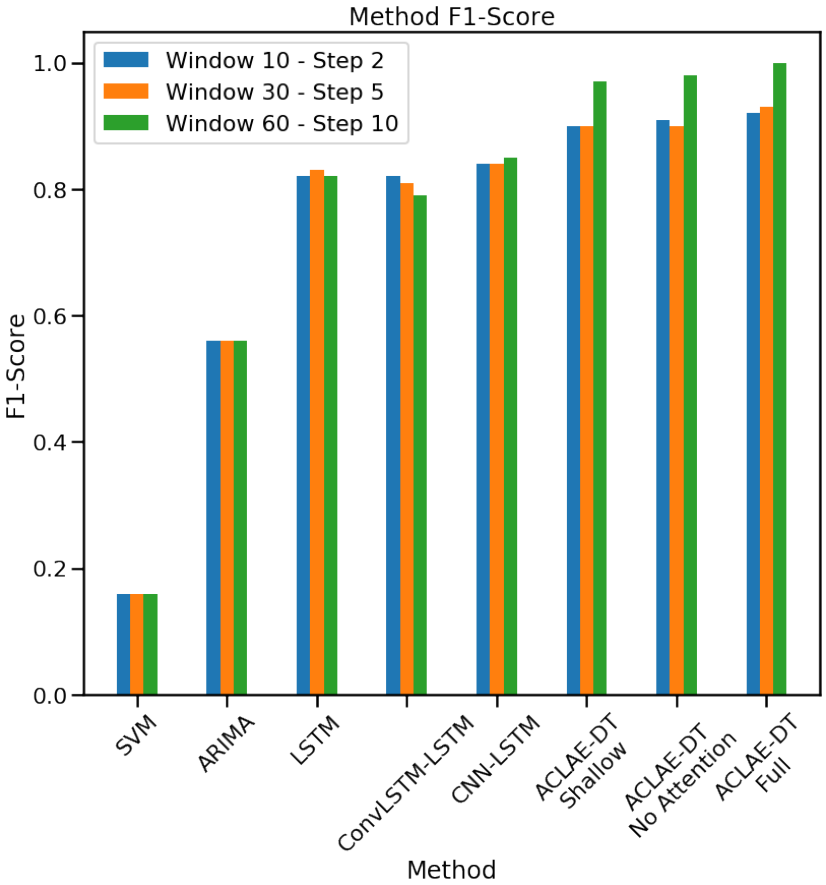

Figure 3.9: Method F1-Score Under Each Experimental Setting

window size and step size increased, the training time for all ACLAE-DT variants decreased, whilst average performance improved. This was not the case with the DL baseline models however, as training time and performance did not follow a general trend as window sizes and step sizes changed. In Experiment 2, the full ACLAE-DT model performed at least 5.8% better in every single evaluation metric, while taking at least 65.8% less time to train than the three baseline DL models. In Experiment 3, the full ACLAE-DT model performed at least 16.5% better in every single evaluation metric, while taking at least 196.7% less time to train than the three baseline DL models. In both experiments, the full ACLAE-DT model performed better than the two variant baseline models, while taking less time to train than the shallow model but more time than the no-attention model.

All the previous results demonstrate the strength of utilizing a DL-based anomaly detection model in multivariate time series, as SVM and ARIMA failed to capture complex relationships and detect anomalies appropriately. Moreover, the results demonstrate the effectiveness of deploying ConvLSTM networks in both the encoder and decoder in an autoencoder compared to deploying LSTM networks or CNNs in either the encoder or decoder, as ACLAE-DT was capable of capturing the inter-sensor correlations and temporal patterns of multivariate time series effectively. The results further demonstrate the effectiveness of constructing a deeper model and adding attention to it, particularly for when the window size is 30 and above, as performance constantly improved. The full ACLAE-DT model and all its variants performed the best in Experiment 3, whilst taking the least amount of training time. The full ACLAE-DT model had the best model performance out of all the compared models in the aforementioned experimental setting, scoring a perfect recall and F1-score, and close to perfect precision score, with a total average training time of 1,736 seconds.

3.6.2 Anomaly Root Cause Identification Results

If the reconstruction error of an inter-correlation between two time series crossed the set threshold for a particular window, then the corresponding pair of sensors were signified as contributors towards the anomalous window. The three sensor readings and univariate time series measurements that contributed the most towards the flagged anomalous windows in Experiment 3 are visualized in Figure 3.10. The x-axis indicates the sensor readings, or features, and the y-axis indicates the feature's anomalous window appearance percentage. It can be observed from the figure that the readings from the X-axis motor had the greatest influence on the success of the visual inspection check as they contributed the most towards flagging a window as anomalous. The X1-OutputCurrent feature had the greatest influence as it passed its threshold in 95.7% of the windows, followed by the X1-DCBusVoltage feature as it passed its threshold in 88.2% of the windows, followed by the X1-ActualAcceleration feature as it passed its threshold in 78.7% of the windows.

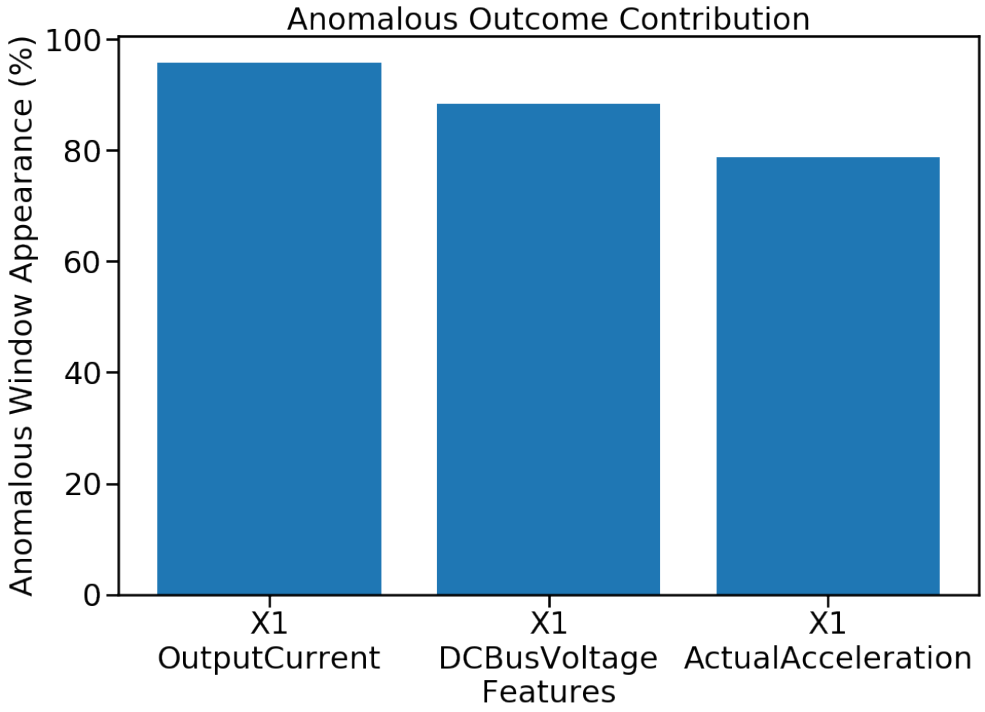

Figure 3.10: Anomaly Root Cause Feature Analysis

X1-OutputCurrent was further analyzed in order to have a thorough understanding of ACLAE-DT's mechanism and results. Figure 3.11 visualizes three charts within a specific time series cross-section: (a) X1-OutputCurrent original vs reconstructed normal data, (b) X1-OutputCurrent original vs reconstructed anomalous data, and (c) X1-OutputCurrent reconstructed normal vs anomalous data errors with the calculated dynamic threshold boundary in red. In chart (a), it can be observed that ACLAE-DT was able to reconstruct the original normal data well for most data points with a small margin of error. In chart (b), it can be observed that ACLAE-DT was not able to reconstruct the original anomalous data well, particularly for the reading peaks, as it never observed similar system statuses before. Finally, in chart (c), it can be realized that the reconstructed anomalous data errors crossed the threshold frequently, whereas the reconstructed normal data errors never crossed the threshold. It is important to note that many of the reconstructed anomalous errors were just shy of crossing the threshold, indicating that when the inter-correlation between X1-OutputCurrent and another time series was computed, the results were bound to cross the set threshold if the time series contained any novel system behavior, contributing to X1-OutputCurrent's high anomalous correlation.

3.6.3 Execution Time and Memory Requirements

To further evaluate ACLAE-DT against the baseline methods, the execution time and memory requirements of each method for a single window in Experiment 3 were calculated as visualized in Figures 3.12 and 3.13, respectively. Ten experimental executions were conducted and the average results were used. SVM and ARIMA were not included in the comparison due to their poor anomaly detection performances, deeming both methods unsuitable for use in real-life smart manufacturing processes.

It can be observed from Figure 3.12 that the LSTM autoencoder method took the shortest

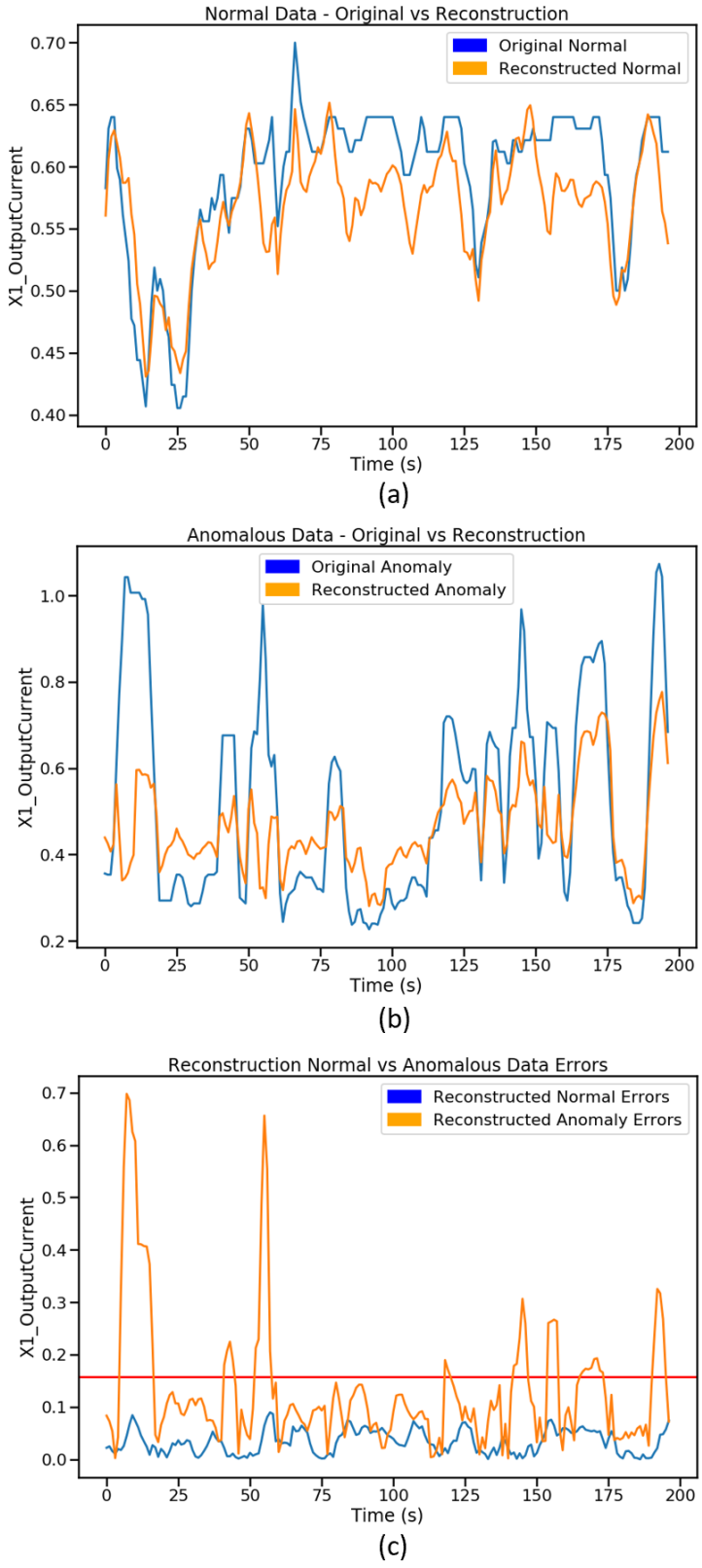

Figure 3.11: X1-OutputCurrent Time Series Measurement Analysis

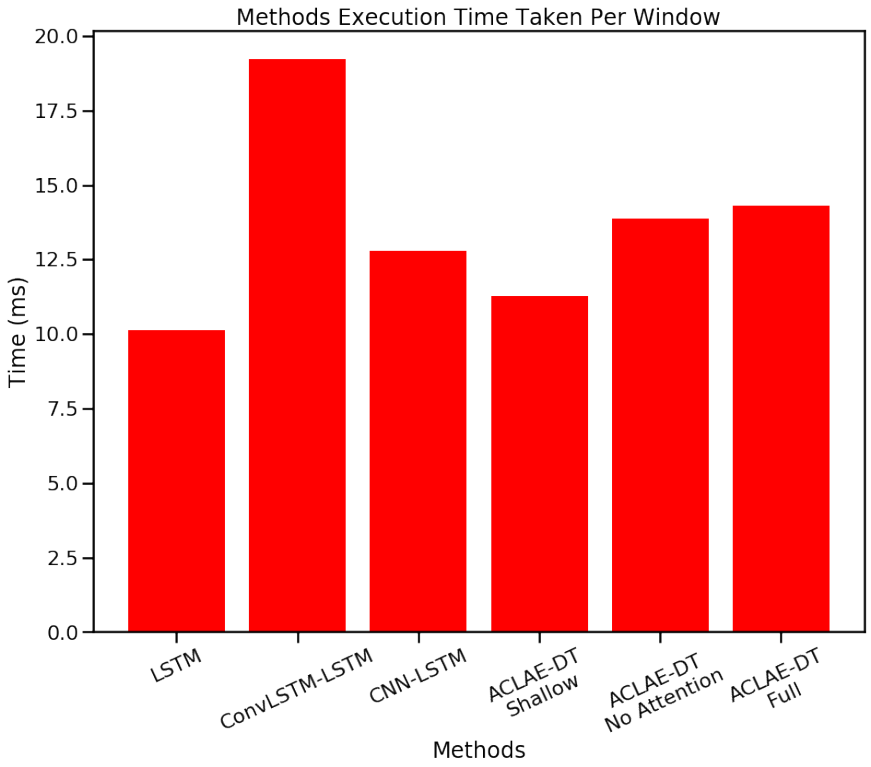

Figure 3.12: Execution Time Taken Per Window

Figure 3.13: Execution Memory Consumption Per Window

execution time per window with around 10 ms, followed by the ACLAE-DT shallow method with around 11.25 ms. The ACLAE-DT full method took around 14 ms, 40% more time than the LSTM autoencoder method and 24.4% more time than the ACLAE-DT shallow method. Moreover, it can be drawn from Figure 3.13 that the ACLAE-DT shallow method required the least amount of memory during execution with around 0.14 MB. ACLAE-DT no-attention and ACLAE-DT full required the most amount of memory out of all methods, with memory consumptions of around 0.25 MB and 0.26 MB, respectively. From the drawn observations, it is evident that the tradeoff of using the full ACLAE-DT method with its superior anomaly detection performance, root cause detection identification, and short training time, is its greater execution time and memory consumption. If a method's execution time and memory consumption is of a higher importance than the method's anomaly detection performance and training time during real-life production, then the ACLAE-DT shallow method can be utilized as it performed very closely to the ACLAE-DT full method and required the second least amount of execution time and the least amount of execution memory out of all methods.

3.7 Conclusion

In this chapter, a novel unsupervised attention-based deep ConvLSTM autoencoder with a dynamic thresholding mechanism framework, ACLAE-DT, was proposed to detect anomalies in a real-life manufacturing multivariate time series data set. The framework first normalized and enriched the raw time series with contextual information and sliding windows, before constructing feature images to capture system statuses across different time steps. The feature images were then input into an attention-based deep ConvLSTM autoencoder to be reconstructed, with an aim to minimize the reconstruction errors. The computed reconstruction errors were then subjected to a dynamic, nonparametric thresholding mechanism that utilized the mean and standard deviation of the normal reconstruction errors to compute a specific threshold for each

time series pair, in order to detect and diagnose the anomalies.

Results demonstrated the effectiveness of ACLAE-DT, as it outperformed a classical approach, an ML approach, and three state-of-the-art DL approaches in detecting anomalous windows, while requiring less time to train than the latter approaches. Results further illustrated how ACLAE-DT was able to effectively diagnose the anomalies and locate the contributing features towards the anomalous windows. Moreover, the shallow variation of ACLAE-DT consumed the least amount of execution memory and the second least amount of execution time out the three state-of-the-art DL methods. All these results indicated the practicality and suitability of adopting ACLAE-DT in real life smart manufacturing processes. As a future extension to this work, ACLAE-DT can be applied to another public data set to benchmark its performance with the conventional anomaly detection algorithms, and a reduction in the execution time and memory consumption of the full ACLAE-DT model while maintaining the superior anomaly detection and anomaly root cause identification performances can be explored.

References

[1] Andrew Kusiak. Smart manufacturing. *International Journal of Production Research*, 56(1-2):508–517, 2018.

[2] Hyoung Seok Kang, Ju Yeon Lee, SangSu Choi, Hyun Kim, Jun Hee Park, Ji Yeon Son, Bo Hyun Kim, and Sang Do Noh. Smart manufacturing: Past research, present findings, and future directions. *International journal of precision engineering and manufacturing-green technology*, 3(1):111–128, 2016.

[3] Ivan Russo, Ilenia Confente, David M Gligor, and Nicola Cobelli. The combined effect of product returns experience and switching costs on B2B customer re-purchase intent. *Journal of Business & Industrial Marketing*, 32(5):664–676, 2017.

[4] Progress. Anomaly Detection & Prediction Decoded: 6 Industries, Copious Challenges, Extraordinary Impact, https://www.progress.com/docs/default-source/datarpm/progress_datarpm_cadp_ebook_anomaly_detection_in_6_industries.pdf?sfvrsn=82a183de_2, 2017.

[5] Tianqi Yu, Xianbin Wang, and Abdallah Shami. Recursive principal component analysis-based data outlier detection and sensor data aggregation in IoT systems. *IEEE Internet of Things Journal*, 4(6):2207–2216, 2017.

[6] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Transactions on Network and Service Management*, pages 1–1, 2020.

[7] Li Yang, Abdallah Moubayed, Ismail Hamieh, and Abdallah Shami. Tree-based intelligent intrusion detection system in internet of vehicles. In *2019 IEEE global communications conference (GLOBECOM)*, 2019, pp. 1–6.

[8] Li Yang and Abdallah Shami. A lightweight concept drift detection and adaptation framework for iot data streams. *IEEE Internet of Things Magazine*, pages 1–6, 2021.

[9] Tung Kieu, Bin Yang, and Christian S Jensen. Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, 2018, pp. 125–134.

[10] Sulaiman Aburakhia, Tareq Tayeh, Ryan Myers, and Abdallah Shami. A transfer learning framework for anomaly detection using model of normality. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0055–0061.

[11] Yifan Guo, Weixian Liao, Qianlong Wang, Lixing Yu, Tianxi Ji, and Pan Li. Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. In *Asian Conference on Machine Learning*, 2018, pp. 97–112.

[12] Li Yang, Abdallah Moubayed, and Abdallah Shami. Mth-ids: A multi-tiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. vol. 13, Cambridge, MA, USA: MIT Press, 2016.

[14] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.*, 48:144–156, 2018.

[15] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019.

[16] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[17] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[19] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[20] Anas Saci, Arafat Al-Dweik, and Abdallah Shami. Autocorrelation integrated gaussian based anomaly detection using sensory data in industrial manufacturing. *IEEE Sensors Journal*, 21(7):9231–9241, 2021.

[21] Wunjun Huo, Wei Wang, and Wen Li. Anomalydetect: An online distance-based anomaly detection algorithm. In *International Conference on Web Services*, 2019, pp. 63–79.

[22] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, 100:106919, 2021.

[23] Kaan Gokcesu and Suleyman S Kozat. Online anomaly detection with minimax optimal density estimation in nonstationary environments. *IEEE Transactions on Signal Processing*, 66(5):1213–1227, 2017.

[24] Miao Xie, Jiankun Hu, Song Han, and Hsiao-Hwa Chen. Scalable hypergrid k-NN-based online anomaly detection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1661–1670, 2012.

[25] Yanxin Wang, Johnny Wong, and Andrew Miner. Anomaly intrusion detection using one class SVM. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop*, 2004, pp. 358–364.

[26] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. *Advances in neural information processing systems*, pages 943–949, 1998.

[27] Waqas Rasheed and Tong Boon Tang. Anomaly detection of moderate traumatic brain injury using auto-regularized multi-instance one-class SVM. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(1):83–93, 2019.

[28] Andrew Kusiak. Smart manufacturing must embrace big data. *Nature News*, 544(7648):23, 2017.

[29] Tareq Tayeh, Sulaiman Aburakhia, Ryan Myers, and Abdallah Shami. Distance-based anomaly detection for industrial surfaces using triplet networks. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0372–0377.

[30] Cheng Feng, Tingting Li, and Deeph Chana. Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 261–272.

[31] Barış Bayram, Taha Berkay Duman, and Gökhan Ince. Real time detection of acoustic anomalies in industrial processes using sequential autoencoders. *Expert Systems*, 38(1):e12564, 2021.

[32] Maria V Valueva, NN Nagornov, Pave A Lyakhov, Georgiy V Valuev, and Nikolay I Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[33] T Jayalakshmi and A Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):1793–8201, 2011.

[34] Yufeng Yu, Yuelong Zhu, Shijin Li, and Dingsheng Wan. Time series outlier detection based on sliding window prediction. *Mathematical problems in Engineering*, 2014, 2014.

[35] Manish Gupta, Abhishek B Sharma, Haifeng Chen, and Guofei Jiang. Context-aware time series anomaly detection for complex systems. In *Workshop Notes*, 2013, vol. 14.

[36] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[37] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 215–223.

[38] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, no. 1, pp. 1409–1416.

[39] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*, 2019.

[40] Marc Claesen, Jaak Simm, Dusan Popovic, Yves Moreau, and Bart De Moor. Easy hyperparameter search using optunity. *arXiv preprint arXiv:1412.1114*, 2014.

[41] Marc-André Zöller and Marco F Huber. Benchmark and survey of automated machine learning frameworks. *arXiv preprint arXiv:1904.12054*, 2019.

[42] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[43] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, 2013, no. 1, vol. 30, p. 3.

[44] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[45] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] T Tieleman and G E Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[48] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[49] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[50] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Survey of anomaly detection. *ACM Computing Survey*, 41(3):1–72, 2009.

[51] Ilya Kovalenko, Miguel Saez, Kira Barton, and Dawn Tilbury. SMART: A system-level manufacturing and automation research testbed. *Smart and Sustainable Manufacturing Systems*, 1(1), 2017.

[52] Sharon Sun. CNC Mill Tool Wear, https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill, 2018.

[53] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[54] Francois Chollet et al. Keras, https://github.com/ fchollet/keras, 2015.

Chapter 4

Distance-Based Anomaly Detection for Industrial Surfaces using Triplet Networks

4.1 Introduction

Surface quality inspection is an essential part of the production process in all manufacturing industries around the globe. Scratched, bent, and other imperfect products may result in costly returns, resulting in financial and operational issues [1]. Quality control tasks rely largely on human experts who are trained on identifying anomalies, to ensure defective products are filtered out from the non-defective products that are ready to be used or shipped to consumers. However, as the production rates become faster and products become more complex, it becomes infeasible for humans to keep up with the throughput demand while trying to realize a near-perfect quality inspection. A 1% productivity improvement across the industry can result in $500 million in annual savings [2]. Furthermore, predicting anomalies on time can decrease

over scheduled repairs by up to 12%, maintenance costs by up to 30%, and the number of breakdowns by up to 70% [2]. As a result, machine-based visual inspections have been utilized in recent years to identify defective products, and in particular, Convolutional Neural Networks (CNNs) [3, 4, 5].

A CNN is a class of deep neural networks that uses convolution in place of general matrix multiplication in at least one of their layers, and reduces the number of parameters very efficiently without losing out on the quality of models, making it a prime choice for analyzing visual imagery. A basic requirement for training CNNs efficiently on a classification objective is the accessibility of an adequately large amount of training data for each class. However, with well-optimized processes, there is often an abundance of non-defective samples and a relatively small amount of defective samples. To address this data imbalance challenge, the training objective can be shifted from defect classification to anomaly detection, eliminating the need for any defective samples for training [6]. An additional advantage of this approach is potentially detecting anomalies in novel classes that are not part of the training data set, providing a general solution to the surface quality inspection task.

Some previous approaches in the literature use defective samples for training, or involve training the CNN features to assign a lower feature-space distance to similar parts and a higher feature-space distance to dissimilar parts based on solving a classification task, not solving the anomaly detection task described in this work. Other approaches involve detecting anomalies in certain regions of an image, not capturing potential defects across the entire image.

In this chapter, efficient CNNs based on residual networks [7] are trained to explicitly learn a similarity metric for defective and non-defective patches of surface textures through the use of a triplet network model [8]. The defective samples used in the triplet network are artificially augmented exclusively from non-defective samples via random erasing [9] techniques. After training, a simple prototype pixel-wise subtraction method is performed between the evaluated surface and the prototype measurement for that particular texture surface in the feature space

learned by the CNN, producing an array of distance values (see Figure 4.1). A Support Vector Machine (SVM) with a simple linear kernel is then applied against the mean and maximum distance values collected, with the intention to maximize the separation between the defective and non-defective evaluated samples. Anomaly detection for known classes that are part of the training data set, as well as novel classes that are not part of the training data, are explored for the industrial surfaces. In addition to the surfaces, the different defect types present in the testing batch are explored individually to gain a deeper insight into the anomalies perceptibility.

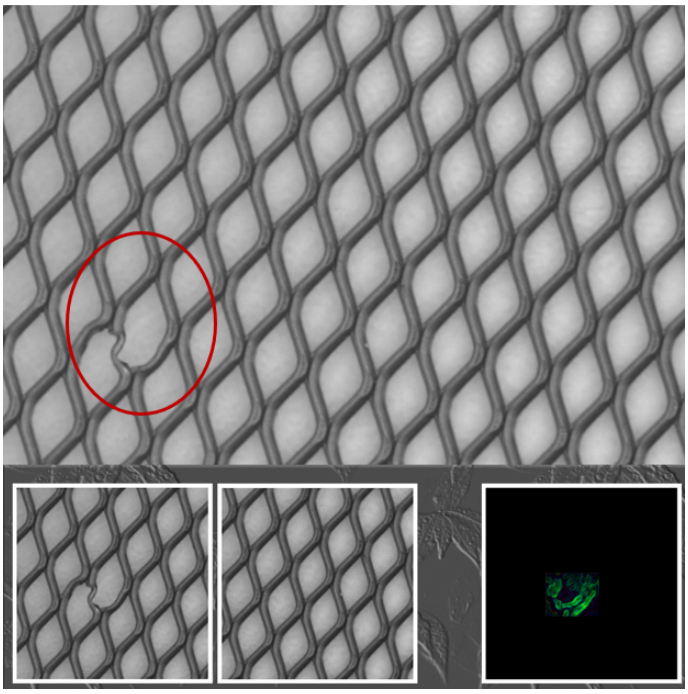

Figure 4.1: Defective surface example. The three images below show the defective patch, the prototype patch and the resulting pixel-wise anomaly score as learnt by the CNN, respectively.

The remainder of this chapter is structured as follows. Section 4.2 presents the motivations behind the use of machine learning and the related work in the field of industrial surface anomaly detection. Section 4.3 contains background information about the data set used. Section 4.4 details the methodology and implementation. Section 4.5 discusses the obtained results and performance evaluation. Finally, Section 4.6 concludes the chapter and discusses opportunities for future work.

4.2 Motivation and Related Work

The following outlines some of the work being done in the field related to the use of machine learning in anomaly detection of industrial surface textures.

Haselman, *et al.* [10] propose a one-class unsupervised learning on fault-free samples by training a deep CNN to complete images whose center regions are cut out. Results demonstrate the approach's suitability for detecting visible anomalies, but fails to accurately detect weakly contrasted anomalies and focuses on the center regions of an image only. Chai, *et al.* [11] propose an approach that utilizes sparse representation to identify anomalies obtained during image reconstruction. Since sparse representation is a computationally expensive approach, lower image resolutions are used, which in turn lowers the accuracy of the anomaly detection task. Natarajan, *et al.* [12] propose the use of transfer learning and a majority voting mechanism for image representations and fuse extracted features. This approach still requires defective samples during training to achieve the anomaly detection objective. Napoletano, *et al.* [13] propose a region-based method for detecting and localizing anomalies via evaluating the degree of abnormality of each subregion of an image compared to an anomaly-free image. The CNN features are trained to specifically assign a low distance to similar parts and a high distance to dissimilar parts based on a pre-built dictionary of normal images, followed by a pre-set anomaly thresholding mechanism rather than a model learnt thresholding mechanism.

The previous work of Staar, *et al.* [14] tackles the shortcomings of the work mentioned above, by using Gaussian noise to synthesize defective samples from non-defective images and utilizing a residual-based [7] triplet network [8]. The network directly learns a similarity metric for the distance-based surface anomaly detection objective, where same-class samples have lower feature space distances than out-of-class samples. Defective images are separated from non-defective images during testing via varying threshold values against the maximum distance value collected.

The work presented in this chapter further extends Staar, *et al.* [14] methodology and results, while training and testing on a real-world data set with a focus on industrial surfaces rather than an artificially generated general texture data set.

The main contributions of this chapter include:

- Artificially synthesize defective images exclusively from non-defective samples using lightweight random erasing data augmentation that do not require any extra memory consumption or parameter learning.
- Implement a deep, residual-based CNN architecture with very small convolution kernels. This allows the network to learn more interesting features while having a small number of trainable parameters.
- Maximize the margin of separation between the defective and non-defective images during evaluation by utilizing a hard-margin SVM against both the mean and maximum feature space distance values collected.
- Evaluate the approach on a real-world data set with a focus on industrial surfaces, where results demonstrate its performance strength in detecting different types of anomalies for known surfaces that are part of the training data and unseen novel surfaces.
- Analyze the individual defect types to gain a deeper insight into the anomalies' perceptibility when using the proposed approach.

4.3 Background

The data set used in this chapter is the MVTec Anomaly Detection Dataset (MVTec AD) [15]. It is a modern, comprehensive, real-world data set with over 5000 images, divided into 15 different industrial object and texture category classes. Each class consists of non-defective

data and a range of different types of defective data, such as bent, broken, and contaminated surfaces. Image resolutions lie between 700x700 and 1024x1024 pixels. In order to evaluate anomaly detection for both known and novel classes, 11 classes are used in training and testing, whereas the remaining 4 classes are only used in testing. The classes are pre-selected randomly. Of the classes to be trained, half of their non-defective images are used for training, and the other half are available for testing. Figure 4.2 shows a sample of defective images from the data set and marks the defective positions in red.

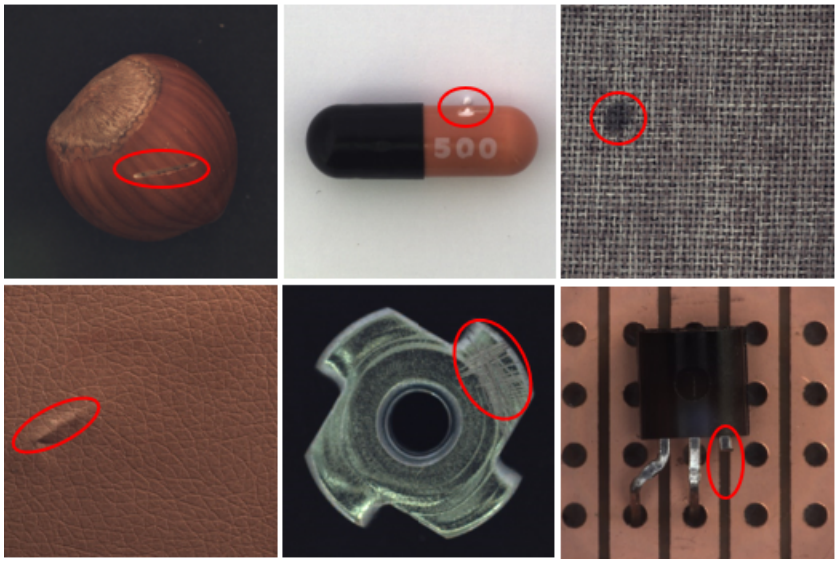

Figure 4.2: Defective Samples from MVTec AD [15] with Defects Marked in Red

4.4 Methodology

The following section details the model's design, implementation and evaluation metrics.

4.4.1 Model

The triplet network model [8] is utilized in this chapter for training . This architecture allows an input of three different images to be fed into the same CNN, where the weights are shared.

The first image is called the anchor (a) image, the second image is called the positive (p) image, and the third image is called the negative (n) image. The a and p images belong to the same class, and the n image belong to a different class. Features of each image are computed, before calculating the Euclidean distance ($d1$) between the resulted features of a and p and the Euclidean distance ($d2$) between the resulted features of a and n. This allows the network to directly learn a similarity metric via deep metric learning, which is beneficial as the number of object classes is not specified and could be boundless. The Euclidean distance is used as initial experiments show better results when it is employed compared to the Manhattan and the Minkowski distance metrics. Equations 4.1 and 4.2 show the Euclidean distance measurements for $d1$ and $d2$ respectively.

$$d1 = \sqrt{\sum_{i=1}^{n}(a_i - p_i)^2} \tag{4.1}$$

$$d2 = \sqrt{\sum_{i=1}^{n}(a_i - n_i)^2} \tag{4.2}$$

Afterwards, the resulting $d1$ and $d2$ scores are inserted into a row vector that is then placed into the softmax function, yielding an output vector who's sum is equal to one. The output vector is trained with the target vector [1,0] for each triplet of (a, p, n), ensuring the network assigns a higher similarity score between images a and p and a lower similarity score between images a and n. Figure 4.3 visualizes the complete training model.

4.4.2 Data Pre-Processing

Before training the network, all training images are converted to grayscale and normalized via Min-Max scaling to rescale their values between 0 and 1. Feature scaling eliminates a large

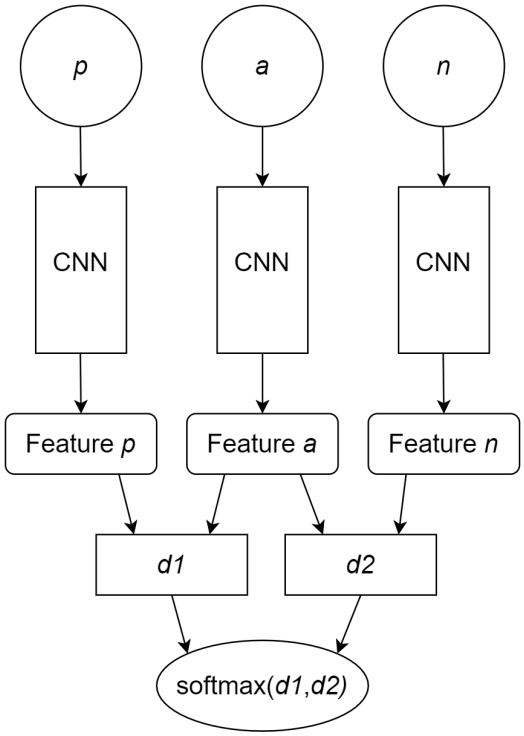

Figure 4.3: The Training Model

scaled feature to be dominating, as the Euclidean distance measure is sensitive to magnitudes, while preserving all relationships in the data [16]. Furthermore, it allows gradient descent to converge much faster.

The CNN implemented in this chapter is only required to learn similarities between patches of 64x64 pixels. The idea is inspired by previous work of Weimer, *et al.* [17] where 32x32 pixels is sufficient for defect classification of images of 512x512 pixel resolution. As a generalization and for research purposes, 64x64 pixels is deemed as sufficient for images of 1024x1024 pixel resolution. In addition, the use of small image patches extracted from an image enables the amount of training data to be increased. To keep the patch size of 64x64 fixed and to capture textural regularities that are expressed on different spatial scales, the input training images are randomly resized to either 1024x1024, 512x512, 256x256 or 128x128 pixel resolutions. 16 random patches are then extracted from the 1024x1024 and 512x512 pixel resolution images, and 8 random patches are extracted from the 256x256 and 128x128 pixel resolution images.

The network is trained without any defective images on an anomaly detection objective, due to the limited number of defective samples to train the network efficiently on a classification objective and to further generalize the solution to the surface inspection task. Defective images used as the n image input into the network are instead artificially synthesized exclusively from non-defective images using random erasing data augmentation [9]. The random erasing technique utilized essentially erases a random part of an image of a random size, enhancing the sensibility to local deviations. Random erasing has the advantage of being a lightweight method, as it does not require any extra memory consumption or parameter learning. Figure 4.4 visualizes an input sample of (a, p, n) to the triplet network.

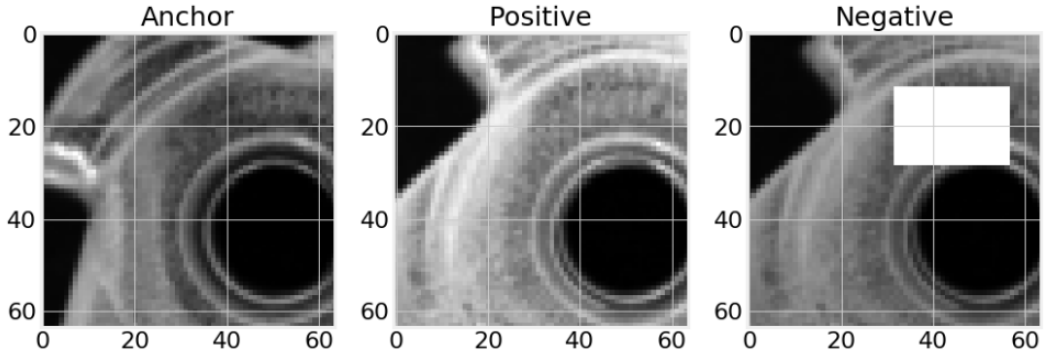

Figure 4.4: Triplet Input Sample of (a, p, n)

4.4.3 Network Architecture

The CNN proposed in this chapter is based on deep residual learning [7] and uses very small convolution kernels, inspired by the work of the widely-known VGG16 network [18]. Residual networks ease the training and optimization of networks, by enabling layers to learn a residual function referenced to the input layer as opposed to learning unreferenced functions. Moreover, the skip connections between layers add the outputs from previous layers to the outputs of stacked layers, allowing networks to become deeper. Deeper networks allow more interesting hierarchical features to be learned and can produce a better generalization [19]. In addition, the VGG16 network demonstrates how a deeper network with very small 3x3 convolution ker-

nels produce more accurate results compared to the previous configurations on the ImageNet Large Scale Visual Recognition Challenge 2014 (ILSCVRC2014) [20] that use larger convolution kernels. Small convolution kernels significantly reduce the number of parameters in the network, where a single 3x3 convolution kernel has a number of parameters ratio of 1:1.4 with a 5x5 convolution kernel, and a 1:2 with a 7x7 convolution kernel.

The first 3 layers in the network presented in this work consist of 2D convolutional layers with Rectified Linear Unit (ReLU) [21] activation functions and kernel sizes of 3x3, followed by a 2D maxpool layer with a window size of 2x2 and strides 2x2. Afterwards, seven residual blocks are stacked next to each other. Each block consists of two consecutive 2D convolutional layers with ReLU activation functions and kernel sizes of 3x3 applied on the input, which produce an output spatial resolution that is reduced by 4x4. Their output is then added to the output of a 2D cropping layer, which applies a spatial dimension crop of size 2x2 to the height and 2x2 to the width of the input. That is required so the output spatial dimension matches that of the 2D convolutional layers, in order for the learned residual to be added and produce the final residual block output. Figure 4.5 visualizes the residual block setup and parameters used in the network, whereas Figure 4.6 visualizes the proposed CNN architecture.

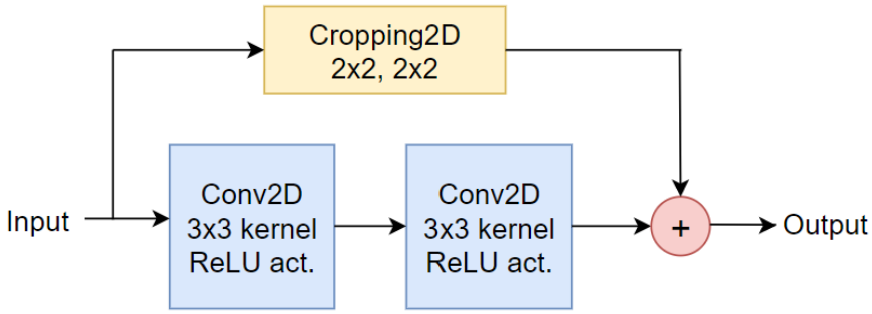

Figure 4.5: The Residual Block Used in the Proposed CNN

Higher-level features are learnt with every layer and residual block. Besides the 3x3 kernel, all convolutional layers across the network use 16 number of filters, valid padding option, and the ReLU activation function. ReLU is utilized as it solves the vanishing gradient problem and is very efficient to compute, as it is linear for positive values and zero for negative values

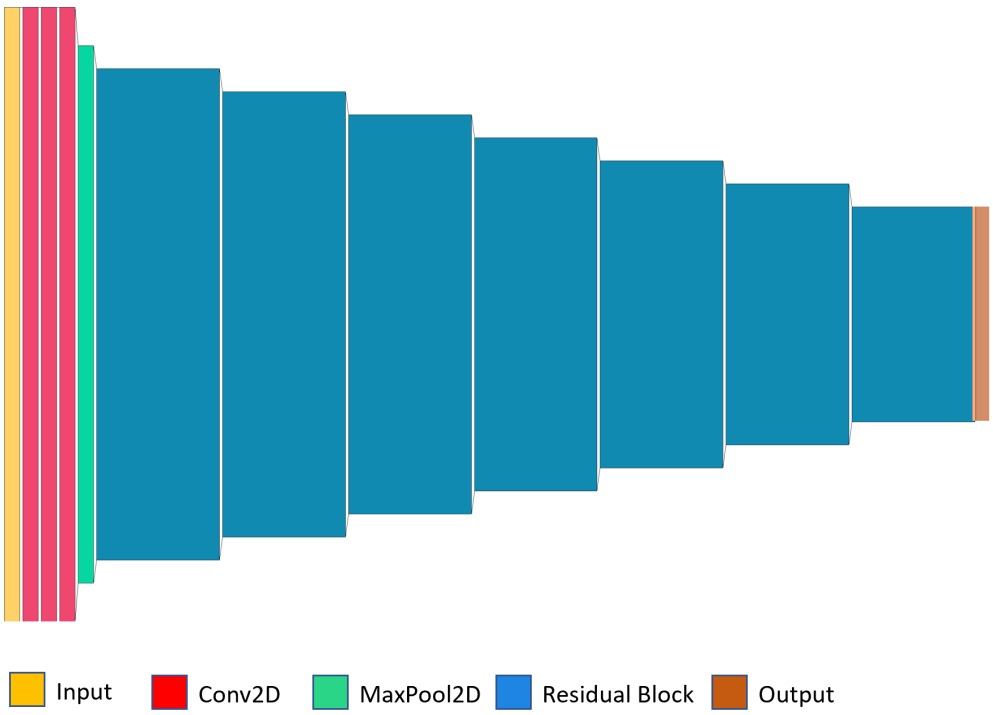

Figure 4.6: The Proposed CNN Architecture

[19]. The entire network is built fully convolutional to allow weights to be shared between the training and testing models, as they process different image pixel resolutions. During training, 64x64 pixels are transformed to tensors with dimensions of 1x1x16, whereas during testing, 1024x1024 pixels are transformed to tensors with dimensions of 481x481x16.

4.4.4 Network Implementation and Hyperparameters

All networks are built and implemented in Python 3.7.4, using the Tensorflow [22] and Keras [23] libraries. All work is run on a machine which comprises of an NVIDIA GeForce GTX 1650 4GB, a 16GB DDR4 2666MHz RAM, and 9th Generation Intel Core i7-9750H Processor.

For training, the Adam optimizer is utilized with a fixed learning rate of 0.0001 and a batch size of 256 samples. The Mean Absolute Error (MAE) loss function is utilized, as initial experiments show much better results with it compared to the Mean Squared Error (MSE) and

the categorical cross entropy loss functions. The network is trained for 40 epochs.

4.4.5 Evaluation Metrics

As for defect detection evaluation, a class prototype is first created by embedding a large amount of non-defective data of that class into the CNN to obtain a set of features. Afterwards, the testing images are embedded into the same CNN and the Euclidean distances between the prototype and the testing images features are calculated. The resulted maximum and mean distances are then subjected to an SVM with a linear kernel, with the intention of maximizing the separation between the defective test images and the non-defective test images. The SVM is pre-trained with the normal and synthesized defective images used in training. The maximum and mean Euclidean distances were selected based on recent work which addressed the setting of working-point decision thresholds for an anomaly detection task [24]. An SVM with a linear kernel allows the robust classification algorithm to be fast and memory efficient, as it is a single inner product with a O(d) prediction complexity, where d is the number of input dimensions. The regularization parameter (C) is kept constant at 1E8, for a harder-margin SVM that imposes a larger penalty for wrongly classified examples.

To evaluate how well the proposed approach discriminates between the defective and non-defective images for each surface and the individual defect types, the Area Under the Curve (AUC) of the Receiver-Operating Characteristic (ROC) is calculated. The higher the AUC, the better the model is at predicting true positives and true negatives. True positives in this work indicate defective images correctly classified as defective and true negatives indicate non-defective images correctly classified as non-defective. Experiments are repeated three times and the average AUC score is recorded. As the work presented in this chapter extends the approach used by Staar, *et al.* [14], their approach is tested against the MVTec AD [15] for a comparison of results.

4.5 Performance Evaluation

Figures 4.7 and 4.8 show the individual and mean AUC scores for Staar, *et al.*'s [14] approach and the proposed approach, respectively. Upon examining the table of results, the proposed approach produces a better AUC score for the majority of the individual class-defect types, and a better mean AUC score for every single class and defect type. The proposed approach scores an average class mean AUC of 0.093 higher, and an average defect type mean AUC of 0.097 higher.

	AUC score															
	Known											Unknown				
Defect Type / Class	Bottle	Cable	Capsule	Carpet	Grid	Hazelnut	Leather	Metal Nut	Pill	Screw	Transistor	Tile	Toothbrush	Wood	Zipper	Mean
Broken	0.634				0.816						0.846				0.845	0.785
Contamination	0.820			0.788	1.000				0.858							0.867
Crack			0.883			1.000			0.770			1.000				0.913
Imprint			0.798			0.948			0.782							0.843
Poke		0.883	0.827				1.000									0.903
Scratch			0.823					0.734	0.725	0.639				0.827		0.750
Cut		0.943		1.000		0.969	0.866				0.719				1.000	0.916
Hole				1.000		1.000								0.899	0.953	0.963
Thread				0.940	0.685					0.817						0.814
Bent		0.819			0.823			0.756			0.822					0.805
Missing		0.849														0.849
Combined		0.932							1.000				0.835	1.000	0.926	0.939
Swap		1.000														1.000
Squeeze			1.000												0.838	0.919
Glue					1.000		0.971					0.861				0.944
Fold							0.884									0.884
Flip								0.671								0.671
Misplaced											0.604					0.604
Liquid												0.896		0.994		0.945
Rough												0.942			0.941	0.942
Mean	0.727	0.904	0.866	0.932	0.865	0.979	0.930	0.720	0.827	0.728	0.748	0.925	0.835	0.930	0.917	0.863 / 0.856

Figure 4.7: Staar, *et al.*'s [14] Anomaly Detection AUC Scores

Furthermore, it can be observed from the class mean scores that the proposed approach achieves some state-of-art results for both known classes and novel classes. In total, only three classes achieved a mean AUC score less than 0.90, four classes achieved a mean AUC score between 0.90 and 0.95, and the remaining eight classes achieved a mean AUC score between 0.95 and 1.0. In particular, three classes achieved a perfect mean AUC score of 1.0, where only one class was known and two were novel, further demonstrating how well the method translates to unknown inputs.

	AUC score															
	Known											Unknown				
Defect Type / Class	Bottle	Cable	Capsule	Carpet	Grid	Hazelnut	Leather	Metal Nut	Pill	Screw	Transistor	Tile	Toothbrush	Wood	Zipper	Mean
Broken	1.000				1.000						1.000				0.945	0.986
Contamination	0.883			0.932	0.947				0.860							0.905
Crack			0.941			0.939			0.819			1.000				0.925
Imprint			0.845			1.000			0.813							0.886
Poke		1.000	0.860				1.000									0.953
Scratch			0.920					0.702	0.883	0.925				1.000		0.886
Cut		1.000		1.000		1.000	0.866				1.000				1.000	0.978
Hole				1.000		1.000								1.000	0.925	0.981
Thread				1.000	0.893					0.854						0.916
Bent		0.935			1.000			0.870			1.000					0.951
Missing		1.000														1.000
Combined		0.959							1.000				1.000	1.000	0.956	0.983
Swap		1.000														1.000
Squeeze			1.000												1.000	1.000
Glue					0.962		0.866					0.911				0.913
Fold							1.000									1.000
Flip								1.000								1.000
Misplaced											1.000					1.000
Liquid												0.856		1.000		0.928
Rough												1.000			1.000	1.000
Mean	0.942	0.982	0.913	0.983	0.960	0.985	0.933	0.857	0.875	0.890	1.000	0.942	1.000	1.000	0.971	0.960 / 0.949

Figure 4.8: The Proposed Approach's Anomaly Detection AUC Scores

Regarding defect types, it can be seen from their mean scores that the proposed approach achieves some state-of-art results. Out of the twenty different defect types, only imprint and scratch defect types achieved a mean AUC score less than 0.90. Contamination, crack, and thread defect types achieved a mean AUC score between 0.90 and 0.95, and the remaining fifteen defect types achieved a mean AUC score between 0.95 and 1.0.

Figure 4.9 shows two scatter plot examples with perfect linear SVM separation between the defective and non-defective samples. The plot at the top shows a separation mainly via features max distance with the prototype for the hazelnut class with imprint defects, and the plot at the bottom shows a separation mainly via features mean distance with the prototype for the transistor class with misplaced defects.

4.6 Conclusion

This chapter demonstrated how deep metric learning can be effectively used for surface anomaly detection of different defect types without the use of any defective samples. It further showed

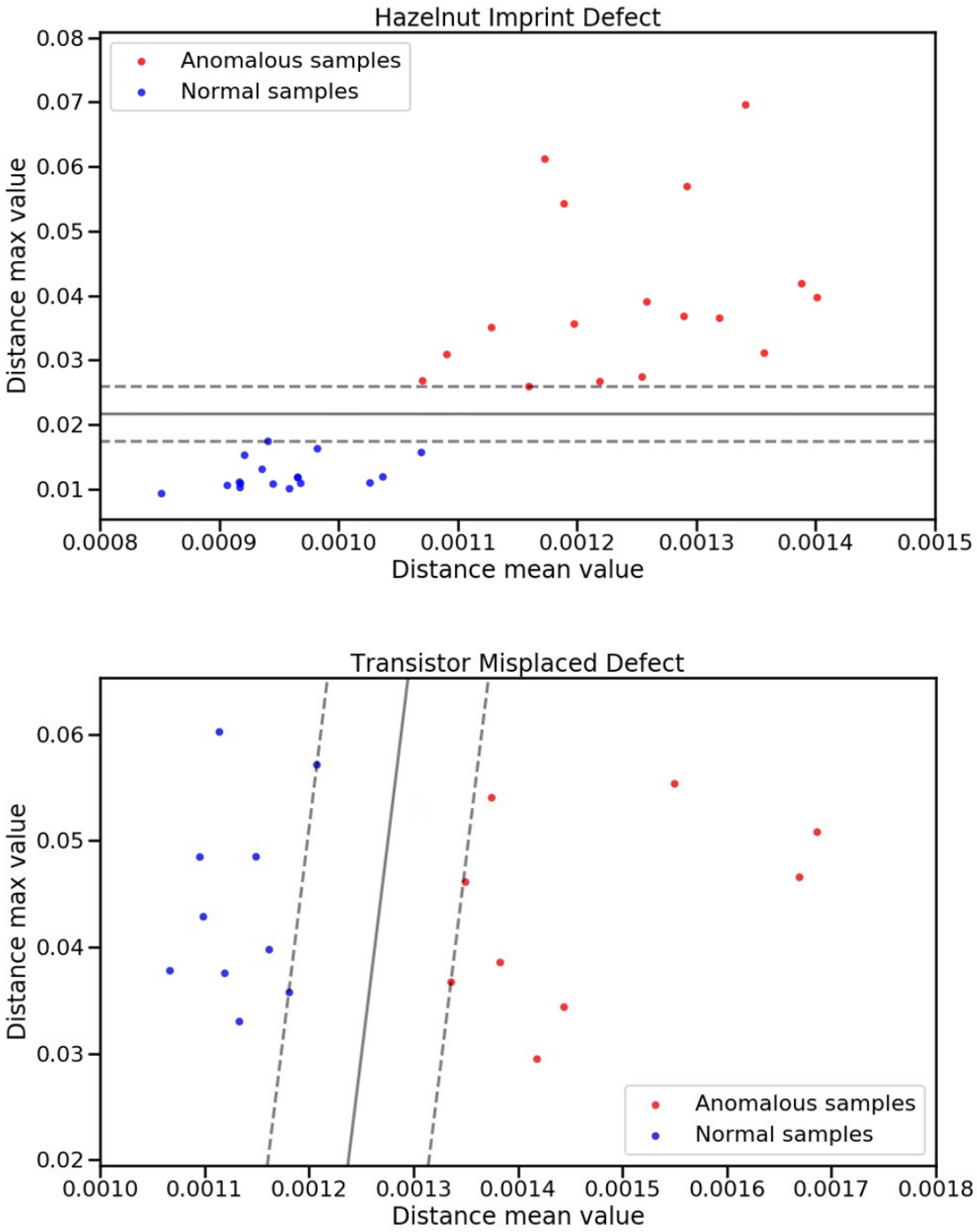

Figure 4.9: Linear SVM Separation Between Defective and Non-Defective Samples

how this approach translated well to novel classes that were not part of the training data set, as two of the four novel classes achieved a perfect AUC score of discrimination between the defective and non-defective images.

As for the methodology, a triplet network model was utilized, which included a deep residual-based CNN with very small convolution kernels. A modern, comprehensive, real-world data set with a focus on industrial surfaces was used for training and testing, where training defective samples were synthesized exclusively from the non-defective samples via random erasing. Image patches of 64x64 were utilized for training instead of using the entire image, allowing training to be more efficient.

Results were promising, but they could still be improved for certain classes and defect types. A potential limitation could be for object classes, where an object comprises a small area of the entire image and most of the extracted 64x64 patches are of the background rather than the object itself. That is backed up by the fact that pill and screw object classes were two of the three classes that achieved a mean AUC score of less than 0.90, where they had the smallest object to whole image ratio in the entire data set. Furthermore, it can be seen that some classes had more defective images than others, and some defect types were present in more classes than others, resulting in an unbalanced comparison of results between the different classes and defect types.

Future work include exploring the addition of training data from other texture and object data sets, as that can allow the network to learn more powerful and complex features. In addition, more testing data could be obtained to gain a deeper understanding and more accurate defect types evaluations. The results could be further enhanced through the use of batch normalization in the CNN, which reduces internal covariate shifts and can accelerate deep network training. The use of maxpooling layers and convolutional layers with same padding could be explored instead of the use of convolutional layers with valid padding. Moreover, the proposed methodology could be further improved in order to diagnose the detected defects and gain

more thorough insights into the production process behaviors. Other potential enhancements include exploring additional loss functions and distance metrics, larger patch sizes, smarter extraction of object patches for surface images with a small object area coverage, different data augmentation techniques, and further tuning the network hyperparameters.

References

[1] Ivan Russo, Ilenia Confente, David M Gligor, and Nicola Cobelli. The combined effect of product returns experience and switching costs on B2B customer re-purchase intent. *Journal of Business & Industrial Marketing*, 32(5):664–676, 2017.

[2] Progress. Anomaly Detection & Prediction Decoded: 6 Industries, Copious Challenges, Extraordinary Impact, https://www.progress.com/docs/default-source/datarpm /progress_datarpm_cadp_ebook_anomaly_detection_in_6_industries.pdf?sfvrsn=82a183de _2, 2017.

[3] Lingteng Qiu, Xiaojun Wu, and Zhiyang Yu. A high-efficiency fully convolutional networks for pixel-wise surface defect detection. *IEEE Access*, 7:15884–15893, 2019.

[4] Luke Scime and Jack Beuth. A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process. *Additive Manufacturing*, 24:273–286, 2018.

[5] Domen Racki, Dejan Tomazevic, and Danijel Skocaj. A compact convolutional neural network for textured surface anomaly detection. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1331–1339. IEEE, 2018.

[6] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[9] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[10] Matthias Haselmann, Dieter P Gruber, and Paul Tabatabai. Anomaly detection using deep learning based image completion. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1237–1242. IEEE, 2018.

[11] Woon Huei Chai, Shen-Shyang Ho, and Chi-Keong Goh. Exploiting sparsity for image-based object surface anomaly detection. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1986–1990. IEEE, 2016.

[12] Vidhya Natarajan, Tzu-Yi Hung, Sriram Vaikundam, and Liang-Tien Chia. Convolutional networks for voting-based anomaly classification in metal surface inspection. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 986–991. IEEE, 2017.

[13] Paolo Napoletano, Flavio Piccoli, and Raimondo Schettini. Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1):209, 2018.

[14] Benjamin Staar, Michael Lütjen, and Michael Freitag. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 79:484–489, 2019.

[15] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9592–9600, 2019.

[16] T Jayalakshmi and A Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):1793–8201, 2011.

[17] Daniel Weimer, Bernd Scholz-Reiter, and Moshe Shpitalni. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals*, 65(1):417–420, 2016.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[21] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[22] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[23] Francois Chollet et al. Keras, https://github.com/ fchollet/keras, 2015.

[24] Sulaiman Aburakhia, Tareq Tayeh, Ryan Myers, and Abdallah Shami. A transfer learning framework for anomaly detection using model of normality. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0055–0061.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

As a way of dealing with the increased system complexities in smart manufacturing, various automated and intelligent anomaly detection methods need to be employed to mitigate any potential system failures. The work presented in this thesis first investigated the domain of anomaly detection in smart manufacturing, before describing a novel, two-part DL-based anomaly detection system for robotic processes in smart manufacturing, with an application focus on robotic surface finishing.

The first part of the presented system proposed the ACLAE-DT framework for anomaly detection and diagnosis in multivariate time series of robotic surface finishing components. Results demonstrated the effectiveness of ACLAE-DT, as it outperformed a classical approach, an ML approach, and three state-of-the-art DL approaches in detecting anomalous windows, while requiring less time to train than the latter approaches. Results further illustrated how ACLAE-DT was able to effectively diagnose the anomalies and locate the contributing features towards the anomalous windows.

Moreover, the second part of the presented system proposed a deep residual CNN-based triplet model for anomaly detection in the produced robotic surface finishes, where defective training samples were synthesized exclusively from non-defective samples via random erasing data augmentation. Results demonstrated how deep metric learning can be effectively used for surface anomaly detection of different defect types without the use of any real-life defective samples in training. Results further illustrated how this approach translated well to novel classes that were not part of the training data set, as two of the four novel classes during evaluation achieved a perfect AUC score of discrimination between the defective and non-defective images.

5.2 Future Work

As a future extension to this work, the following could be addressed. Firstly, the proposed anomaly detection system can be utilized to target other robotic processes in smart manufacturing, such as robotic assembling and robotic molding. Moreover, additional real-life manufacturing data sets can be utilized in order for the proposed system to learn more interesting features and produce better anomaly detection and diagnosis results. Employing supplemental data and data sets further enables the robustness and generalization of the proposed system to be evaluated, as well as benchmark the system's performance with conventional anomaly detection algorithms and methods. Finally, the frameworks utilized in the proposed system can be further optimized by exploring different pre-processing techniques, deep learning architectures, and model hyperparameter optimization methods.

Curriculum Vitae

Publications:

T. Tayeh, S. Aburakhia, R. Myers and A. Shami, "Distance-Based Anomaly Detection for Industrial Surfaces Using Triplet Networks," *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, Nov 2020, pp. 0372-0377.

S. Aburakhia, **T. Tayeh**, R. Myers and A. Shami, "A Transfer Learning Framework for Anomaly Detection Using Model of Normality," *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, Nov 2020, pp. 0055-0061.

B. Jones, S. A. Rohani, N. Ong, **T. Tayeh**, A. Chalabi, S. K Agrawal, and H. Ladak, "A virtual-reality training simulator for cochlear implant surgery", in *Simulation & Gaming*, vol. 50, no. 2, pp. 243-258, May 2019.

Name:	Tareq Tayeh
Post-Secondary Education and Degrees:	Western University London, Ontario, Canada 2019 - 2021 M.E.Sc
	Western University London, Ontario, Canada 2013 - 2018 B.E.Sc
Honours and Awards:	Ontario Graduate Scholarship 2020-2021
	General Motors of Canada Company Graduate Scholarship 2020-2021
	Best Paper Award at the 11th Annual IEEE IEMCON Conference 2020
	Vector Scholarship in Artificial Intelligence 2019-2020
	B.E.Sc Graduation with Distinction 2018
Related Work Experience:	Research Assistant Western University 2019 - 2021
	Teaching Assistant Western University 2019 - 2021
	Lead DevOps Engineer & Software Developer IBM 2018 - 2019
	Software Developer Intern IBM 2016 - 2017