UNIVERSITY OF CALIFORNIA,
IRVINE

# Data-Driven Modeling of Cyber-Physical Systems using Side-Channel Analysis

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by

Sujit Rokka Chhetri

Dissertation Committee:
Associate Professor Mohammad Al Faruque, Chair
Professor Fadi Kurdahi
Professor Pramod Khargonekar

2019

# DEDICATION

To my parents and my brother
for their remarkable
belief, support, and love.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I would like to thank my committee chair, Professor Mohammad Abdullah Al Faruque, for constantly guiding me through the research and providing me his valuable insights. Without his excellent supervision and persistent mentoring, I would not have been able to complete this thesis.

I would also like to thank my Ph.D. committee: Professor Fadi Kurdahi and Professor Pramod Khargonekar for finding time from their hectic schedule to provide supervision, support and insightful comments for the thesis. My sincere thanks go to Dr. Arquimedes Canedo for providing me an opportunity to join his intern team and widening the scope of my research.

I thank Dr. Haeseung Lee, Dr. Jiang Wan, and Dr. Korosh Vatanparvar for engaging and inspiring me to get new ideas for the research. I also thank my colleagues Anthony B Lopez, Sina Faezi, Nafiul Rashid, Shih-Yuan Yu and Anomadarshi Barua Shuvro for collaborating in various research endeavors. I also thank Palash Goyal for the insightful discussions we had during our internship together. I thank my fellow lab-mates for engaging me in stimulating discussions.

Last but not the least, I would like to thank my family and my friends, specially Sayam Moktan for the much-needed love and support, which helped me to carry on through my Ph.D.

# ABSTRACT OF THE DISSERTATION

Data-Driven Modeling of Cyber-Physical Systems using Side-Channel Analysis

By

Sujit Rokka Chhetri

Doctor of Philosophy in Computer Engineering

University of California, Irvine, 2019

Associate Professor Mohammad Al Faruque, Chair

Cyber-Physical System consists of the integration of computational components in the cyber-domain with the physical-domain processes. In cyber-domain, embedded computers and networks monitor and control the physical processes, and in the physical-domain the sensors and actuators aid in interacting with the physical world. This interaction between the cyber and physical domain brings unique modeling challenges one of which includes the integration of discrete and sequential models in cyber-domain with the continuous and parallel physical domain processes. However, the same cyber-physical interaction also opens new opportunities for modeling. For example, the information flow in the cyber-domain manifests physically in the form of energy flows in the physical domain. Some of these energy flows unintentionally provide information about the cyber-domain through the side-channels.

In this thesis, the first part consists of an extensive analysis of the side-channels (such as acoustic, magnetic, thermal, power and vibration) of the cyber-physical system is performed. Based on this analysis data-driven models are estimated. These models are then used to perform security vulnerability analysis (for confidentiality and integrity), whereby, new attack models are explored. Furthermore, the data-driven models are also utilized to create a defensive mechanism to minimize the information leakage from the system and to detect attacks to the integrity of the system. The cyber-physical manufacturing systems are taken

as use cases to demonstrate the applicability of the modeling approaches.

In the second part, side-channel analysis is performed to aid in modeling digital twins of the cyber-physical systems. Specifically, a novel methodology to utilize low-end sensors to analyze the side-channels and build the digital twins is proposed. These digital twins are used to capture the interaction between the cyber-domain and the physical domain of the manufacturing systems, and aid in process quality inference and fault localization. Using side-channels these digital twins are kept up-to-date, which is one of the fundamental requirements for building digital twins.

Finally, challenges relating to performing data-driven modeling using non-Euclidean data in cyber-physical system is addressed in third part of the thesis. Moreover, a novel structural graph convolutional neural network and a dynamic graph embedding algorithm is presented to handle non-Euclidean data.

# Chapter 1

# Introduction and Background

## 1.1 Cyber-Physical System

Cyber-Physical Systems (CPS) consists of integration of computational components in the cyber-domain with the physical domain processes [1, 2]. The physical-domain processes consist of actuators which are coordinated and controlled by the computational components via a communication network, where the computational processes are usually affected by the feedback provided by the sensors in the physical domain. In the cyber-domain, the computational and communication cores monitor and manipulate the discrete signals, whereas, in the physical-domain, energy flows, which are mostly continuous domain signals, govern the physical dynamics of the system. Due to the juxtaposition of cross-layer components (*physical, network, control, system, operation*, etc.) and cross-domain components, CPS provide various technology solution to multiple fields (*automotive, manufacturing, health care*, etc.) [3].

CPS consists of complex interaction between heterogeneous (different types of computation and communication platforms) and hybrid (discrete and continuous) components. Although

integration of cyber and physical processes is covered by embedded systems, the networking and multi-domain interaction among embedded systems, makes CPS system different and hence, challenging to design. Traditionally, there has been extensive research carried out for design and automation tools targeted at hardware and software co-design for embedded system design, and for multi-physics modeling and simulation of the control systems. However, due to the tight integration of cyber and physical domain unique challenges mostly related to integration are still being tackled. To this end, this thesis also tackles the challenges introduces by the cross-domain aspect of the CPS.

One particular example of CPS considered throughout the thesis is manufacturing systems. CPS are expected to be an integral aspect of manufacturing systems for monitoring and controlling the machines. Manufacturing system consists of multi-domain embedded systems which interact among each other through a network. They consists of physical actuators and sensors to interact with the physical environment.

## 1.2 Data-Driven Modeling

Traditionally, CPS modeling consists of various techniques depending on type of modeling (such as component-oriented, multi-agent-based, actor-oriented, event-based, or structural or behavioral based), based on the characteristics (such as domain-specific or multi-domain, timed or un-timed, discrete, continuous or hybrid) or based on the modeling requirements (such as specification, analysis, verification and validation [4]. Modeling consists of formal methodologies for designing and engineering CPS. Most of the existing modeling approaches are first-principle based, as they are derived after complex analysis by the domain experts [5]. There are various tools and resources for first-principle based CPS modeling. There are CPS modeling languages such as Architecture Analysis and Design Language(AADL) [6], Unified Modeling Language (UML) [7], Planning Domain Definition Language(PDDL)

[8], CyPhyML [9], ESMol [10], etc. Moreover, there are various simulation tools such as Modelica, Simulink, LabView, Ptolemy [11], etc.

The first principle based approach provide us with the CPS model which have formal and provable deterministic properties, and capable of enabling simulation and analysis for detecting design defects [12]. However, it has some limitations which arise due to the fact that they are poor at modeling non-deterministic or stochastic properties of CPS, which are introduced while physically realizing the model. These non-deterministic behavior is introduced due to environmental variation, physical noise of the underlying platform, part failures, network delays, uncontrollable scheduling, etc [13].

These limitation can be alleviated using data-driven modeling approaches. Data-driven modeling consists of utilizing the data about the specific system to estimate or infer the relationship between the system variables (for example input and output variables) without requiring the domain expertise or detailed knowledge of the behavior of the system [14]. Although data-driven modeling required large amount of data, due to proliferation of sensors incorporated in CPS, high volume, velocity and variety of data can be acquired from operational CPS to aid in modeling the next generation of CPS. The main advantage of data-driven modeling approach is that it is able to utilize the side-channels (explained in next section) to infer various cross domain system variables. Moreover, data-driven modeling methodology enjoy the advancement in the field of artificial intelligence, machine learning, data mining, etc., to build models capable of accurately estimating relationship between system variables. To this end, in this thesis we explore various data-driven modeling methodology to model the interaction between the cyber and the physical domain variables.

## 1.3 Side-Channel Analysis

Side-channel analysis consists of analyzing the analog emission from the physical domain of CPS to infer about the various system state (especially in the cyber-domain). Traditionally, side-channel were used to infer the cryptographic keys by utilizing the vulnerability in the physical implementation of the system which unintentionally leak the information [15] rather than using brute force or attacking theoretical weakness of the algorithms. In CPS, computational and communication cores, governed by cyber-processes, interact with physical-domain sensors and actuators. This interaction is unique due to the fact that the information flow in the cyber-domain may manifests physically in the form of energy flows in the physical-domain. These energy flows may be observed in the form of various analog emissions such as *vibration, acoustic, magnetic, power*, etc. These analog emissions may behave as side-channels and allow us observe various cyber-domain state variables from the physical domain.

In this thesis, the data-driven modeling utilizes the side-channels to infer about the relationship between cross-domain variables. The inference is performed to demonstrate vulnerability of the system to confidentiality breach, to build defense mechanism, and to build digital twin of the CPS.

## 1.4 Digital Twins

A *digital twin* is the virtual representation of a physical system (its *physical twin*) [16]. The concept of the *digital twin* was first used by NASA to describe a digital replica of physical systems in space maintained for diagnosis and prognosis. Digital twin consists of large historical context and performance data and utilizes the direct (through inbuilt sensors) and indirect (through latent variable analysis) sensing to provide the near real-time representation

of the physical system. Moreover, it consists of various models (for simulation, monitoring, control, optimization, etc.,) in a hierarchical manner (consisting of a representation of the system, process, component, etc.,) which can provide the blueprint of the whole system [17]. The digital twin allows the user to monitor, simulate, optimize, and control the entire physical system in the virtual domain . Organizations like Siemens [18], General Electric [19], NASA [20], and the Air Force [21] are currently building *digital twins* of gas turbines, wind turbines, engines, and airplanes that allow them to manage the assets, optimize the system and fleets, and to monitor the system health and provide prognostics.

Digital twins can be modeled to accurately estimate the interaction between the cyber and the physical components by building data-driven models to observe the input (cyber-domain variables) and the output (side-channels emissions) variables of the system. Since the side-channel rely on the physical implementation of the CPS, they are able to capture various stochastic behavior in CPS that arise due to the environmental variation, human interaction, process variation, etc. In this thesis, digital twin models are estimated using data-driven modeling methodology by observing the side-channels.

## 1.5 Euclidean Versus Non-Euclidean Data

Data-driven modeling approaches utilize various algorithms from the field of machine learning and artificial intelligence. Over recent years, due to availability of large amount of data and decrease in price of computation, complex algorithms such as deep neural network have been able to perform data-driven modeling with high accuracy in task such as classification and regression. Most of the algorithms developed so far are however for euclidean space. Euclidean data are defined as set of points in euclidean space which satisfy certain relationship or are expressible in terms of distance and angle. They can be operated using various well known operations such as euclidean norm, dot product, etc. Some of the example of

these data are time-series data, image, etc.

However, in CPS, there are large amount of data which are non-euclidean in nature. These data consists of no common system of coordinates, no vector space structure, no description of property like shift in-variance in convolutional neural network for images to name few. Examples of non-euclidean data include social network data, graphs, manifolds etc. In CPS, there are large amount of non-euclidean data, such as call-graphs, finite-state-machines, etc. The engineering design data mostly varies from domain to domain. In electronic design, it consists of high-level design descriptions, register transfer level descriptions in Verilog or VHDL, schematics, etc. In mechanical design, it consists of data regarding structural designs, modeling, and analysis of components, etc. Moreover, there is a wealth of data generated throughout the supply chain of engineering including Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) tools. In order to perform meaningful learning from these data, we need to utilize non-euclidean or graph learning algorithms that are able to extract, categorize, and label these sparse data. Hence, in this theis, to aid data-driven modeling, various algorithms capable of handling non-euclidean data are proposed.

## 1.6 Major Challenges and Thesis Contribution

The major challenges for data-driven modeling of cyber-physical systems are as follows:

- Modeling interaction between cyber and physical system.
- Modeling non-functional requirements of CPS.
- Modeling and updating digital twins of legacy CPS.
- Handling non-Euclidean data.

The major contribution of the thesis to tackle the above mentioned challenges are as follows:

1. Novel data-driven modeling for exploring new attack models.

2. Cross-domain security models for integrity.

3. Internet of Things (IoT)-enabled living digital twin for legacy CPS.

4. Novel algorithms for handling non-Euclidean data.

## 1.7 Summary

The overall summary of the contributions, chapters and the referred publications are presented as follows:

| Contribution | Chapters | Chapter References (in order) | Supporting References |
|---|---|---|---|
| 1 | 2 | [22] | [23, 24, 25, 26, 27, 28, 29, 30] |
| 2 | 3,4,5 | [31, 32, 33] | [34, 35] |
| 3 | 6,7 | [36, 37] | - |
| 4 | 8,9 | [38, 39] | [40, 41] |

The hierarchy of the thesis based on sub-division of the chapters is presented in figure 1.1.

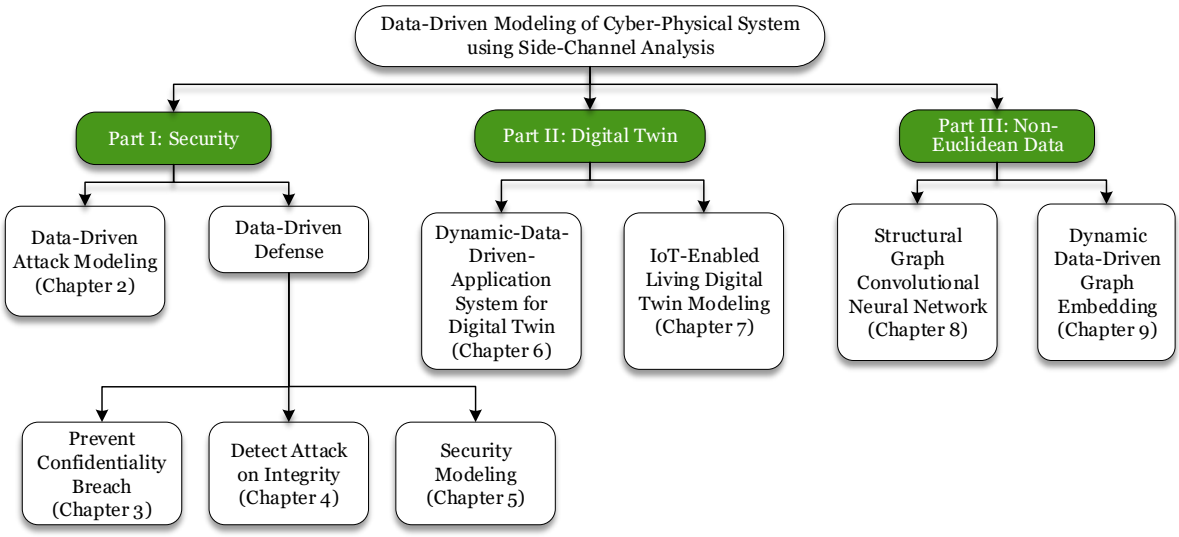



Figure 1.1: Structure of the thesis

# Chapter 2

# Part I: Data-Driven Attack Modeling

## 2.1 Introduction

Cyber-physical systems consist of the integration of computation, physical, and networking components. The synergy of these components results in a new form of vulnerabilities, which cannot be addressed by traditional security solutions designed for the individual components. In this chapter, the focus is put on cyber-physical additive manufacturing systems, where 3D objects are created one layer at a time. Fused Deposition Modeling (FDM) is one of the technologies used in additive manufacturing, where plastic or metal filaments, heated slightly above their melting point, are deposited to construct a 3D object. Several sectors, such as medical and aerospace, are increasingly adopting the use of these additive manufacturing systems [42, 43]. In addition, agencies like the U.S. Air Force, Navy, and NASA are also incorporating additive manufacturing into their manufacturing processes [43, 44]. This trend shows that new vulnerabilities, such as cross-domain attacks, can have a large economic impact on manufacturing industries utilizing additive manufacturing. Attackers who target cyber-physical manufacturing systems are often motivated by either industrial es-

pionage of Intellectual Property (IP), alteration of data, or denial of process control [45, 46]. The world economy relies heavily on IP-based industries, which produce and protect their designs through IP rights. IP in cyber-physical manufacturing consists of *the internal and external structure of the object*, *the process parameters*, and *the machine-specific tuning parameters* [47]. To produce a 3D object, design information (which contains IP) is supplied to the manufacturing system in the form of G-code. G-code, a programming language, is primarily used in FDM to control the system components and parameters such as *speed*, *extrusion amount*, etc [48]. If these designs are stolen, they can be manipulated to harm the image of the company, or even worse, can cause the company to lose its IP (as it is stolen before production) [49]. Currently, IP theft mainly occurs through the cyber-domain (e.g., Operation Aurora, GhostNet) [50, 51], but IP information can also be leaked through the physical-domain (side-channels). A common example of this is to use side-channel information (e.g., timing data, acoustics, power dissipation, and electromagnetic emission) from devices performing the cryptographic computation to determine their secret keys [15]. In this chapter, a data-driven modeling approach is adopted to highlight the possibility of physical-to-cyber attacks on CPS, and motivate a general research interest in novel ways to minimize side-channel leakage during design time and manufacturing.

### 2.1.1 Research challenges and contributions:

It is probably not possible to make a system completely secure. This is because many vulnerabilities are not known during design time. Hence, it is necessary to continue to investigate this field to identify novel threats. The contribution of the research presented in this chapter to the security research in cyber-physical additive manufacturing systems are as follows:

- **Source of acoustic emission (Section 2.3)** where the source of acoustic emission

in a FDM technique based 3D printer is analyzed, and various equations to better understand the working principles of the proposed attack model is presented.

- **Acoustic Leakage Analysis (Section 2.4)** where the side-channel leakage model is provided and leakage quantification is performed to understand the relation between the cyber-data and acoustic emission.

- **A data-driven acoustic side-channel attack model to breach confidentiality (Section 2.5)** which describes our novel attack methodology. It consists of an exploration of time and frequency domain features, learning algorithms trained to acquire specific information (axis of movement, speed of the nozzle) about the G-code, context-based post-processing, and algorithms used to reconstruct the G-code by reverse engineering.

## 2.2 Background and related work

A typical digital process chain in cyber-physical additive manufacturing systems is presented in Figure 2.1. Designers start their design of 3D objects with 3D Computer-Aided Design (CAD) modeling tools such as Sketchup [52] and the extended version of Photoshop [53]. Next, the CAD tool generates a standard STereoLithography (STL) file for the manufacturing purpose. Computer Aided Manufacturing (CAM) software is then required to slice the STL file into layer-by-layer description file (e.g., G-code, cube, etc.). Then, the layer description file is sent to the manufacturing system (e.g., 3D printer) for production [48].

In the physical-domain of the cyber-physical additive manufacturing system, components such as a stepper motor, fan, extruder, base plate, etc., carry out operations on the basis of information provided by the cyber-domain (G-code). In carrying out the operation, these physical components leak cyber-domain information (G-code) from the side-channels, such

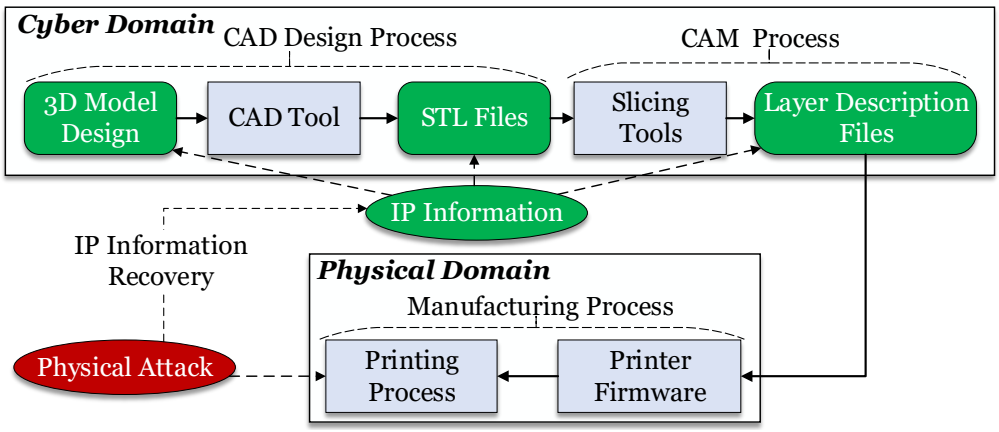



Figure 2.1: Physical attack during printing process of cyber-physical additive manufacturing system

as acoustic and power, which may be used to steal IP by performing a physical-to-cyber attack. The issues regarding the theft of IP and the framework for preventing IP theft have been studied in [47, 54]. The study of attack in the process chain, starting from the 3D object design to its creation, along with a case study of cyber attacks in the STL file, is presented in [55]. However, physical-domain attacks are not well studied by the existing works.

There are several publications utilizing the side-channel information to gather data related to the cyber-domain in other systems. [56] has used the acoustics emanated from the dot matrix printer while printing to recover the text it was sent to print. Authors in [57] have been able to decode the keys pressed in the Enigma machine by analyzing the sound made by the device while pressing the keys. However, these methodologies are not applicable to 3D printers since, unlike printed words on paper, a 3D printer's movement has infinite possibilities. Researchers from MIT have found that even the minor movement of physical devices can leak information about the cyber-domain. In [58], they have successfully retrieved digital audio being played by capturing the vibration of objects near a sound source by a high-speed camera. However, in a 3D printer, there are multiple sources of sound and vibration. Therefore, the task of analyzing sound for G-code reconstruction requires a completely new

approach. Authors in [59] have considered using side-channel for providing security, but they have not demonstrated any methodology for using it to steal the IP. In summary, the literature is focused on retrieving the text being printed (either in a keyboard or dot matrix printer), analyzing acoustic emissions for observing mechanical degradation of the physical components in a manufacturing plant, etc. However, the possibility of using the acoustic emissions for the reconstruction of a 3D object has not been considered. Hence, in this chapter, a data-driven acoustic side-channel attack model to breach the confidentiality of cyber-physical additive manufacturing systems is presented.

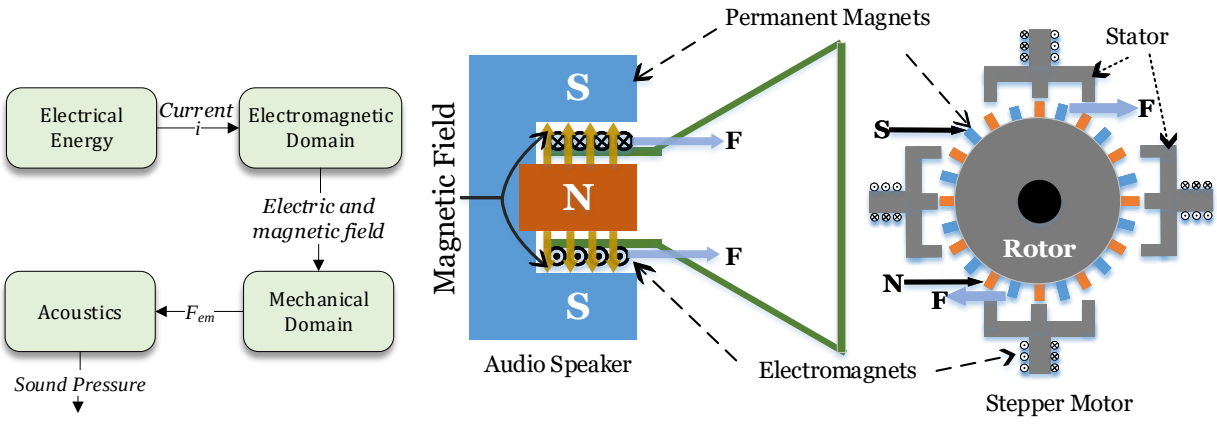
## 2.3 Sources of Acoustic Emission

A 3D printer has various sources of acoustic emissions. A strong attacker may be able to describe the behavior of acoustic emission in the side-channel based on the various G-codes using a single physical model (using a single equation). However, this task is not trivial, as it has to consider all the sources of vibration in the system. Moreover, they may require the system design knowledge to model the mechanical and electrical model of the system, which may not readily be available. Instead, data-driven modeling of the system can easily be created by treating the system as a black box. However, to better understand why these data-driven models work, a preliminary analysis of individual sources of sound in a typical FDM technique based 3D printer is provided in this chapter. Based on this, in Section **??**, leakage analysis is performed to understand why various information about G-code can be inferred from the acoustic emission.

### 2.3.1 System Description

State-of-the-art FDM based additive manufacturing systems consist of four to five stepper motors depending upon their structural design and number of filaments available for extrusion. Due to high torque/size ratio, hybrid two-phase stepper motors have been widely used in these 3D printers. However, these stepper motors are the major source of sound that enables the leakage of cyber-domain information from the side-channel. Hence, in the rest of the sections, the focus is given on describing various mathematical models of the stepper motor and analyzing how they aid in sound production.

The energy conversion steps for a stepper motor is shown in Figure 2.2. Here, the electrical energy (current $i$) is first converted to an electric and magnetic field, which in turn guides the rotors. The electromagnetic field acting upon the various components produces force $F_{em}$, which causes them to vibrate and produce sound with power $P$. However, the electrical energy is controlled by the G-codes, which gives the printer instruction to control the dynamics of the system (such as nozzle speed, axis movement, etc.). Hence, the acoustics emitted by the printer eventually depends on the supplied G-code. Moreover, the stepper motor behaves likes a traditional audio speaker due to its structure (see figure 2.2 (b)).




a) Energy flow in stepper motor

b) Comparison of stepper motor with audio speaker

Figure 2.2: Internal structure of the stepper motor

### 2.3.2 Equation of Motion

To understand the production of mechanical energy, the equation of motion for a hybrid stepper motor can be analyzed as follows [60]:

$$J\frac{d^2\theta}{dt^2} + D\frac{d\theta}{dt} + p\Psi_m i_A sin(p\theta) + p\Psi_m i_B sin(p(\theta - \lambda)) = 0 \tag{2.1}$$

Where $J$ is the moment of inertia of the rotor and the load combined, $J = J_M + J_L$, and $D$ is the damping coefficient based on eddy current, air friction, hysteresis effects, etc. $i_A$ and $i_B$ are the current flowing through the two phases. $\psi_m$ is the maximum stator flux linkage, $p$ is the number of rotor pole pairs, $\lambda$ is the angle between the two stator winding, and $\theta$ is the mechanical rotational angle. Equation 2.1 states that the inertia in the motor depends on the current being supplied to the two phases of the stator. This inertia determines the amount of vibration produced by the motor when it is rotating. Moreover, based on this equation the natural oscillation frequency of the rotor can be derived, which plays a major role in generating unique acoustics for the given stepper motor.

### 2.3.3 Natural Rotor Oscillation Frequency

The radiated sound power is higher when the stepper motor vibrates with the rotor's natural oscillation frequency. Using Equation 2.1, the natural frequency of rotor oscillation can be calculated as follows [61]:

$$\omega_{np}^2 = \frac{2p^2\Psi_m I_o cos(\frac{p\lambda}{2})}{J} \tag{2.2}$$

Where $I_o$ is the stationary current flowing in the two phases $A$ and $B$. When the stepper motor is rotating with the harmonic frequency of the natural frequency such as ..., $\frac{\omega_{np}}{4}$, $\frac{\omega_{np}}{3}$, $\frac{\omega_{np}}{2}$, $2\omega_{np}, 3\omega_{np}$, $4\omega_{np}, ...$ the vibration is more prominent due to resonance. Given the fact,

that $J$ is the inertia of load and the rotor combined, varying load in the stepper motor will change its natural rotor oscillation frequency as well.

### 2.3.4 Stator Natural Frequency

Natural rotor oscillation frequency corresponds to the dynamic response of the stepper motor. However, the stator itself has a natural frequency which depends on various parameters. One of the major parameters is vibration modes. Due to the prominence of the radial force acting on the stator, consideration may be given only to the circumferential radial vibration modes and the corresponding stator natural frequencies. The structure of the stator is complex and many attempts have been made to calculate the natural frequencies of the stator with various considerations, an example being single-ring type stator [62, 63]. Since the external structure connected to the stator also influences its mass and stiffness, the natural frequency of the stator with circumferential vibration mode $m$ and axial vibration mode $n$ of the frame may be calculated as follows [64]:

$$\omega^2_{stator\ np} \approx \frac{K^{(c)}_m + K^{(f)}_{mn}}{M_c + M_f} \tag{2.3}$$

Where $K^{(c)}_m$ is the lumped stiffness of the stator core, $K^{(f)}_{mn}$ is the lumped stiffness of the frame, and $M_c$ and $M_f$ are the mass of the stator and the frame, respectively. Equation 2.3 has been derived by assuming that the lumped stiffness of the core and the frame are in parallel.

### 2.3.5 Source of Vibration

The main sources of vibration in stepper motors are electromagnetic, mechanical, and aerodynamic [65]. These vibrations help in radiating sound from the stepper motor stator surface

and the frame to which the motor is connected. In this chapter, consideration is given to the electromagnetic and mechanical sources as they are the major sources of leakage.

**Electromagnetic Source:** The fundamental source of vibration in hybrid stepper motors is due to the fluctuation of electromagnetic force produced by the winding of the stator. The two types of vibration produced by the electromagnetic force are:

i) *Radial Stator Vibration*: In a hybrid stepper motor, both stator and the rotor are responsible for exciting the magnetic flux density in the air gap between the rotor and the stator. These magnetic fluxes contribute to generating the radial force. If $\sigma_{l,k}$ be the radial force at pole $l$ for $k^{th}$ harmonic then the total radial force acting on the stepper motor may be calculated as follows [66]:

$$\sigma_{total} = \sum_{k=1}^{\infty} \sigma_{l,k} cos(k\omega t + \phi_{l,k}) \tag{2.4}$$

Where $\phi_{l,k}$ is the phase angle of the radial force at pole $l$ for $k$ harmonic, $\omega = 2\pi f$, and $f$ is the frequency determined by the stepping rate of the motor. This radial force acts on the stator and rotor surface and deforms its structure. This produces vibration and eventually sound in the stepper motor. When the radial force excites the harmonics of the natural frequencies of the stator/frame structure and the rotor oscillation, vibration is more prominent due to resonance.

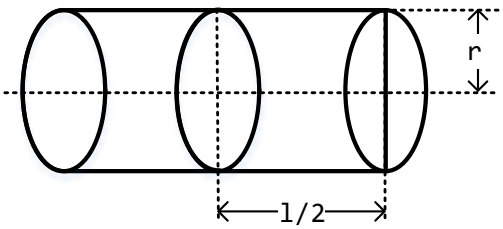



Figure 2.3: Outer mechanical structure of a stepper motor

Let us assume that the structure of the stator of the hybrid-stepper motor is cylindrical (as shown in Figure 2.3). With this, we may express the total sound power radiated by the

electric machine due to the varying radial force acting upon the stator as follows [62, 65]:

$$P = 2\rho c\pi^2 f^2 {A_{rd}}^2 (4\pi rl/2) I_{rel} = 4\rho c\pi^3 f^2 {A_{rd}}^2 rl I_{rel} \tag{2.5}$$

Where $P$ is the radiated sound power ($W$), $\rho$ is the density of the medium ($kg/m^3$), $c$ is the speed of the sound in the medium ($m/s$), $f$ is the excitation frequency of the vibration with multiple harmonics ($Hz$), $A_{rd}$ is the surface vibratory displacement ($m$), $r$ is the radius of the cylindrical stator ($m$), $l$ is the length of the stepper motor ($m$), and $I_{rel}$ is the relative sound intensity. $I_{rel}$ depends on the mode of stator vibration $R$, the radius, and the length-diameter ratio. Hence, stepper motors with different geometry and design in the 3D printer will emit different sound power.

ii) *Torque Ripple*: Even though torque ripple is substantially reduced by using the micro-stepping for driving the stator windings, micro-stepping position ripple is still produced due to non-conformity to the ideal sine/cosine waves required for absolute removal of the torque ripple. However, the vibration produced by the torque ripple is less compared to the radial stator vibration.

**Mechanical Source:** The rotor and load connected to the stepper motor may also produce vibration and sound at various frequencies due to friction, rotor unbalance, shaft misalignment, loose stator lamination, etc. These vibrations produce a loud noise due to resonance.

In summary, the major source of acoustic emission are the stepper motors. However, any actuator (for example DC motors used in cooling fans) that responds to the G-code are also capable of leaking information from the acoustic side-channel. A strong attacker will consider all these facts to make their attack model accurate. In the subsequent sections, a detailed analysis of the information that may be leaked through the acoustic emission is presented.

## 2.4 Acoustic Leakage Analysis

### 2.4.1 Side-Channel Leakage Model

Using an acoustic data acquisition device, an attacker may physically observe analog emissions $[o_1, o_2, o_3, ..., o_i]$, where $o_i$ denotes the $i^{th}$ sample observed in the time domain. This is in fact the measurement of the acoustic power radiated by the stepper motor, as given in Equation 2.5. Let $O$ be a random variable denoting observable analog emissions, then side-channel leakage can be modelled as follows:

$$O = f_d(.) + N \tag{2.6}$$

Where $f_d(.)$ represents a deterministic leakage function, that may be modeled by an attacker. $N$ represents a random variable denoting noise independent from $f_d(.)$ added to the side-channel. Leakage function $f_d(.)$ depends on the G-code instruction $[g_1, g_2, g_3, ..., g_j]$, where $g_j$ is the $j^{th}$ instruction supplied to the 3D printer. The G-code may be denoted by a random variable $G$, then the leakage model may be re-written as follows:

$$O = f_d(G) + N \tag{2.7}$$

Since some of the G-code take longer time to execute than others, for each G-code instruction $g_j$, there will be $k$ samples of analog emissions corresponding to it. The length $k$ depends on the sampling frequency of the audio device used. The fundamental information contained in $g_j$ are speeds in each axis $\{v_{x_j}, v_{y_j}, v_{z_j}\} \in \mathbb{R}_+$, presence or absence of axis movements $\{a_{x_j}, a_{y_j}, a_{z_j}\} \in \{0, 1\}$ with 0 representing absence of movement, positive or negative distance moved in each axis $\{d_{x_j}, d_{y_j}, d_{z_j}\} \in \mathbb{R}$, and extrusion amount $d_{e_j} \in \mathbb{R}$. Hence, the Equation

3.1 may be rewritten as follows:

$$o_i = f_d(v_{x_i}, v_{y_i}, v_{z_i}, a_{x_i}, a_{y_i}, a_{z_i}, d_{x_i}, d_{y_i}, d_{z_i}, d_{e_i}) + n_i \tag{2.8}$$

Where the smaller alphabets represents the value of the random variables $O$, $G$ and $N$ during $i^{th}$ time interval, with random random variable $G$ further partitioned into random variables $\{V_x, V_y, V_z, A_x, A_y, A_z, D_x, D_y, D_z, D_e\}$ representing speed, axis movement, and distance, respectively, with their corresponding values passed to the function $f_d(.)$ in Equation 2.8. Using data-driven approach, various machine learning algorithms such as regression, classifications, etc., may be used to estimate a function, such that for the $i^{th}$ sample of observable emissions the corresponding G-code $\hat{g}_i = \hat{f}_d(o_i, \alpha) + n_i$ can be estimated. Where $\hat{f}_d$ represents the estimated function, and $\alpha$ represents the tuning parameter learned for the function. Due to the presence of multiple parameters, classification and regression machine learning algorithms may be used to estimate multiple functions to estimate individual parameters separately.

**Assumption 1.** *Given the acoustic leakage $O$, the frequency of the radiated sound varies according to the speed of the nozzle in the X and Y axis, respectively.*

The radial force generated in Equation 2.4 in each pole depends on the magnetic flux density, stator tooth width, and rotor cap thickness. The magnetic flux density depends on the current passing through each winding. To increase the angular speed of the stepper motor, the stepping rate is increased. From Equation 2.4, it can be seen that this increases the frequency of the radial force acting on the stepper motor. From Equation 2.5, it can also be seen that the radiated power increases with the excitation frequency of the vibration.

**Assumption 2.** *Given the acoustic leakage $O$, the power frequency spectrum of the radiated sound from the stepper motors X, Y, Z, and the one for the extruder are different.*

The natural rotor oscillation frequency in Equation 2.2 is inversely proportional to the moment of inertia of the load and the motor $(J_L + J_M)$. The load moved by each stepper motors

$X$, $Y$, $Z$, and $E$ are different in state-of-the-art stepper motors. The natural frequency in Equation 2.3 also depends on the mechanical structure of the frame to which the stepper motor is connected. Due to the mechanical structure of the 3D printers, stepper motors are placed in various locations and are connected to different frame structures. Therefore, the natural frequencies of the stepper motors vary according to the load and the frame to which they are attached. This means that the resonance can occur at different frequencies of the vibration for different stepper motors and the frame structure. This causes the power spectrum of the radiated sound to vary according to the source of the sound, i.e., the stator motor and the frame structure.

**Assumption 3.** *Given the acoustic leakage $O$, the intensity of the radiated power will vary according to the direction of the nozzle movement in different directions from the audio device.*

According to the inverse square law, the intensity of the sound decreases drastically with the square of the distance from the sound source. If $P$ is the power of the sound source and $r$ be the distance from the sound source, then:

$$I = \frac{P}{4\pi r^2} \tag{2.9}$$

Hence, for analyzing the direction of movement, the intensity of the sound radiated by each motor and frame structure may be measured.

**Assumption 4.** *The direction of movement in Z-axis during printing is always in either a positive or negative direction.*

In additive manufacturing systems, materials are extruded layer-wise. Hence, the direction in Z-axis should always be in one direction. This allows us to exclude the estimation of direction motor in Z-axis.

### 2.4.2 Leakage Quantification

In the data-driven attack model, reconstruction of the G-code will depend on accurate estimation of the individual parameters of the G-code. Hence, in this chapter, the capability of the data-driven attack model to accurately estimate the individual parameters is focussed. The estimation is done using regression models to predict the continuous speed value, whereas, classification models are used to predict the axis information. Hence, leakage may simply be quantified by measuring the separability of the nozzle movement and prediction accuracy of the nozzle speeds in each axis. The separability is specifically measured using the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, whereas the speed prediction accuracy can be measured using the Mean Square Error (MSE). The accuracy of the classifiers can be calculated as follows:

$$Accuracy = \frac{TP + TN}{Total\ Sample} \tag{2.10}$$

Where $True\ Negative\ (TN)$ and $True\ Positive\ (TP)$ are the total number of right classifications made by the classifier.

### 2.4.3 Leakage Exploitation

The accuracy with which an adversary is able to exploit the acoustic leakage depends on their ability to estimate the functions $\hat{f}_d(.)$ . Breaking the process for estimating $\hat{g}_i$ into multiple estimation functions improves the adversarial attack model by focusing on only those parameters in $g_i$ that are required for breaching the confidentiality of the system. However, the accuracy of the attack model now becomes a function of the successful estimation of individual functions. Moreover, to completely steal the intellectual property inherent in the geometry of a 3D object, each line segment in each layer must be reconstructed with error

$e \geq e_P$. Where $e_P$ is the error introduced due to process variation. A strong attacker will be able to acquire accuracy with $e = e_P$. However, in a competitive market for products, losing even minute details about the geometry of their product can be disastrous for a company. Hence, in such scenarios, an attacker may just have to infer about the geometry of the 3D objects without 100% accuracy. Based on the domain knowledge of an attacker, they may be able to reconstruct the 3D object with further processing. However, this capability (what kind of domain knowledge an attacker might have) of an attacker is not covered in this chapter.

## 2.5 Attack Model Description

### 2.5.1 Attack Model

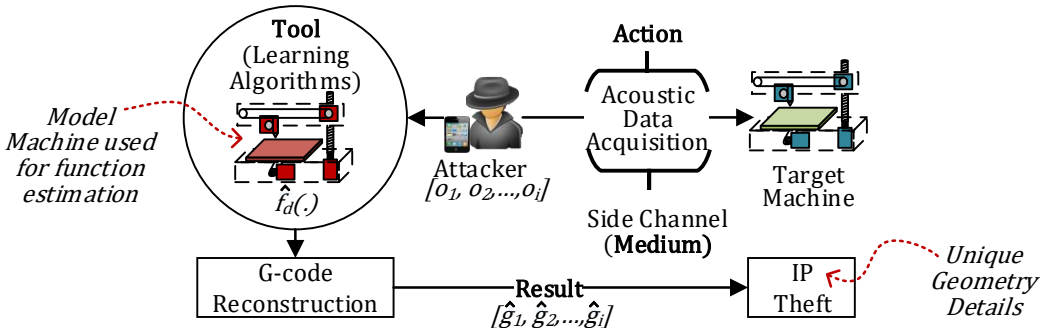



Figure 2.4: Data-driven acoustic side-channel attack model

In the attack model (shown in Figure 2.4), the intention of an attacker is to steal the geometry details of an object which is one of the intellectual properties for a company using a 3D printer. Even though these details may be stolen by acquiring the 3D object itself, our attack model is most suitable for stealing the IP during the prototyping stage, where 3D printers like the one based on FDM technique are used to visualize the geometry of the company products. An attacker may be a person who has low-level access (can be in a vicinity of the 3D printer) to the 3D printer but not to the digital process chain files (for

example STL, G-codes, etc.) itself. Moreover, they do not have any access to the digital process chain tools and software either. They will have access to a replica model of the target machine, that has to have the same physical structure as the target 3D printer. On this replica model, they can perform any sort of prior experiments to train their learning algorithms to estimate the function $\hat{f}_d(.)$. However, the learning should be performed in an environment that is as close to the target machine as possible. The learning algorithms consists of multiple estimated functions $\hat{f}_d(.)$ to predict individual parameters of a G-code. These predicted values are then combined to reconstruct the G-code, and eventually the geometry of an object.

### 2.5.2 Components of the Attack Model

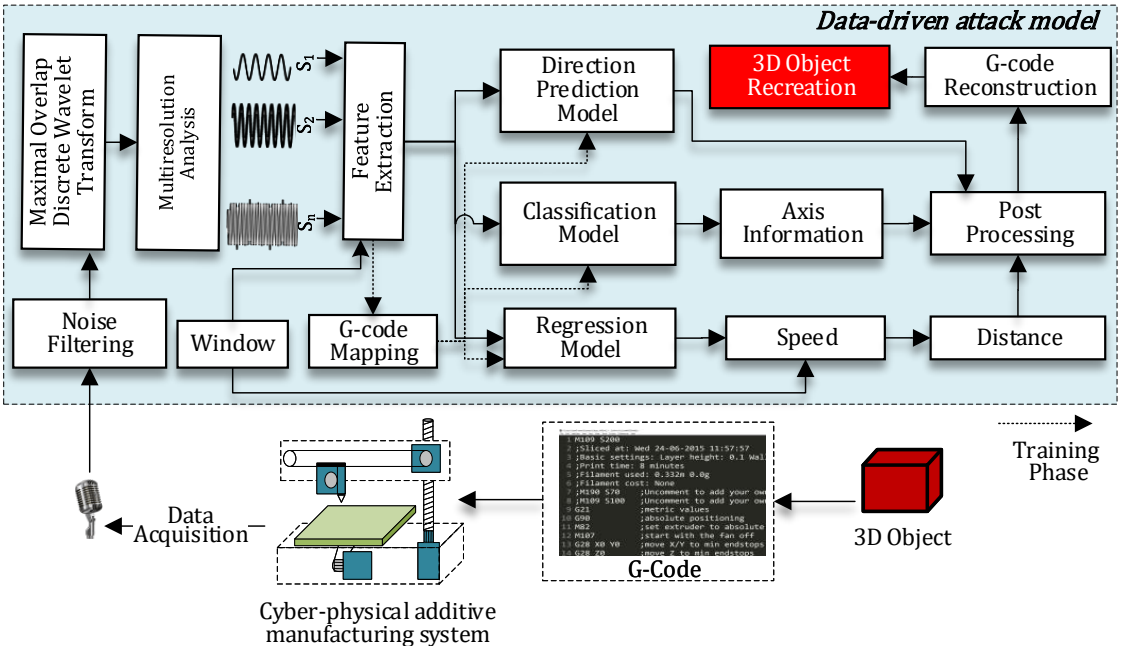



Figure 2.5: Components of the data-driven attack model

### Data Acquisition

The first step of acquiring the observable analog emissions, $[o_1, o_2, .., o_i]$, is to place an audio recording device such as a mobile phone near the 3D printer. The sampling frequency of the recording device must be higher than 40 kHz to capture the sound in the audible range to avoid aliasing effect [67]. The distance of the audio device from the 3D printer and the angle to the different sources of sound (stepper motor X and stepper motor Y) will also determine the accuracy of leakage exploitation. Moreover, in case of devices that are enclosed, a contact microphone may be utilized to estimate $\hat{f}_d(.)$. This will also reduce the influence of environmental noise on the acquired acoustic emission. For better accuracy, even multiple microphones may be placed to localize and remove environmental noise during the attack.

### Noise Filtering

A digital finite impulse response bandpass filter is used to eliminate the noise from low-frequency alternating current from the power source, and the high-frequency noise generated by the hybrid stepper motor winding when it is in charged and in idle state. The pass-band frequency for noise removal is between 100 Hz and 20 kHz.

### Maximal Overlap Discrete Wavelet Transform and Multiresolution Analysis

In [68], a fixed length window was used to extract various time and frequency domain features. With this configuration, feature extraction is more challenging. Capturing smaller movements requires smaller frame sizes to improve the temporal resolution while improving the speed and frequency resolution requires larger frame sizes. This trade-off prescribes the use of discrete wavelets transforms, to preserve both time and frequency domain features.

Maximal Overlap Discrete Wavelet Transform (MODWT) is a redundant form of the discrete wavelet transform, where a time series signal is transformed into coefficients related to the variation of a set of scales of the mother wavelet. Due to the added redundancy, MODWT makes it easier for the alignment of the decomposed wavelet and scaling coefficients at each level with the original time series. The MODWT allows us to decompose the time series data into multiple levels of scales and different frequency regions without having to worry about the length of the window. These signals are then passed through the multiresolution analysis block to decompose it into multiple signals $(S_1, S_2, \ldots, S_n)$ for feature extraction. In [68], uniform frequency scales are used to extract features using *Short-Term Fourier Transform* (STFT). Even though, *Mel-Frequency Cepstral Coefficients* (MFFC) are used in [68], which uses non uniform scaling focusing on higher number of frequency features in the lower frequency range, it is found that this feature extraction is not feasible for the 3D printer acoustic analysis due to variation of frequency distribution based on the travel feed-rate. Hence, analysis of signals decomposed by MODWT is performed to define the non-uniform frequency scales for feature extraction. Moreover, Five levels of MODWT signals is computed to define the non-uniform frequency scales.

**Feature Extraction**

Features commonly used in speech pattern recognition [69] in the time and frequency domains is used to train the learning algorithms. In the time domain, the features extracted are *frame energy*, *Zero Crossing Rate (ZCR)*, and *energy entropy* [69]. The features extracted from the frequency domain are *spectral entropy* and *spectral flux*. Multiple signals $(S_1, S_2, \ldots, S_n)$ are obtained from the multiresolution analysis block. For each signal, varying number of frequency scales are allocated to extract spectral energy values. Two frame sizes, 50 $ms$ and 20 $ms$, are used for testing the performance of the attack models with MODWT and STFT based features. From each frame, Features are extracted to create a feature vector

and supply it to the training algorithm. For a given frame of length $F_L$ with audio signals $x(i) = 1, 2...F_L$, different features are extracted as follows:

$$Frame\ Energy\ (E) = \sum_{i=1}^{F_L} \mid x(i) \mid^2 \tag{2.11}$$

Frame energy is enough to predict direction when the printer is only printing in one axis, however spectral energy is required while predicting the direction in multiple axes movement. ZCR is calculated as follows:

$$ZCR = \frac{1}{2F_L} \sum_{i=1}^{F_L} \mid sign[x(i)] - sign[x(i-1)] \mid^2 \tag{2.12}$$

ZCR is high when the printer is not making any sound, due to the noise, and low when it is printing. For energy entropy, the frame is divided into short frames of length $K$. If $E_j$ is the energy of the $j^{th}$ short frame then:

$$Energy\ Entropy = -\sum_{j=1}^{K} e_j log_2(e_j), \quad e_j = \frac{E_j}{\sum_{i=1}^{K} E_i} \tag{2.13}$$

Energy entropy measures the abrupt change in the energy of the signal, and may be used to detect the change of motion. For frequency domain data, let $X_i(k),\ k = 1, ..., F_L$ be the magnitude of the Fast Fourier Transform (FFT) coefficient of the given frame. For spectral entropy, the spectrum is divided into L sub bands. Let $E_f$ be the energy of the $f^{th}$ sub band then:

$$Spectral\ Entropy = -\sum_{f=1}^{L-1} n_f log_2(n_f), \quad n_f = \frac{E_f}{\sum_{j=1}^{L-1} E_j} \tag{2.14}$$

Spectral flux measures spectral change between two successive frames, and may be used to detect the change of speed of the nozzle while printing within each layer.

$$Spectral\ Flux_{i,i-1} = \sum_{k=1}^{F_L} (EN_i(k) - EN_{i-1}(k))^2, \quad EN_i(k) = \frac{X_i(k)}{\sum_{j=1}^{F_L-1} X_i(j)} \tag{2.15}$$

### Regression Model

The regression model consists of a collection of models, each using a supervised learning algorithm for regression as shown in Figure 2.6. These models are used for estimating the functions $v_{x_i} = \hat{f}(o_i, \alpha)$ and $v_{y_i} = \hat{f}(o_i, \alpha)$. These functions are used to extract information about speed in X direction given only one axis movement, and speed in X direction given the motion in two axis. Similarly, this is done for speed in Y direction as well.

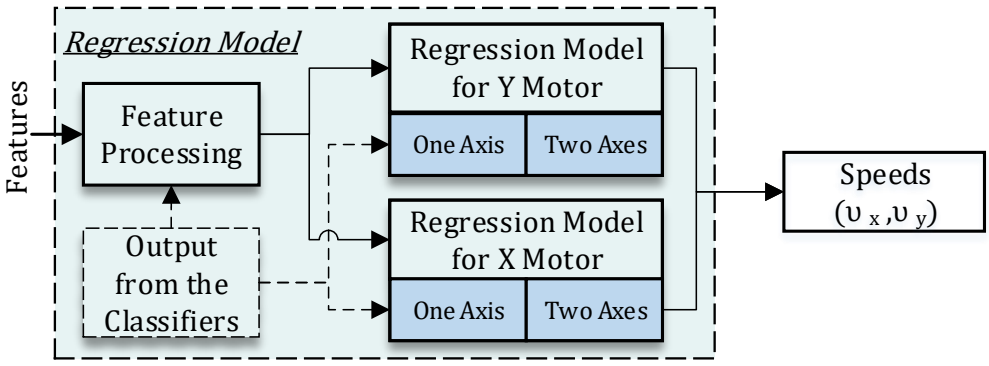



Figure 2.6: Regression model for motor speed prediction

**Assumption 5.** *The speed in the Z direction while printing the given model with the given printer is fixed and the speed of extrusion can be calculated as a function of layer height and nozzle diameter.*

For a given 3D printer, the layer height is assumed to be fixed. This relaxes the complexity for leakage exploitation by reducing the need for estimating speeds $v_x$ and $v_e$. The speed of the printing, also known as the *travel feedrate* is determined by training these regression

algorithms [70]. After gaining the information about the *travel feedrate*, the distance moved by the nozzle can be calculated as follows:

$$Distance = Framesize\ (ms) \times Speed\ (mm/ms) \tag{2.16}$$

**Algorithm 1:** Feature processing and speed calculation with motion in XY axes

**Input:** Feature Vectors $xy_\beta, x_\beta, y_\beta$ // $\beta \rightarrow$ Total features
**Output:** Speed $\vartheta_x, \vartheta_y$ // $\vartheta \rightarrow$ Speed

1 $\vartheta^i_{x_{mean_{\beta j}}} = \frac{1}{N_i} \sum_{n=1}^{N_i} x_{\beta_{n,j}}$ // $j = 1 \longrightarrow \beta n$ $N_i \longrightarrow Total\ data\ for\ speed\ i$
2 $\vartheta^i_{y_{mean_{\beta j}}} = \frac{1}{N_i} \sum_{n=1}^{N_i} y_{\beta_{n,j}}$
3 **for** *each xy* **do**
4  **for** $\vartheta^i$ *in* $range(v^1, v^n)$ // $n$:Total number of speed used in training
5  **do**
6  $xy_{beta(xy-y)} = xy_{beta} - \vartheta^i_{y_{mean_\beta}}$
7  $\vartheta^i_x \leftarrow RegressionModel1(xy_{beta(xy-y)})$
8  $xy_{beta(xy-x)} = xy_{beta} - \vartheta^i_{x_{mean_\beta}}$
9  $\vartheta^i_y \leftarrow RegressionModel2(xy_{beta(xy-x)})$
10  $diff_i = \mid \vartheta^i_y - \vartheta^i \mid$
11  $\vartheta_y = \vartheta^i_y with\ minimum\ diff_i$
12  $\vartheta_x = \vartheta^i_x with\ minimum\ diff_i$
13 **return** $\vartheta_x, \vartheta_y$

When the nozzle is moving in only one axis, the regression model may just take the features directly without further processing, however, when the nozzle is moving in two or more axes, the audio signal from one motor is combined with the others. Hence, it becomes imperative to separate these signals before the regression model can be used to predict the speed. Algorithm 1 provides the pseudo code for performing the spectral subtraction necessary when motion is involved in both the X and Y axes. It takes features, extracted from the audio when both the X and Y motors are running, and the features from the training phase for individual motor X and Y as the input. Spectral subtraction is not performed for Z motor because it only moves one layer at a time and the distance it moves is normally fixed for a given object. While training, $n$ number of speeds, in incremental number is taken to train the regression models. For each of these speeds, lines 1 and 2 calculate the average magnitude of spectral features. Then, for each of the speeds, line 6 assumes the speed of the

Y motor and the spectral components are subtracted from the combined spectral features of X and Y. By subtraction, the spectral components present in Y is removed from the combination of these features. Line 7 gives the predicted speed for the given value of speed in the Y direction. This speed is used to subtract the spectral features of X in the particular speed and again use this value to predict the speed for motion in Y-axis. In lines 11 and 12, the speed of X and Y that gives the minimum difference in the predicted speed and output speed in Y-axis is chosen as an output.

### Classification Model

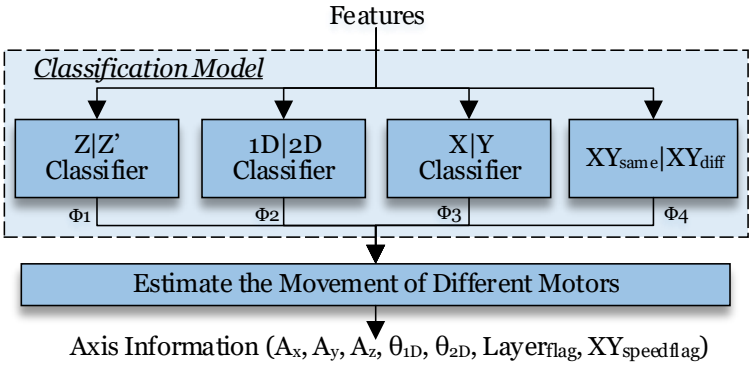



Figure 2.7: Classification model for axis prediction

As shown in Figure 4.5, to determine the axis in which the nozzle is moving, the classification model consists of a collection of classifiers to convert the classification problem into two-class separation model. This in fact will estimate the functions $a_{x_i} = \hat{f}(o_i, \alpha)$, $a_{y_i} = \hat{f}(o_i, \alpha)$, and $a_{z_i} = \hat{f}(o_i, \alpha)$. It is found that this model gives better prediction results than multi-class classifier models. Each of these classifiers consists of supervised learning algorithms for classification. Algorithm 2 gives the pseudo code which takes the output from the classifiers to determine the axis of movement. It also gives information such as whether the layer has changed or not, and whether the nozzle is moving in X and Y axis with the same or different speed.

**Algorithm 2:** Estimate the axis of movement

**Input:** Classifier Outputs $\phi_1, \phi_2, \phi_3, \phi_4$
**Output:** Axis Parameters $A_x, A_y, A_z, \Theta_{1D}, \Theta_{2D}, Layer_{flag}, XY_{speedflag}$

$\Theta_{1D} = 0, \Theta_{2D} = 0$ // $A \longrightarrow axis, \Theta \longrightarrow dimension$
$Layer_{flag} = 0, XY_{speedflag} = 0$
$A_x = 0, A_y = 0, A_z = 0$ // Initialize to zero
**if** $\phi_1 == 1$ **then**
    $Layer_{flag} = 1, A_z = 1$
**else**
    **if** $\phi_2 == 1$ // One dimension movement
    **then**
        $\Theta_{1D} = 1$
        **if** $\phi_3 == 1$ // Movement in X-axis
        **then**
            $A_x = 1$
        **else**
            $A_y = 1$ // Movement in Y-axis
    **else**
        $\Theta_{2D} = 1, A_x = 1, A_y = 1$
        **if** $\phi_4 == 1$ // X and Y move with same speed
        **then**
            $XY_{speedflag} = 1$
        **else**
            $XY_{speedflag} = 0$ // Different speed

**return** $A_x, A_y, A_z, \Theta_{1D}, \Theta_{2D}, Layer_{flag}, XY_{speedflag}$

**Direction Prediction Model:**

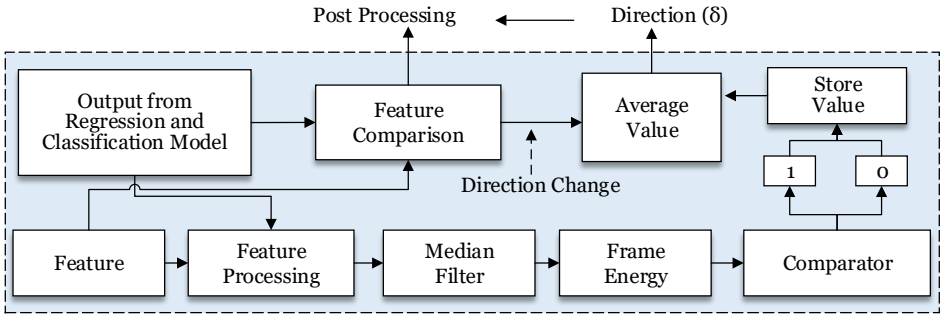



Figure 2.8: Direction prediction model

Most of the 3D printers have motors in a fixed location. However, the base plate, the nozzle or combination of both are always in motion while printing. Therefore, vibration is conducted from the motor to the nozzle and the base plate of the printer. This means that the audio source physically gets closer or away from the recording device while printing. The frame

energy of the audio signal can be used to check the direction of motion. For multiple motor movements, the difference of feature in the frequency domain is used to calculate the energy of only those spectral components that represent the specific motor. In order to suppress the high fluctuation, median filtering is applied to the sequence of frame energies to smooth the curve of frame energies. The prediction model will output 1 if the frame energy is increasing and 0 if the frame energy is decreasing. In order to aid the direction prediction model and the post-processing, a feature comparison block measures the distance (Euclidean distance) between consecutive frame features. If the motion of direction changes, then there is a large difference in the features between the consecutive frames. This spike is used to detect the change in the direction of motion of the nozzle.

**Algorithm 3:** Calculate distance moved in each axis, and check extrusion

**Input:** Output from Classifier and Regression Models $\vartheta_x, \vartheta_y, \vartheta_z$,w, $A_x, A_y, A_z, \delta_x, \delta_y, \delta_z$
**Output:** Distance Values $d_x, d_y, d_z, d_E$ // $d \rightarrow Distance, x_\vartheta \rightarrow Speed\ in\ X\text{-}axis$

$d_x = 0, d_y = 0, d_z = 0$ // $w \rightarrow Frame\ length, A \rightarrow AxisFlag, \delta \rightarrow Direction$
**for** *each* $i$ *in* $x, y, z$ **do**
  **if** $A_i == 1$ // Axis flag set
  **then**
    **if** $\delta_i == 1$ **then**
      $d_i = \vartheta_i \times w$ // Positive distance
    **else**
      $d_i = \neg\vartheta_i \times w$ // Negative distance
**if** $\vartheta_x >= Speed_{High}$ || $\vartheta_y >= Speed_{High}$ **then**
  $d_E = 0$ // *No extrusion in high speed*
**else**
  $d_E = e_d$ // $e_d \longrightarrow Machine\ specific\ extrusion$
**return** $d_x, d_y, d_z, d_E$

### Model Reconstruction:

For reconstructing the G-code, it is necessary to determine whether the 3D printer nozzle is actually extruding the filament or not. From the analysis, it was found that the printer nozzle moves at a higher speed when it is not extruding the filament. Hence, determining whether it is printing or not printing becomes a task of finding out the speed at which the nozzle is moving. This information is acquired from the regression model. The extrusion

amount for a given segment is machine-specific, and can be calculated as a function of the layer height, and the nozzle diameter. After acquiring the output from the regression model, classification model, and direction prediction model, Algorithm 3 calculates the positive or negative distance movement. Finally, Algorithm 4 reconstructs the G-code for the printed object.

**Algorithm 4:** Generate G-code of the object

**Input:** Distance and Frame Length $d_x, d_y, d_z, d_E, w$
**Output:** *G-code*

1 $dr_x = 0, dr_y = 0, dr_z = 0, dr_E = 0$ // Initialize to zero
2 $\vartheta = \frac{\sqrt{d_x^2+d_y^2+d_z^2}}{w}$ // Travel feedrate
3 $dr_x = dr_x + d_x$ // Distance moved in X-axis
4 $dr_y = dr_y + d_y$ // Distance moved in Y-axis
5 $dr_z = dr_z + d_z$ // Distance moved in Z-axis
6 $dr_E = dr_E + d_E$ // Extrusion amount
7 *G-code* $\leftarrow G1\ F(\vartheta)\ X(dr_x)\ Y(dr_y)\ Z(dr_z)\ E(dr_E)$
8 **return** *G-code*

### Post-Processing for Model Reconstruction

There is high mutual information between the G-code and the sound retrieved from the physical medium. For G-codes, let $G$ be a discrete random variable with $f(g)$ as its probability distribution function at $g$. Let $O$ be a discrete random variable representing the feature extracted from the acoustics with $f(o)$ as its probability distribution function. Then the entropy of each of these random variables may be given as:

$$H(G) = -\sum_{g \in G} f(g) log_2 f(g) \qquad H(O) = -\sum_{o \in O} f(o) log_2 f(o) \tag{2.17}$$

If $f(g, o)$ and $f(g|o)$ are the joint and conditional probabilities of the random variables, respectively, then the conditional entropy $H(G|O)$ is calculated as:

$$H(G|O) = -\sum_{o \in O} \sum_{g \in G} f(g, o) log_2 f(g|o) \tag{2.18}$$

The conditional entropy measures the amount of information required to describe the outcome of a random variable $G$, given the information about a random variable $O$. In this context, in addition to the information gathered from $O$, the amount of additional additive manufacturing context-based information required to reconstruct the G-code is directly related to the mutual information. This is calculated as:

$$I(G;O) = H(G) - H(G|O) \tag{2.19}$$

It is found that the uncertainty of reconstruction of G-code or the entropy $H(G|O)$ increases when the distance of the microphone is further away from the printer or when there is added noise in the environment. It also increases when the speed of the printer is high and there are more short and rapid movements. During these scenarios, we can use the properties of additive manufacturing to post-process the data achieved from the learning algorithms. Specifically, we have used two post-processing stages which utilize specific additive manufacturing context-based information.

*Post-Processing Stage I*: In this stage, $H(G|O)$ is reduced by utilizing the fact that until the change of motion occurs, the nozzle moving in one particular dimension with a particular speed has a similar feature vector. By taking the output from the feature comparison model, the acquired acoustic data is segmented into sections with a similar movement. In this post-processing stage, then the output of the classifiers is chosen to be the highest occurring value in the given segment, and for regression, the speed obtained is averaged within the same section. This is similar to averaging used in digital signal processing to increase the Signal-to-Noise Ratio (SNR).

$$SNR_{dB} = 10log_{10}(\frac{Power_{signal}}{Power_{Noise}}) \tag{2.20}$$

When the SNR is increased, the entropy of the signal is reduced. As there is a high correlation

among the features extracted from successive frames of the audio collected from the 3D printer, averaging the output of the classification and the regression model increases the SNR and thus reduces $H(G|O)$.

*Post-Processing Stage II*: After applying post-processing stage I, the second stage measures the similarity between the two layers. The similarity of two layers is measured in terms of the number of segments, the sequence of motions in each layer, and the length of each segment. This post-processing stage helps the attack model in reducing the error due to the miscalculated direction and fluctuating lengths by taking the average of segment lengths and direction among the similar layers of the 3D object.

### 2.5.3 Attack Model Training and Evaluation

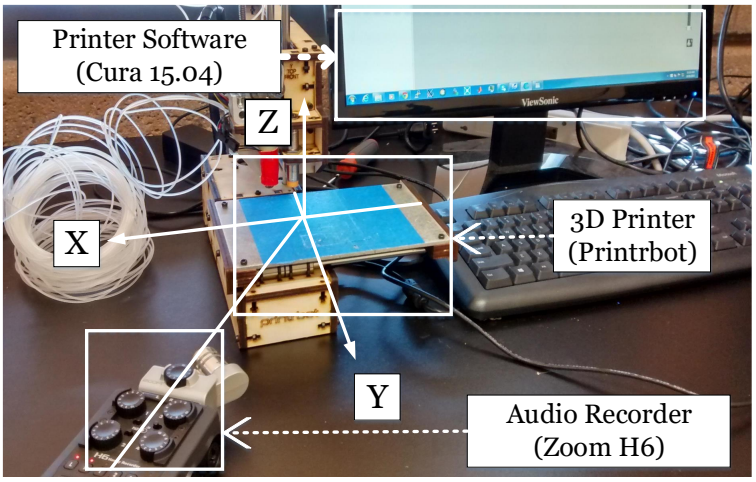



Figure 2.9: Setup for training and testing

*Testbed for Training and Testing:* Our testbed, shown in Figure 2.9, consists of a Printrbot 3D printer [71] with open source *marlin* firmware. It has four stepper motors. Motion in the X-axis is achieved by moving the base plate, whereas the nozzle itself can be moved in the Y and Z directions. The audio is recorded using a cardioid condenser microphone (Zoom H6) [72], which has a sampling frequency of 96 kHz and stores the data at 24 bit per sample. The audio recorder is placed within 20 cm of the 3D printer. From the experiments, it is

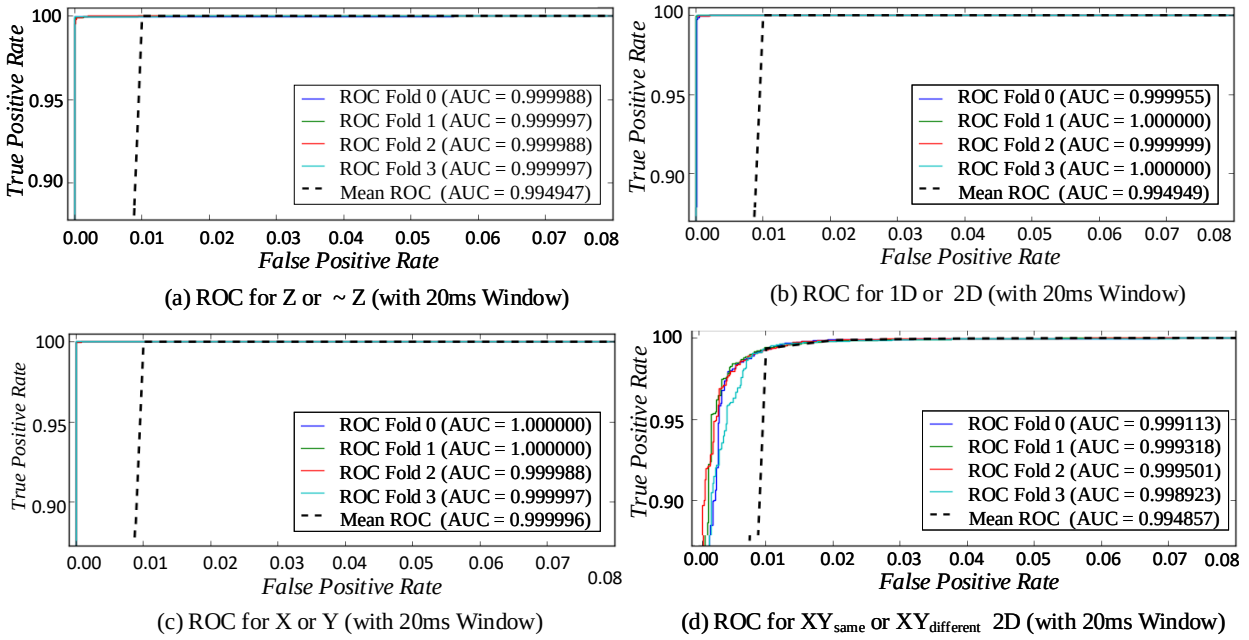
observed that for the direction prediction model to work efficiently, the audio device has to be placed at 45° angle to both the X and the Y-axis as shown in Figure 2.9. This allows the audio device to capture the variation of sound in both X and Y directions. The digital signal processing, feature extraction, and post-processing are performed in MATLAB [73], whereas the training of learning algorithms, their evaluation and testing are done using Python [74]. The attack model consists of supervised learning algorithms described in Section 2.5.2. For training these algorithms, initial training data has to be determined. The training data consists of G-code to move the printer nozzle at different speeds (500 mm/min to 4500 mm/min) and different axes. The speed range chosen is specific to the 3D printer. The G-code for training phase consists of movement in just one axis (X, Y, and Z), two axes (XY, XZ, and YZ), and all three axes (XYZ). The audio signal corresponding to each of these G-codes is pre-processed and labeled for training the learning algorithms. The total length of audio recorded for training is 1 hour 48 minutes. The total numbers of features extracted is 328 for the window size of 50 $ms$ and 318 for the window size of 20 $ms$. For regression model, *Decision Trees* boosted using *Gradient Boosting* algorithm is used, whereas for the classification model *Decision Tree Classifier* boosted using *AdaBoost* algorithm [70] is used. The learning algorithms is trained by performing K-fold cross validation, with k = 4, to test the efficiency of the learners as well as to avoid over or under fitting of the learning algorithms. In the experiments, the regression model is trained only for the nozzle movements in the X and Y directions. *Classification Models:* Table 2.1 shows the accuracy of

Table 2.1: Accuracy of the classification models

| Classification Model | Classifying | Accuracy(%) [68] win 50 ms | Accuracy(%) (MODWT and STFT) win 50 ms | Accuracy(%) (MODWT and STFT) win 20 ms |
|---|---|---|---|---|
| $\phi_1$ | $Z$ or $\sim Z$ Axis | 99.86 | 99.9559 | 99.9400 |
| $\phi_2$ | $1D$ or $2D$ Axis | 99.88 | 99.9700 | 99.9606 |
| $\phi_3$ | $X$ or $Y$ Axis | 99.93 | 99.9910 | 99.9778 |
| $\phi_4$ | $XY_{same}$ or $XY_{different}$ | 98.89 | 99.4644 | 99.2349 |

the various classifiers. The MODWT and STFT based features extraction are compared with

the features extracted in our previous work [68]. It can be observed that compared to the accuracy of the previous classifiers, the accuracy of the MODWT and STFT features based classifiers is higher. Even when the window size is reduced to 20 $ms$, the accuracy of the classifiers is still higher than the classifiers using features described in [68]. For measuring




(a) ROC for Z or ~ Z (with 20ms Window)

(b) ROC for 1D or 2D (with 20ms Window)

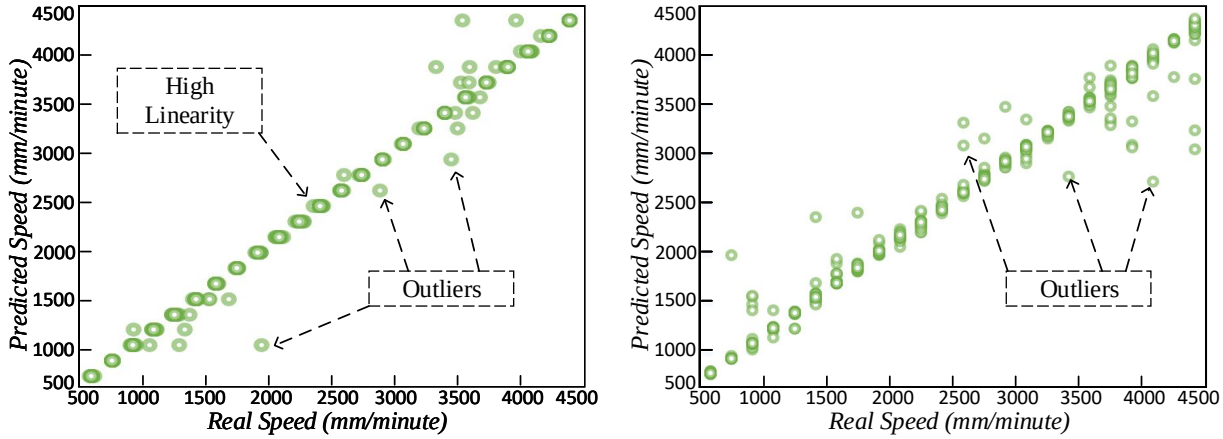
(c) ROC for X or Y (with 20ms Window)

(d) ROC for $XY_{same}$ or $XY_{different}$ 2D (with 20ms Window)

Figure 2.10: Receiver operating characteristics curve for classifiers

the accuracy of the classifiers, Receiver Operating Characteristics (ROC) curves are also analyzed. The classifiers capability to separate the two classes is high if the graph lies closer to the upper right corner. This region corresponds to 100% sensitivity (zero false negatives) and 100% specificity (zero false positives). As the collection of classifiers are arranged in a hierarchy, the bottleneck in terms of accuracy is the longest path followed while making the decision. In this case, the longest path involves all the classifiers. From Figures 2.10 (a)-(d), it can be observed that the classifiers have a high sensitivity and specificity with a high Area Under the Curve (AUC). It means the different classes can be accurately classified based on the observed leakage from the side-channel. The AUC for the classifier classifying whether the movement X and Y axis have the same speed or different speed is comparatively less than other AUCs. This is intuitive as, in multiple axis movement, separation of individual

movement is difficult. *Regression Models:* The accuracy of the regression model is measured




(a) Speed Prediction for X with one Axis Motion

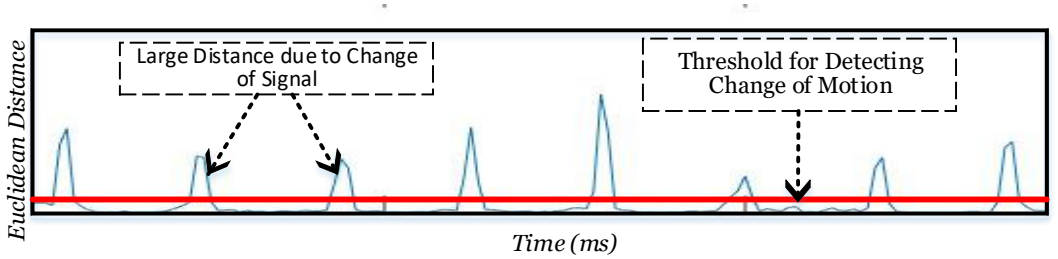
(b) Speed Prediction for Y with one Axis Motion

Figure 2.11: Prediction results for regression models

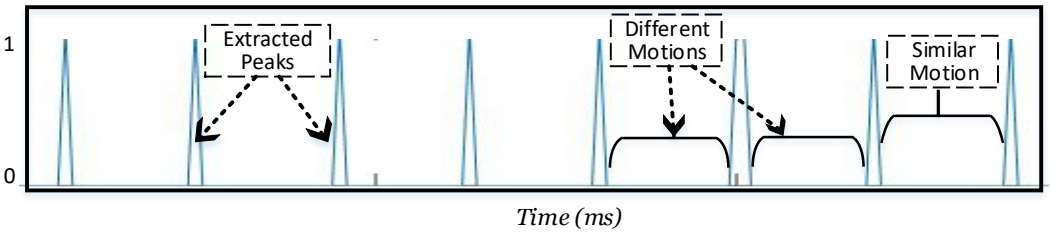
in terms of Mean Square Error (MSE) with the data normalized with zero mean and unit variance. The mean absolute error is presented to understand how the speed prediction varies from the real speed. From Table 2.2, it can be seen that the MSE with new features is less than compared to the MSE of the regression models used in our previous work [68]. With the new features, lower MSE is achieved when the window size is reduced to 20 $ms$. Lower window size allows us to have higher 3D object reconstruction accuracy when the dimension of the 3D object being printed is lower. It can be seen that the MSE is relatively higher for the value predicted by the regression model for the motion in Y-axis when the motion is occurring at two axes. However, this error can be removed during the post-processing stage as the travel feed rate is generally similar between consecutive frames in each layer of printing. Figure 2.11 shows that there is a linear relationship between the real speed and the predicted speed computed by the regression model. Figure 2.12 shows the feature comparison conducted for the audio recorded while the 3D printer is printing an object. It can be observed that when the nozzle changes its direction by analyzing the distance of features between successive frames. Peaks are extracted by applying the threshold obtained during the training phase. A value higher than the threshold is 1, otherwise 0.

Table 2.2: Accuracy of the regression models

| Regression Model | Movement Axis | MSE [68] Normalized (win 50 ms) | MSE Normalized (MODWT and STFT)(win 50 ms) | Mean Absolute Error (win 50 ms) (MODWT and STFT) *mm/minute* | Mean Absolute Error (win 20 ms) (MODWT and STFT) *mm/minute* |
|---|---|---|---|---|---|
| X | Only X | 0.00616 | 0.00292 | 5.876 | 8.6786 |
| Y | Only Y | 0.01874 | 0.01201 | 18.3721 | 23.5821 |
| X | X and Y | 0.1658 | 0.04641 | 91.4723 | 140.045 |
| Y | X and Y | 0.4290 | 0.25120 | 268.07 | 294.0423 |




(a) Distance Between Features of Consecutive Frames




(b) Peak Extracted from the Distance Measurement

Figure 2.12: Features Segmented with varying motion

## 2.6 Results for Test Objects

In order to test our attack model, various benchmark parameters which affect the accuracy of the attack model are defined. While printing, multiple similar 2D layers are printed to achieve a 3D object. Hence, our test objects also consist of 3D objects with simple 2D geometry (such as square, triangle, etc.) repeated over multiple layers. The various benchmark parameters in designing these test objects are as follows:

***1. Speed of Printing:*** The fixed frame rate affects the temporal and spectral features extracted from the audio. With the increase in the speed, faster rate of change of spectral features will not be captured and this can degrade the performance of the attack model.

Hence, the speed of printing is varied to test the accuracy of the attack model.

***2. The dimension of the Object:*** With smaller objects, shorter nozzle movements are present. To represent these shorter movements, the temporal resolution of the features is increased by making the frame size smaller. To test our attack model with smaller objects, we vary the size of the object being printed.

***3. The complexity of the Object:*** Printing a complex object incorporates movement in more than one axis. Hence, to increase the complexity of the object being created, we have tested the acoustic model with shapes consisting of simultaneous multiple axis movement, such as a triangle.

For each of the test objects, to test the reconstruction capability of the attack model for 3D objects, we have printed it in five layers. Since the attack model uses post processing to remove any layer that is not overlapping with the previous layers, having larger number of layers with same base 2D outline throughout would increase the accuracy of the attack model. Hence, we have chosen smaller number of layers as a proof of concept, and for demonstration purpose. In order to provide the result in a meaningful manner, instead of calculating the mean square error, the Mean Absolute Percentage Error (MAPE) is used for the distance prediction.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \mid \frac{A_t - P_t}{A_t} \mid \tag{2.21}$$

Where $A_t$ is the actual speed and $P_t$ is the speed predicted by the attack model. Since the frame size (20 ms) is same for all the features, the distance calculation error is also be given by Equation 2.21. Table 2.3 provides results for the different parameters used to test the accuracy of the attack model. The average classification accuracy and regression MAPE before the post-processing stage are 74.43% and 13.17%, respectively. This is an improvement over the results of our previous work [68], where the classification accuracy is 66.29% and

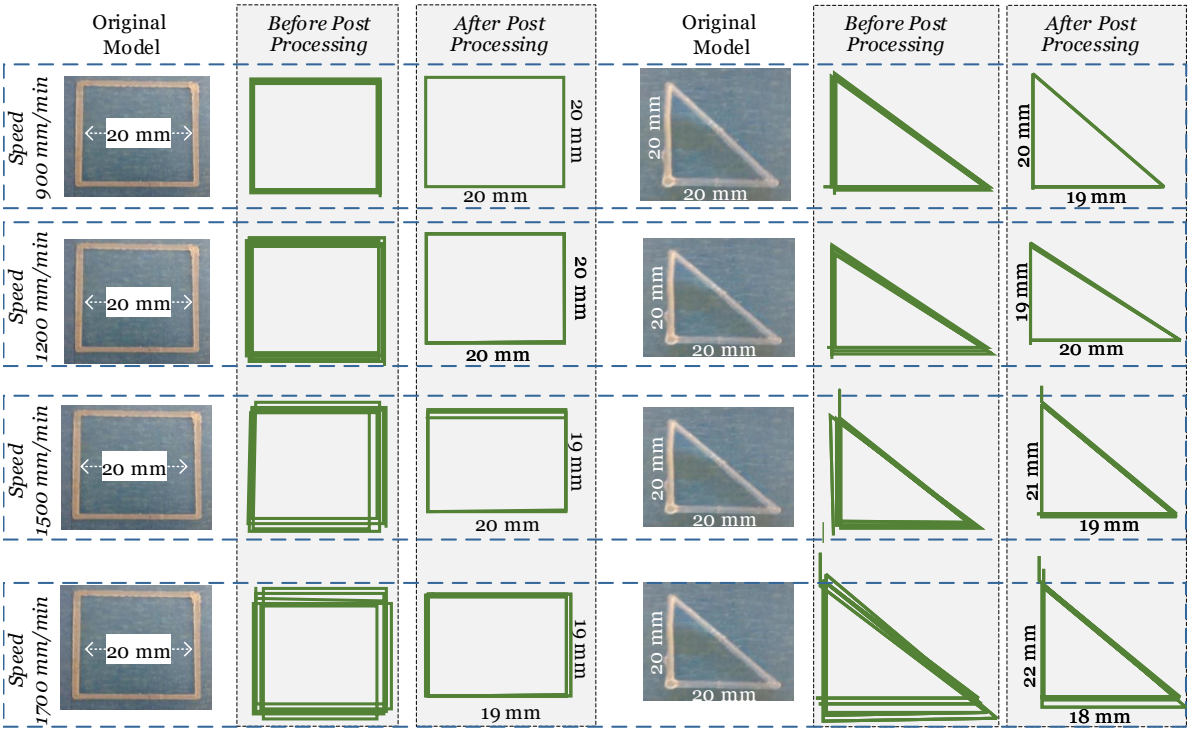
regression MAPE is 20.91%. After the post-processing stages, the classification accuracy is 86%, and the regression MAPE is 11.11%. This result is also an improvement over the classification accuracy of 78.35% and regression MAPE of 17.82% after post processing stages presented in our previous work [68] due to better feature extraction achieved by combination of MODWT and STFT.

Table 2.3: Test results for square and triangle

| | Dimension ($mm$) | Speed ($mm/min$) | Regression MAPE (%) | Classification Accuracy (%) | Classification Accuracy App I (%) | Classification Accuracy App II (%) | Regression MAPE App II (%) |
|---|---|---|---|---|---|---|---|
| Square (side) | 20 | 900 | 4.83 | 92.56 | 99.51 | 99.51 | 3.39 |
| | | 1200 | 6.80 | 91.82 | 97.68 | 97.68 | 5.14 |
| | | 1500 | 9.37 | 85.91 | 89.34 | 92.12 | 7.65 |
| | | 1700 | 15.47 | 82.43 | 88.27 | 88.27 | 11.21 |
| | 10 | 900 | 4.97 | 89.28 | 98.55 | 98.55 | 3.67 |
| | | 1200 | 7.58 | 83.66 | 87.81 | 88.61 | 6.79 |
| | | 1500 | 11.88 | 78.81 | 86.11 | 90.23 | 9.81 |
| | | 1700 | 18.67 | 74.93 | 82.12 | 85.31 | 14.32 |
| | 5 | 900 | 8.72 | 76.72 | 84.63 | 86.62 | 5.61 |
| | | 1200 | 12.64 | 69.44 | 76.22 | 76.22 | 8.91 |
| | | 1500 | 17.44 | 62.05 | 74.07 | 74.07 | 12.57 |
| | | 1700 | 23.91 | 57.29 | 62.64 | 62.64 | 18.99 |
| Triangle (base, height) | 30,20 | 900 | 4.48 | 90.85 | 98.71 | 98.71 | 4.46 |
| | | 1200 | 5.19 | 88.70 | 95.06 | 96.34 | 5.11 |
| | | 1500 | 6.85 | 85.15 | 91.18 | 91.18 | 6.73 |
| | | 1700 | 10.33 | 78.77 | 84.30 | 84.30 | 9.82 |
| | 20,20 | 900 | 5.45 | 87.50 | 94.77 | 97.35 | 4.80 |
| | | 1200 | 7.41 | 84.01 | 90.14 | 90.14 | 6.37 |
| | | 1500 | 9.22 | 79.99 | 85.61 | 85.61 | 8.88 |
| | | 1700 | 16.02 | 77.41 | 79.39 | 83.13 | 14.25 |
| | 10,5 | 900 | 16.57 | 70.84 | 78.42 | 82.37 | 14.77 |
| | | 1200 | 21.43 | 68.23 | 74.90 | 74.90 | 19.69 |
| | | 1500 | 32.78 | 63.93 | 72.28 | 74.44 | 28.55 |
| | | 1700 | 38.41 | 58.51 | 65.71 | 65.71 | 35.21 |
| **Average** | | | **13.17** | **74.43** | **84.89** | **86.00** | **11.11** |
| **Average [68]** | | | **20.91** | **66.29** | **76.04** | **78.35** | **17.82** |

### 2.6.1 Reconstruction of a Square

A square incorporates movements of stepper motors in all axes, however, one at a time. From Table 2.3, it can be seen that the accuracy of the classifier for reconstructing the G-code is as high as 92.56% with MAPE of just 4.83%. After post-processing stages the same accuracy has been increased to 99.51% for the classifier with MAPE of just 3.39%. It can also be observed that as the *travel feedrate* increases to 1700 mm/min, the accuracy of the classifier and the regression model decreases. Also for short movements such as 5mm, the accuracy of the attack model decreases. Figure 2.13 (a) shows the square reconstructed by the attack model for a side length of 20 mm.




(a) Square Reconstruction (b) Triangle Reconstruction

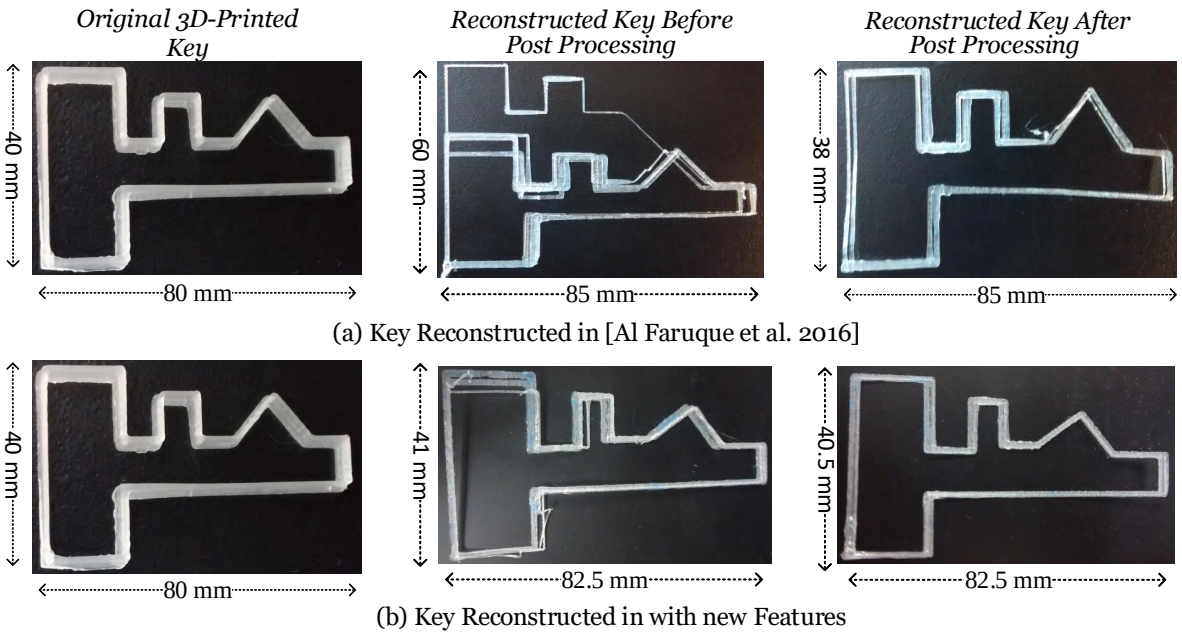
Figure 2.13: Reconstructed test objects

### 2.6.2 Reconstruction of a Triangle

While constructing a triangle, both the X and Y stepper motors move, and this affects the reconstruction accuracy of the attack model. From Table 2.3, it can be seen that the accuracy of the classifier of the attack model is as high as 90.85% with MAPE of just 4.48% before post-processing and after post-processing, they are 98.71% and 4.46%, respectively. As expected, the accuracy of the learning algorithms decreases with increasing speed and decreasing length of the movement. Also, classification and regression accuracy of reconstructing the triangle is less when compared to the square. Figure 2.13 (b) shows the reconstructed shape of the triangle. For higher travel feed-rates, the accuracy of the reconstruction is lower.

### 2.6.3 Case Study: Outline of a Key

As a case study, to test the attack model against a combination of shapes, an object representing the outline of a key at 900 mm/min *travel feedrate* is printed. The classification accuracy obtained for this object before post-processing is 88.83% and the regression MAPE is 5.62%. This is better than the results presented in our earlier work [68], where the classification accuracy obtained before the post-processing is 83.21% and the regression MAPE is 9.15%. After the post-processing, the classification accuracy obtained is 96.32% and the regression MAPE obtained is 3.92%. This is also an improvement over the result presented in our previous work [68], where the classification accuracy obtained is 92.54% and the regression MAPE obtained is 6.35%. The object reconstructed by the attack model is shown in Figure 2.14. As can be seen, before post-processing stage II, there are some non-uniform lengths in each of the layers of the object. However, after post-processing stage II, these errors are corrected. In terms of dimension, it can be observed that the reconstructed key varies in length and width compared to the original object. Nevertheless, the general outline of the key is reconstructed accurately. Moreover, the accuracy in terms of the length




(a) Key Reconstructed in [Al Faruque et al. 2016]

(b) Key Reconstructed in with new Features

Figure 2.14: Reconstruction of a key as a case study

obtained after the post-processing stage is 92.48%, which is calculated by dividing the difference between the original length and the predicted length of each segment of each of the layer by the total length of all the segments in all the layers. This is better than the results presented in [68], where the accuracy is 89.72%. On increasing the travel feed rate for the given test case, it can be expected to have higher distortion as demonstrated by the results for the reconstruction of test objects such as square and triangle. Where increasing the speed caused drastic misalignment of layers, error in direction prediction, and imprecise dimension prediction.

## 2.7 Discussion

***Technology Variation:*** In this chapter, the experiment was performed using a FDM technology based 3D printer. The key assumption for the attack model is that there is a correlation between the G-code and the radiated sound, and an attacker is capable of acquiring these radiated signal. With this, any 3D printer that is capable of suppressing the acoustic emission from the printer can effectively avoid this kind of attack. For example, other

3D printing technologies such as StereoLithography (SLA), Selective Laser Melting (SLM), Electron Beam Melting (EBM), etc., that uses UV-laser, high beam laser, electron beam etc., to harden either the liquid resin or metal powder, have minimum number of components that can generate acoustic emissions. In such scenarios, different analog emissions (magnetic, power, thermal, etc.) may be used for breaching the confidentiality of the system.

***Sensor Position:*** In the experiment, the sensor is placed at a fixed position (20 cm) from the 3D printer. However, the sensor position plays an important role in the accuracy of the system. For instance, if the sensor is placed on the top parallel to Z-axis of the 3D printer, the variation in X and Y axis direction will be difficult to capture. A better sensor position exploration can be done to understand how the position affects the accuracy of the attack model.

***Sensor Number:*** In this chapter, only a single acoustic sensor is used to acquire the acoustic emissions. However, a motivated attacker might have multiple sensors placed around the 3D printer to acquire the signals. This may change the accuracy of the attack model. It will help in better localization of the source of acoustic, and remove other environmental noise (for example using blind source separation method). However, the angle and distance between these sensors have to be explored before it can be used for data acquisition.

***Dynamic Window:*** Due to the fixed frame size incorporated for feature extraction in our experiments, the accuracy of the attack model is reduced for higher speeds and smaller dimensions. In order to capture smaller movements, the temporal resolution of the features extracted has to be increased by making the frame size smaller. However, for faster speeds, larger frame size is necessary to increase the frequency resolution for better spectral features. This trade-off dictates that an adaptive frame size needs to be incorporated to increase the accuracy of the attack model.

***Feature Separation during Multiple Axis Movement and Noise:*** The separation of

the sound source from a combination of sound is a well-known problem in speech processing. In our attack model, spectral subtraction is incorporated to acquire features that are unique to each of the stepper motors. However, there are other sound separation methods [75, 76]. Moreover, only one audio sensor is used for separating two features. Incorporating two or more audio sensors may improve the results further.

***Target Machine Degradation:*** Over a longer period of time, due to mechanical degradation, the vibration produced by the 3D printer will vary compared to the new models. Since an attacker cannot access the target model for estimating the leakage function, they might have less accuracy in stealing the information. However, one possible solution to tackle this issue would be to continuously update the model function using a 3D printer model that closely handles the same workload as is done in the industry through accelerated aging test methods, and capture the degradation trend to isolate any noise caused by the mechanical wear and tear that does not aid in information leakage.

## 2.8 Summary

This chapter introduces and demonstrates a novel data-driven acoustic side-channel attack model on cyber-physical additive manufacturing systems. An analysis on sources of acoustic emission in a fused deposition modeling technique based 3D printer is presented, and leakage analysis is performed to highlight the parameters of G-code that can be inferred from the acoustic side-channel. Maximum overlap discrete wavelet transform is incorporated in feature extraction to acquire better G-code and 3D object reconstruction results than short term Fourier transform and Mel-frequency cepstral coefficient features based attack model. The validation of our novel attack model with a state-of-the-art 3D printer shows that objects with different benchmark parameters such as speed, dimension, and complexity can be reconstructed using acoustic side-channel attacks. Our experiments show an average axis

prediction accuracy of 74.43%, and average length prediction error of 13.17%. Furthermore, with post-processing, a high average axis prediction accuracy of 86% and average length prediction error of 11.11% is achieved. As it is explained in the discussion section, there are several research challenges that remain open. The work presented in this chapter serves as a proof of concept for the possibility of physical-to-cyber attacks on cyber-physical additive manufacturing systems.

# Chapter 3

# Part I: Data-Driven Defense: Leakage Minimization

## 3.1 Introduction

In chapter 2, data-driven modeling was performed to demonstrate how novel attack models utilizing the cross-domain relationship of cyber-physical system can be used to breach the confidentiality of the system. Moreover, authors in [77, 68, 78] have demonstrated how various analog emissions such as *acoustic*, *electromagnetic*, *vibration*, and *thermal* can behave as side-channels and reveal valuable Intellectual Property (IP) [47] (such as geometry information) during the printing stage. These side-channels allow attackers to acquire the cyber-domain information (such as the printing instructions) without using the brute force approach.

There are various research works that have tackled the issue of *integrity* [79, 32, 80, 81], however, at the time of writing this chapter, only a few have worked on *thwarting* the side-channel attacks. In cyber-physical manufacturing systems, attackers are motivated to breach

the security of manufacturers for their *Intellectual Property* (IP) and *operations information* that is worth a large amount [82]. Although a large body of work has been focused in preventing duplication of the printed 3D object [83, 84] to maintain *confidentiality*, in this chapter we tackle the problem of IP loss during prototyping stage or early stage of product design, before the 3D object is accessible to an attacker. This stage can be crucial for a company as any information leakage at this stage can cause the company to lose IP to competitors [49].

### 3.1.1 Motivation for Leakage-Aware Security Tool

In cyber-physical additive manufacturing, critical information that needs to be protected are the intellectual properties hidden in the geometry of the 3D objects, machine specific processes, etc [47]. In this chapter, the unique geometric properties of the object, which gives their product an edge over other competing products in the market, is considered as the crucial IP. These geometric designs are eventually described and stored in the form of cyber-domain data, which flows through the digital process chain. The digital process chain of additive manufacturing consists of Computer Aided Design (CAD) tools for modeling the 3D objects, and Computer Aided Manufacturing (CAM) tools for converting the 3D models to slices of 2D polygons [85], and then generating tool-path (G/M-codes) based on those 2D polygons [86]. These G/M-codes are eventually converted to control signals that actuate the physical components. During actuation, mechanical and electrical energies flow through the system and may leak the information about the G/M-codes, eventually leaking the IP. An attacker may use multi-channel fusion [87] attacks to reverse-engineer the 3D geometry of an object.

Information leakage from the side-channels may be prevented by reducing the mutual information between the cyber-domain data and the physical emissions. This may be achieved

by changing the mechanical structure of the 3D printer to make emissions corresponding to multiple G/M-code actuation identical, adding noise generators to reduce the signal-to-noise ratio in the individual side-channels, or adding secure chambers. Compared to these measures which add cost, it is possible to change CAM tools to minimize the information leakage from the acoustic side-channel. The CAM tool can change process and design parameters in slicing and tool-path generation algorithm without affecting the quality of the 3D object. Moreover, costly countermeasures for intrusively changing the mechanical structure or noise generating sources are avoided.

### 3.1.2 Problem and Research Challenges

Designing a methodology to minimize information leakage in the physical domain through the incorporation of security aware solutions in the cyber-domain of the cyber-physical additive manufacturing system poses the following key challenges:

1. Determining the design variables in cyber-domain (computer aided-manufacturing tools such as slicing and tool-path generation algorithm) that can be optimized to minimize the information leakage.

2. Formulating an optimization problem that can be placed in the digital process chain, which can be generalized for all side-channels, and can balance the trade-off between the design variables and the associated costs (*leakage amount*, *printing time*, etc.).

3. Understanding the implication of information leakage from the side-channels.

### 3.1.3 Our Novel Contributions

To address the above-mentioned challenges, a novel methodology is proposed, which is capable of generating information leakage-aware cyber-physical additive manufacturing tool, and consists of:

1. **Leakage modeling and quantification for an additive manufacturing system** (**Section 3.2.1**), which highlights the relationship between the G/M-codes and the emissions in the side-channel, and performs information quantification using *mutual information*.

2. **Formulation of an optimization problem** (**Section 3.2.3**), that describes various design variables (orientation $\theta$ and travel feed-rate $v$) to optimize, and provides it as an input to the slicing algorithm and the tool-path generation algorithm in the digital process chain.

3. **A case study analysis**, for which we formulate the success rate (**Section 3.2.4**), provide an attack model (**Section 3.2.2**), and reconstruct the 3D object (**Section 3.4.2**) to visualize how leakage-aware CAM tool obstructs the reverse engineering of 3D objects from the side-channels.

## 3.2 System Modeling

The proposed leakage-aware computer aided-manufacturing tool is presented in Figure 3.1. Traditional digital process chain lacks the feedback from the physical domain, which can convey knowledge about the amount of information leaked from the side-channels. In our work, CAM tools are not only aware of the leakage but also change the design variables to minimize the leakages. The leakage quantification may be done both in design-time and run-

time. During design-time, the state-of-the-art slicing and tool-path generation algorithms are used to generate G/M-code $(g_1, g_2, \ldots, g_k)$ corresponding to the benchmark 3D models. These G/M-codes are sent to the 3D printer and the corresponding analog emissions, such

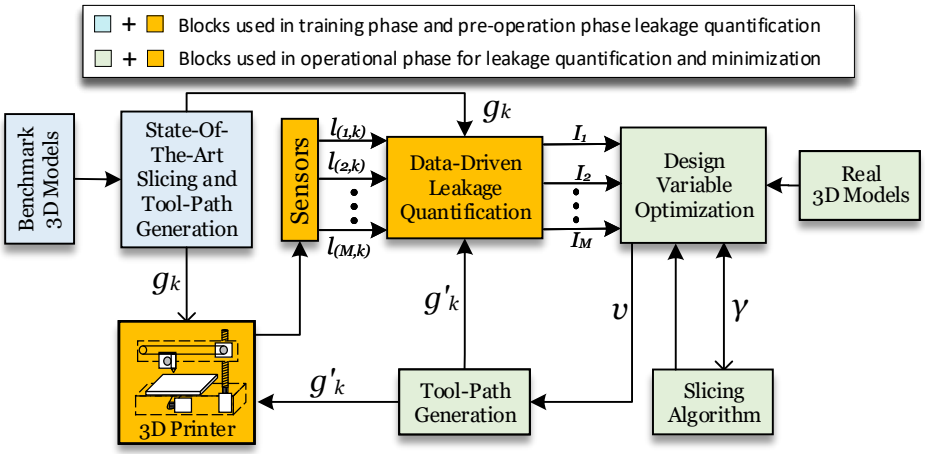



Figure 3.1: Data-driven leakage-aware computer aided-manufacturing tool

as acoustic, power, vibration, and magnetic, are collected using the sensors. The leakage values $(l_{(M,k)})$ from each side-channels (assuming there are $M$ side-channels) and the corresponding G/M-codes $(g_1, g_2, \ldots, g_k)$ are then passed for leakage quantification. The leakage quantification block calculates mutual information $(I_M)$ corresponding to each of the side-channels. These values are passed to the design variable optimization block, which optimizes the design variables (such as $\gamma$, $v$, etc.) to minimize the mutual information. Then real 3D models (which contains the IP) are passed through the design variable optimization block which modifies the slicing algorithm and tool-path generation algorithm to produce new G/M-codes $(g'_1, g'_2, \ldots, g'_k)$. During run-time, the leakage-aware CAM tool can be constantly updated to quantify the leakage and optimize the design variables. This may be necessary to make sure that the environmental condition and aging of the physical system do not aid in the leakage of the information.

### 3.2.1 Data-driven Leakage Modeling and Quantification

For modeling the leakage, it is assumed that there are $M$ side-channels. The G/M-code is the *sensitive variable*, that an attacker seeks to extract from the 3D printer. Let $G$ represent the *sensitive* discrete random variable, with probability distribution function $p(g)$, where $g_1, g_2, \ldots, g_k$ represents the possible G-code instructions. Then the leakage from each channel can be written as follows:

$$L_i = \delta_i(G) + N_i \quad i = 1, 2, \ldots, M \tag{3.1}$$

where $N_i$ denotes an independent noise (independent from the variable $G$) in the $i^{th}$ channel, $\delta(.)$ represents the deterministic function, and $L_i$ is the leakage in the $i^{th}$ channel. Moreover, for each G-code instruction $g_k$, the corresponding leakage may be given as follows:

$$l_{(i,k)} = \delta_i(g_k) + n_{(i,k)} \quad k = 1, 2, \ldots, K \tag{3.2}$$

where $n_{(i,k)}$ represents the leakage noise value in the $i^{th}$ channel for the $k^{th}$ leakage measurement, and $K$ is the total number of G-code instructions. *Mutual information* is used as a *metric* to quantify the information leakage from each of the channels independently. Given that the *joint probability distribution function* $p(g, l_i)$, marginal probability distribution $p(g)$ and $p(l_i)$ for the discrete random variables $G$ and $L_i$ is available or can be estimated, the mutual information between the G-code instruction and the leakage can be calculated as follows:

$$I(G; L_i) = \sum_{l_i \in L_i} \sum_{g \in G} p(g, l_i) log_2 \left( \frac{p(g, l_i)}{p(g)p(l_i)} \right) \tag{3.3}$$

Here, $p(g) = \sum_{l_i} p(g, l_i)$ and $p(l_i) = \sum_g p(g, l_i)$. The joint probability distribution may be calculated empirically $\hat{p}(g, l_i)$ by acquiring the leakage values in the side-channels to various

G/M-codes that are produced by the CAM tool for various 3D objects. Moreover, a data-driven modeling approach may be used to estimate the joint probability distribution as the empirical probability distribution tend to overestimate the mutual information. Furthermore, since base 2 is used for the logarithm, the unit of the mutual information is *bits*.

### 3.2.2 Attack Model

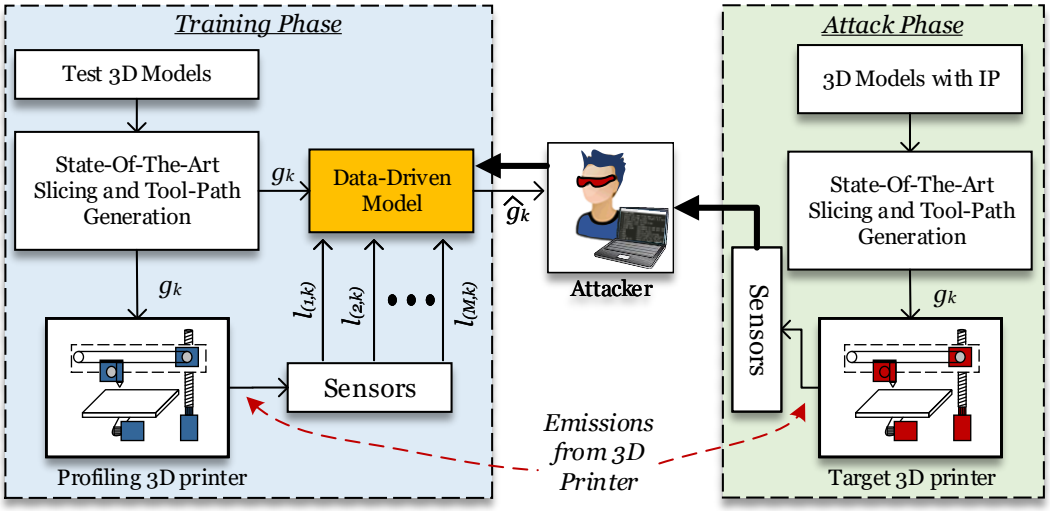



Figure 3.2: Side-channel attack model for 3D printer

The side-channel based attack model presented here for a case study is based on earlier works [77, 68, 88], that have successfully demonstrated the possibility of side-channel attacks on 3D printers. In the adversary/attack model, the main objective of an adversary is to be able to steal/infer the intellectual property inherent in the geometry of the 3D objects, which is described by the G/M-codes. As a medium, they will utilize the emissions from the 3D printer, which may behave as a side-channel. The tools used by the attackers in analyzing the side-channels and reverse engineering the G/M-codes are machine learning algorithms which allow them to estimate leakage model function from the emissions. An attacker may be a weak or a strong attacker, depending on their domain knowledge (layer wise printing, a high correlation between two consecutive layers, a dimension of the 3D printer, machine

specific M-codes, etc.). This demarcation is necessary, as geometry information may not be fully stolen from the side-channel. A weak attacker will rely on just the side-channels to estimate the G/M-codes. However, a strong attacker may be able to post-process the estimated G/M-code using domain knowledge to improve the accuracy of the reconstructed 3D objects. For strong attackers, a partial reconstruction of the geometry from itself may be enough to infer further geometry details, hence any information from the side-channel may help them in stealing partial or complete intellectual property.

Furthermore, an attacker goes through two stages to steal the cyber-domain information (G/M-code) (see Figure 3.2). The first is the training stage, in this stage, we assume that an attacker has high-level access to a 3D printer with a similar physical structure to the target 3D printer. By high-level access, we mean that he/she is able to access the digital process chain (CAD and CAM tools). Hence, the attacker is able to collect the analog emissions from $M$ side-channels corresponding to the G/M-codes. Then he/she uses data-driven modeling (using machine learning algorithms) approach to estimate the leakage model function $\hat{f}(., \alpha)$, which is used to estimate the relation between the leakage and the G/M-code. Where $\alpha$ is the tuning parameter for data-driven models. Moreover, the attacker can extract various features and perform feature selections from multiple side-channels to estimate the leakage model function. Due to the presence of a large number of correlated features from multiple side-channel, the attacker is assumed to perform feature reduction (using methods such as principal component analysis, singular value decomposition, etc.) In doing so, they perform automatic selection of relevant features from multiple side-channels for reverse-engineering the G/M-code. Using this function, an attacker can estimate the G/M-code based on the analog emissions from the side-channel. $\hat{g}_k = \hat{f}(l_{(1,k)}, l_{(2,k)}, \ldots, l_{(M,k)})$. Various machine learning algorithms can be used to model the *leakage model function* $\hat{f}_i(., \alpha)$, such that

$$i = \underset{1 \leq i \leq N}{\arg\min} \sum_{k=1}^{K} |g_k - \hat{f}_i(., \alpha)| \tag{3.4}$$

where $N$ is the various *leakage model function* estimated by the attacker. The accuracy of the estimated function depends on the amount of information leaked about G/M-code in the side-channels. We assume that an attacker has unlimited access to the profiling 3D printer during the training phase and can acquire a large number of leakage values corresponding to G/M-codes to build the leakage model functions. The next phase is the attack phase. In this phase, he/she does not have high-level access to the digital process chain of the 3D printer. However, they are able to surreptitiously place sensors to monitor analog emissions with low-level physical access. By low-level access, we mean that they are not able to have digital access to the process chain, however, may be able to have closer non-intrusive physical access to the 3D printer. A malicious insider with low-level access may be an ideal point of contact for an attacker to place the sensors and acquire the analog emissions. Using these emissions, an attacker utilizes the estimated leakage model functions to predict the G/M-codes.

### 3.2.3 Formulation of Data-Driven Leakage-Aware Optimization Problem

In this section, we will formulate the leakage-aware optimization problem for CAM tools used specifically for the fused deposition modeling technique based 3D printers, and show how we can provide feedback to the CAM tool for optimizing various design variables for minimizing the information leakage.

#### Design Variables for Leakage Minimization

The fundamental principle behind reducing the leakage from the side-channels includes a selection of the design variables which can be optimized without substantially affecting the geometry of the 3D objects being printed. This variable may be selected from various stages, either the CAD tools or CAM tools. Since the optimization algorithm is incorporated in the

CAM tool, design variables are selected from the slicing algorithm and tool-path generation algorithm for minimizing the leakage. The exhaustive list of design variables is not presented here, however, the proposed methodology can be scaled when other design variables are added. Here, two design variables $(\gamma, v)$ are proposed.

**Assumption 1:** *For the 3D printer, while printing in each layer (the XY-plane), let* $\boldsymbol{v}$ *be the velocity of the nozzle head (or the extruder from where the filament is deposited) and* $\theta$ *be the angle made by the line segment being printed with the X-axis. Let* $g_i(\theta)$*, be a function that gives the mutual information between the G/M-code (with varying* $\boldsymbol{v}$*) and the analog emissions from the* $i^{th}$ *side-channel when the angle between the line-segment and the X-axis is* $\theta$*. Then for each side-channel there exist angle* $\beta_i$ *such that* $\beta_i = \arg\min_\theta g_i(\theta)$*. Where,* $0 \leq \beta_i \leq 2\pi$*, and* $i = 1, 2, \ldots, M$*.*

**Remark 1:** For each of the side-channels, changing the angle of the line-segment $\theta$ has the corresponding effect of increasing the source of emission. For example, when the angle is $\theta = 0^o$ corresponds to line-segment being printed parallel to X-axis. Hence, a minimum of X and extruder's stepper motor have to be active. When $\theta = 45^o$, line-segment is printed with equal speed $v$ in both X and Y-axes. Hence, a minimum of three stepper motors has to be active (X, Y, and the extruder). Due to the varying number of source of emission added with varying $\theta$, mutual information for different $\theta$ will vary accordingly. Moreover, mutual information will be low if the complex axis movements leakages cannot be distinguished. Hence, there may exist a certain angle for which minimum $I(G; L_i)$ value can be obtained.

In 3D printing, there is a large number of straight line segments, optimizing the $\theta$ for each segment to minimize leakage can affect the convergence of the optimizing algorithm. Rather, Principal Component Analysis (PCA) is used to find the common orientation angle of all the line segments. In the digital process chain, the 3D model is converted to a file with tessellated triangles that describe the geometry. From this file, using the cross product, a vector, $u$,

normal to the plane of the triangular surface is calculated. Next, PCA is performed on the collection of vectors $u$ calculated for all the triangular surfaces. Then, the first principal component which has the highest eigenvalue is extracted. This value represents the most common normal vector of all the line segments. Here, $\gamma$ is defined as the angle of the vector $u'$ which is perpendicular to the first principal component of the vector $u$. This is the first design variable.

**Assumption 2:** *Given the nozzle movement to print a line segment of length $l$ in xy-plane, let $v_x$ and $v_y$ be its velocity in x and y-axis respectively. Where travel feed rate is $v = \sqrt{v_x^2 + v_y^2}$. Then there exists $v_i' \in \mathbb{R}$ such that $v_i' = \arg\min_v g_i(v)$. Where, $g_i(v)$ gives the mutual information between the G/M-code (with varying $\theta$) with speed $v$ and the analog emissions from the $i^{th}$ side-channel.*

**Remark 2:** $v'$ values that will achieve the minimum mutual information in the side-channel lie in the higher travel feed ranges. Considering the acoustic side-channels, first the higher frequency excitation due to faster travel feed-rates will cause a reduction in the amplitude of the vibration as most of the time this excitation force act in opposing the direction of the vibration, and second, the leakage signal will be corrupted quickly by new analog emission from the next G-code. Due to this, the sample of data collected for the G-codes with large travel feed-rate will be less in number and may be contaminated by another G-code leakage signal. Hence, due to the mixture of the leakage signals for different G-code, the mutual information extracted will be low.

**State-of-the-art CAM Tools:** Current slicing algorithms for fused deposition modeling based desktop 3D printers do not consider the information leakage through the side-channels and have tool-path generation that is optimized only for machining efficiency (such as time, material deposition, precision, etc.).

**Optimization Problem Statement**

For minimizing the information leakage from side-channels, based on assumptions 1 and 2, a new leakage-aware algorithm is proposed. The design variables are defined as, $0 \leq \gamma \leq 2\pi$, and $v = \sqrt{v_x^2 + v_y^2}$. Where $v_x \in \mathbb{R}$ , and $v_y \in \mathbb{R}$. For the speed in x and y axis, the two variable bounds are $v_{xmin} \leq v_x \leq v_{xmax}$ and $v_{ymin} \leq v_y \leq v_{ymax}$. Where $v_{xmin}$ and $v_{ymin}$ are the minimum machine specific travel feed-rate in x and y axis respectively, and $v_{xmax}$ and $v_{ymax}$ the maximum machine specific travel feed-rate in x and y axis respectively. Here, a simple constraint $T \leq kT_{orginal}$ is used. Where $T_{original}$ is the printing time of the state-of-the-art slicing and tool-path generation algorithm, and $k \geq 1$ is the user defined constant. The mutual information between the G-code and the leakage signal $I_{\gamma_i}(G; L_i)$ and $I_{v_i}(G; L_i)$ is calculated either at design-time or at run-time by collecting the various analog emissions corresponding to the G/M-codes of various 3D objects. Then, using a non-linear polynomial functions $f_{\gamma_i}(I_{\gamma_i}, \gamma_i)$ and $f_{v_i}(I_{v_i}, v_i)$, the relation between the mutual information and the design variables in different side-channels (acoustic, vibration, power, and electromagnetic) can be estimated. Then, for reducing the mutual information between the analog emission and the design variables, the multi-objective optimization function can be given as follows:

$$(\gamma, v) = \underset{(\gamma,v)}{\arg\min}(f_{\gamma_1}, f_{\gamma_2}, \ldots, f_{\gamma_M}, f_{v_1}, f_{v_2}, \ldots, f_{v_M}) \tag{3.5}$$

Based on the value given by the optimized design variable, slicing and tool-path generation will generate new G-code with minimum information leakage. The algorithm to generate G/M-codes is shown in Algorithm 5. The input to the algorithm are the probability distribution ($\hat{p}_i(\gamma, l_i)$, $\hat{p}_i(v, l_i)$) estimated by collecting the leakage and the G/M-code data while printing the benchmark 3D models, and STL file of the original 3D objects. In line 1, step size for estimating the cost function based on the design variables $\gamma, v$ are defined, along with their range. Then from line 2 to 6, using the probability distribution $\hat{p}_i(\gamma, l_i)$, $\hat{p}_i(v, l_i)$, various mutual information values are calculated for the varying design variables.

**Algorithm 5:** Data-driven leakage-aware G-code generation

**Input:** Empirical or data-driven joint distribution $\hat{p}_i(\gamma, l_i)$, $\hat{p}_i(v, l_i)$
**Input:** Output of CAD Tool (`STL Files`)
**Output:** Modified G-code $g'$

1 Define step size $\triangle_\gamma$, $\triangle_v$ and range $min_\gamma, min_v$ and $max_\gamma, max_v$
2 **for** $i = 1 : M$ **do**
3  **for** *each* $j \in (\gamma, v)$ **do**
4  **for** $k = min_j \ :\triangle_j: \ max_j$ **do**
5  $I_{(j_i,k)} = I_k(G; L_i)$ `// Based on` $\hat{p}_i(\gamma, l_i)$ `// and` $\hat{p}_i(v, l_i)$
6  Estimate Nonlinear function $f_{j_i}$
7 Optimize $\arg\min_{(\gamma,v)}(f_{\gamma_1}, f_{\gamma_2}, \ldots, f_{\gamma_M}, f_{v_1}, f_{v_2}, \ldots, f_{v_M})$
8 $g'$ =SliceandToolPathGeneration($\gamma, v$, `STL File`) **return** $g'$

In line 6, polynomial function is used to estimate the relation between the design variables and the mutual information calculated in line 5. Then based on the description of the problems statement, mixed multi-objective non linear integer programming is used to optimize the design variables. In line 8, the modified design variables are passed to the slicing and tool-path generation function to generate a G-code with minimum leakage, which is finally returned in line 9.

### 3.2.4 Success Rate of the Adversary

Mutual information provides an idea about how much information is available to an attacker for the exploitation [89]. However, it does not provide information about the capability of an attacker. A strong attacker may be able to exploit little information for a successful attack, whereas, a weak attacker may not be able to exploit the information available for the attack. Hence, in addition to the mutual information, the success rate of an attacker helps in demonstrating how side-channels can be leveraged for breaching the *confidentiality* of the 3D printer. Hence, compared to the work in [34], in this work, the success rate based on the adversary model presented in Section 3.2.1 is also analyzed. It is assumed that an attacker has unlimited access to a replica of the manufacturing system, and by using state-of-the-art learning algorithms he/she is able to generate data-driven leakage model of the system.

3D printer instructions can be divided into two types [90]. The first type is called G-codes and they are responsible for the movement of the nozzle in different axis. The second type is called M-codes, and they are responsible for controlling the parameters of the machine beside motion. An example of a G type instruction is a command which results moving the nozzle of a 3D printer for a certain distance, with a certain angle and speed. On the other hand, an example of the M type instructions is an instruction which sets the acceleration rate for the stepper motor. Here, for the sake of simplicity, it is assumed that the number of M type instructions are limited, and do not impact the geometry of the 3D objects [90], which is the case in 3D printers. Hence, the focus is given only to G type instructions for measuring the success rate. The G type instructions or the G-codes consists of instruction such as *i) movement axis, ii) distance moved in each axis, iii) speed of motion in each axis, and iv) extrusion amount.* The extrusion amount is calibrated to maintain the optimal print quality and is fixed in most of the CAM tools. It does not affect the geometry of the object being printed. The *Z-axes* movement occurs every layer with constant height and can be distinguished or inferred with ease. In state-of-the-art slicer used in the CAM tools, such as Cura [91], speed does not change dynamically. Rather, most of the time, it has constant predefined values for each stage (*first layer, outer boundary, inner filling and traveling without printing*) of printing. Since speed variation in each of the stages is drastically different, they are easier to classify. Hence, the most challenging information that an attacker may require to extract from a G-code is the geometry of the object being printed in each *XY-plane*. This information can be abstracted in terms of angle and length of the line segment being printed by each G-code.

Based on these assumptions, the G-code instruction is reduced to $E_{(\theta,d)}$, where $0 < \theta < 360$ is the angle between the movement direction of a line segment and the $X$ axis, $d$ is the travel length. Then for calculating the success rate of an attacker, only measure of how successfully he or she is able to reconstruct $\theta$ and $d$ for each line segment is required. Let us represent the sequence of G-codes as $\{E_1, E_2, \ldots, E_i\}$, with $E_i = E_{(\theta_i,d_i)}$. Let $L_i$ be the

leakage corresponding to $E_i$. Then an attacker will utilize the function $\hat{f}(.,\alpha)$ estimated using Equation 3.4 to infer about $E_i$ based on $L_i$. Estimated $\hat{E}_i$ consists of $\hat{\theta}$ and $\hat{d}$.

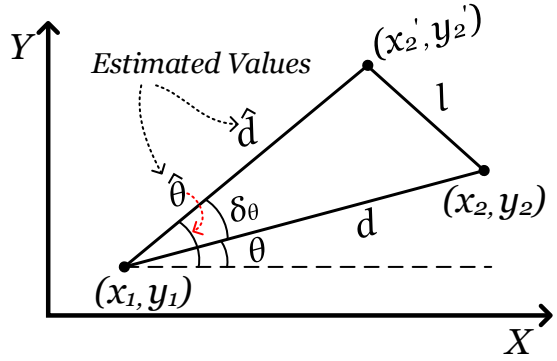



Figure 3.3: Success rate formulation for each G-code instruction.

In our attack model, attacker estimate $\hat{\theta}, \hat{d}$ independently. Hence, there are two functions $\hat{\theta} = \hat{f}_\theta(.,\alpha)$ and $\hat{d} = \hat{f}_d(.,\alpha)$. Now, the success rate is formulated in terms of accuracy in reconstruction of each of the line segment using $\hat{\theta}$ and $\hat{d}$.

For doing so, let $(x_1, y_1)$ and $(x_2, y_2)$ be a line segment a G-code instruction prints on a **XY-plane**. As explained earlier, this segment can be explained in terms of polar coordinates using $\theta$ and $d$. Now, let the estimated polar coordinate value be $\hat{\theta}$ and $\hat{d}$ as shown in Figure 3.3. Then, the success rate is defined in terms of a parameter $l$, which can be given as follows:

$$l = \sqrt{d^2 - 2dd'cos(\delta_\theta) + \hat{d}^2} \tag{3.6}$$

Then the success rate is measure of how accurate is $l$, given as follows:

$$Success\ Rate\ (SR) = \frac{1}{n}\sum_{i=1}^{n} S_l \tag{3.7}$$

Where $S_l = 1$ when $|\hat{l}_i - l_i| < e_l$ and 0 otherwise, and n is the total number of G-code instructions being reconstructed. To prove that success rate encompasses accuracy estimation of both $\theta$ and $d$, three cases are considered. First, let angle estimation be 100% accurate.

In that scenario, $\delta_\theta = 0$, then $l = \pm(d - \hat{d})$ from Equation 3.6, which shows that $l$ still depends on the estimation of $d$. In second case let us assume that estimation of $d$ is 100% accurate, then $d = \hat{d}$. Then Equation 3.6 reduces to $l = \pm d\sqrt{2 - 2cos(\delta_\theta)}$. It shows that in such scenario $l$ still depends on estimated value of $\theta$, as $\delta_\theta = \hat{\theta} - \theta$. In the final case, let estimation of $\theta$ and $d$ be 100% accurate, then $\delta_\theta = 0$ and $\hat{d} = d$. Then from Equation 3.6, $l = \sqrt{d^2 - 2ddcos(0) + d^2} = \sqrt{d^2 - 2d^2 + d^2} = 0$, making success rate 100% accurate as well. Hence, this metric will give the total success rate of the G-code instruction in terms of $\hat{\theta}$ and $\hat{d}$.

## 3.3 Experimental Results

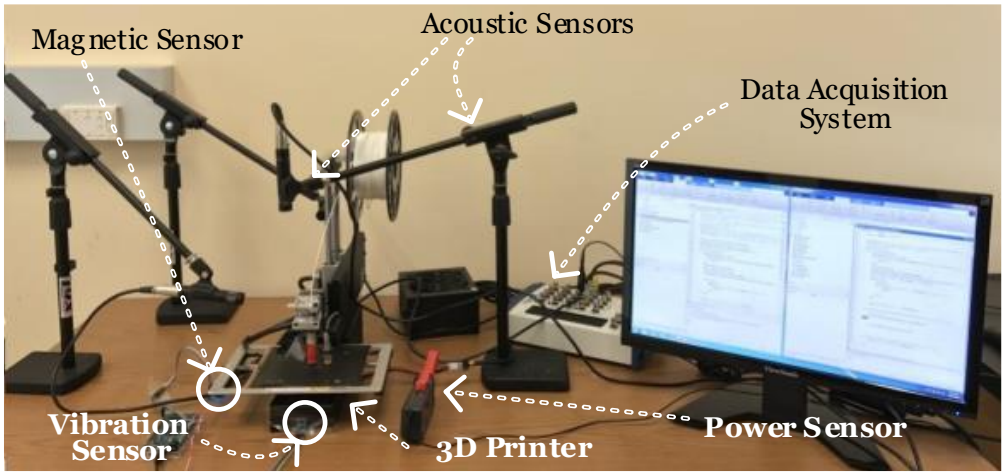



Figure 3.4: Experimental setup with multiple sensors

The experimental setup for the leakage-aware computer aided manufacturing tool is shown in Figure 3.4. The setup consists of a fused deposition modeling based desktop 3D printer [92]. Three AT2021 cardioid condenser audio sensors [93] are placed parallel to $X$, $Y$ and $Z$-*axis*, respectively and treated as individual channels to consider sound emanation in multiple direction. The vibration sensor (Adafruit triple-axis accelerometer [94]) is placed at the static part of the 3D printer, measuring vibration in all three axes. A current clamp [95] is used to non-intrusively measure the current being passed to the main circuit board of the printer

from a power supply. A Honeywell's magnetometer (HMC5883L [96]) is used to measure the magnetic fluctuation around the 3D printer. Beside the current clamp, most of other sensors (audio, vibration, magnetic, etc.) are all present in modern smartphones, and have been used in the attack model presented in [77] for breaching the *confidentiality*.

Since four side-channels are used, here, $M$=4. Audio sensors placed in different axis are fused together and treated as a single side-channel. From the raw signals collected from these sensors, power spectral density in the frequency range of 60 $Hz$ to 10 $kHz$ is calculated, principal components of the corresponding power spectral density are analyzed, and the mutual information between these principal components and the design variables are measured. In our experiment, we have considered only design-time leakage quantification and mutual information calculation. Which means that the mutual information for various design variables are only calculated once and are not updated.

### 3.3.1 Mutual Information

In this section, analysis between the mutual information and the design variables is presented. This will demonstrate how the non-linear function in line 6 of Algorithm 5 can be estimated.

**Design Variable - $\gamma$**

$\gamma$ is varied from 0° to 90° with the step size $\triangle_\gamma = 10°$. For estimating the joint probability function $\hat{p}(\gamma, l_i)$, $\gamma$ is distributed uniformly, and the corresponding analog emissions from the side-channels are collected. This joint probability function $\hat{p}(\gamma, l_i)$ is then used in calculating the mutual information using Equation 3.3.In Figure 3.5, mutual information between the first principle component of the power spectral density values of the four analog emissions and the design variable $\gamma$ are presented. The total entropy of the design variable ($\gamma$) is

$log_2(11) \cong 3.4594$ bits. It can be observed that for the design variable $\gamma$ vibration analog emission give more information leakage compared to the other emissions. For motion in XY-plane, since $v$ is constant for printing a single line-segment, varying $\gamma$ changes the $v_x$ and $v_y$ instead (as $v = \sqrt{v_x^2 + v_y^2}$). This means varying $\gamma$ results in actuation of stepper

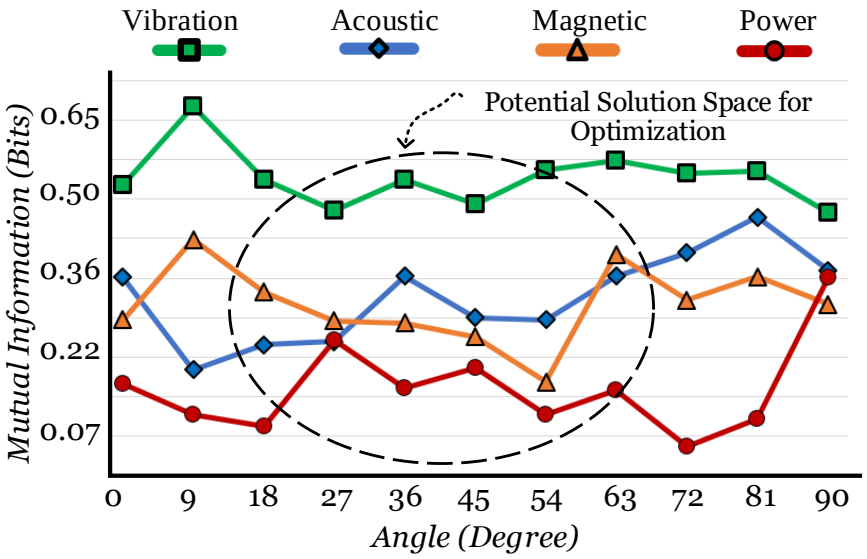



Figure 3.5: Mutual information between angle ($\gamma$) and leakage

motor moving in X and Y-axis with the different angular speed. For vibration side-channel, change of angular speed of stepper motors will result in a change of acceleration measured in X and Y-axis, immediately. The acoustic side-channel would acquire this variation in the acceleration of sound pressure only when the surface of the 3D printer has been excited. This causes the vibration side-channel to pick more information about $\gamma$ variation than acoustic side-channel. The low-field magnetic sensors used in the experiment measures how the mechanical components of the 3D printer affect the strength and directional of the earth's magnetic fields. Variation in $\gamma$ causes the metallic part of the FDM 3D printer to cut the earth's magnetic field at a varying rate. Hence, the variation of $\gamma$ may be inferred from the magnetic side-channel. The power side-channel have lowest mutual information with a variation of $\gamma$. The power signal is one dimensional compared to another emission which measure the signal variation in more than one dimension (X, Y or Z axis). Hence, it is intuitive that it leaks less information compared to the other side-channels. From Figure

3.5, it may also be observed that single axis movement (when $\gamma = 0$ and $\gamma = 90$) have more mutual information and thus leak more information compared to multiple axis movements (as there is emission from multiple components of the 3D printer).

**Design Variable -** $v$

The travel feedrate ($v$) is varied from 700 mm/min to 3300 mm/min (which is the range for the given 3D printer) with the step size $\triangle_v = 200$ mm/min, to demonstrate the relation between the design variable $v$ and the mutual information. Similar to $\gamma$, the joint probability function $\hat{p}(v, l_i)$ is estimate by collecting the analog emissions for the uniformly distributed travel feedrate values. The total entropy of the design variable ($v$) is $log_2(14) \cong 3.8074$, as

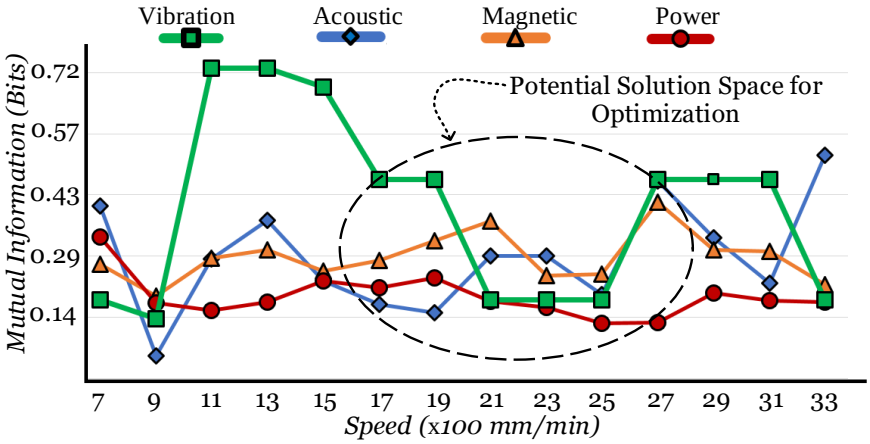



Figure 3.6: Mutual information between speed ($v$) and leakage

there are 14 speed values. In Figure 3.6, the mutual information between the first principal component (with the highest eigenvalue) of the power spectral density and the varying travel feed rate for vibration, acoustic, power, and magnetic emissions are presented. When the travel feed rate is increased in the G/M-code, the stepper motor driver increases the rate of current passing through the electromagnetic stator core of the stepper motor. The presence of a permanent rotor and the electromagnetic stator with a varying rate of constant current supply makes stepper motor capable of producing audible sound. Moreover, the

frame in contact with the stepper motor helps the vibration to be conducted throughout the mechanical frame of the 3D printer, and allow the vibration side-channel to pick subtle variation even before the sound is emanated in the air. Moreover, the varying rate of current supply fluctuates the power consumption in each of the stepper motors used for actuating the G/M-code. The changing current rate affects the magnetic field created in the stator, however, the sensor used in our experiment is not able to capture these small variations of the magnetic field. However, the varying travel feed rate changes the rate of movement of the mechanical structure, which affects the magnetic field of the earth. This can be captured by the magnetic sensors. From Figure 3.6, it can be observed that, for speed variation, vibration side-channel reveals more information compared to other side-channels. As explained earlier, the vibration side-channel is able to capture the minute variation in vibration caused by the variation of the travel-feed rate before the acoustic side-channel, resulting in it having higher mutual information than the acoustic side-channel. Moreover, the mutual information is higher for the lower speeds, lower for the medium range speeds and again higher for the faster speeds. This may be due to the fact that the 3D printer has certain resonant frequencies caused by the cumulative movement and vibration of the individual components, and the speed ranges causing the vibration closer to resonant or harmonics of the resonant frequencies leak more information. The power side-channel have lower mutual information compared to others, this may be due to the fact that the base plate heater and the heater in the nozzle of the 3D printer consume larger current, and variation of stepper motor current fluctuation has lower SNR causing lower mutual information between the power leakage values and the G/M-code.

The optimization algorithm utilizes the non-linear relation estimated in Figure 3.5 and 3.6 to minimize the mutual information between the analog emissions and the cyber-domain data to effectively reduce the information leakage.

### 3.3.2 Test with Benchmark 3D Models

The CAD models of the benchmark models are selected from [97] that cover wide-variation in G/M-codes. The selected benchmark 3D models are shown in Figure 3.7. Some of these objects (numbered *1*, *2*, *3*, *4*, *11*, and *8*) are selected because they are used for calibrating and tuning the 3D printer where as other (numbered *5*, *6*, *7*, *9*, *10*, *12*, and *13*) are selected due to the variation in complexity of the design. For each of these benchmark model, mutual information between the G-codes and each of the side-channels is calculated before and after changing the design variables $(\gamma, v)$. Figure 3.8, 3.9, 3.10, and 3.11 show the results of mutual information calculation for acoustic, power, magnetic, and vibration side-channels, respectively.

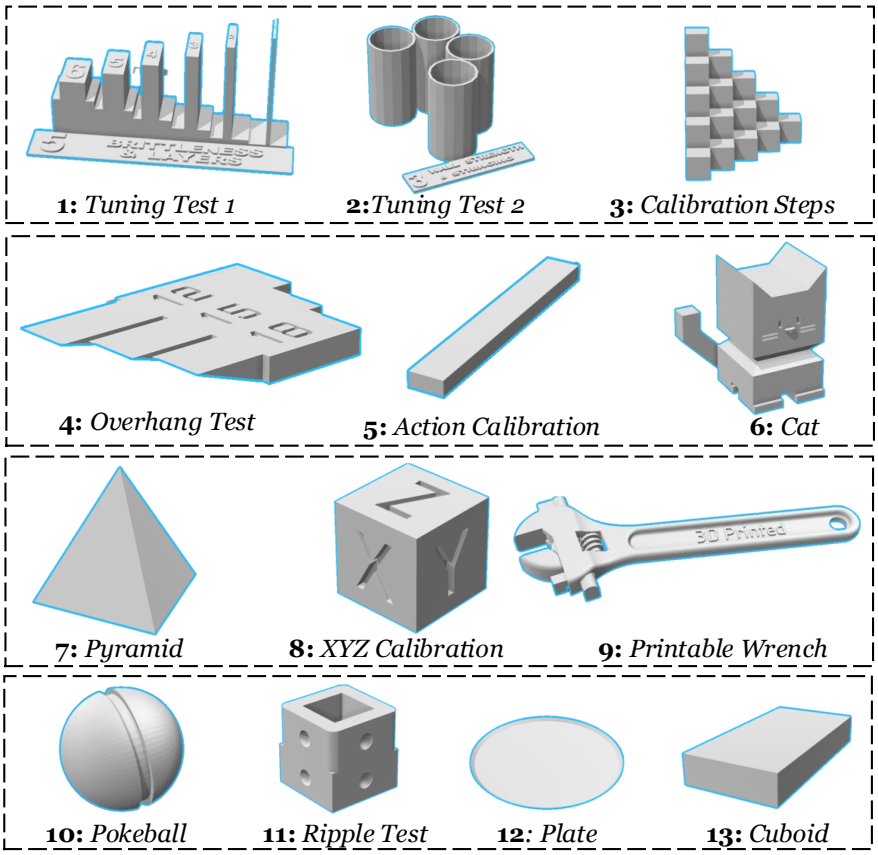
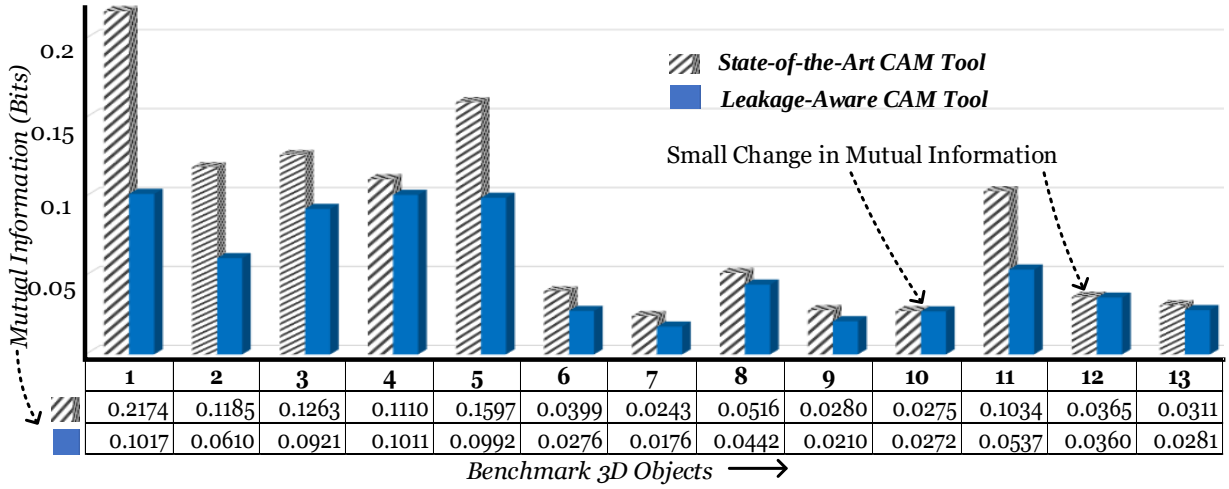



Figure 3.7: Benchmark 3D objects for testing the leakage-aware computer aided manufacturing tool

For acoustic side-channel, the average drop of mutual information by **24.94%** is obtained across all the benchmark models (see Figure 3.8).

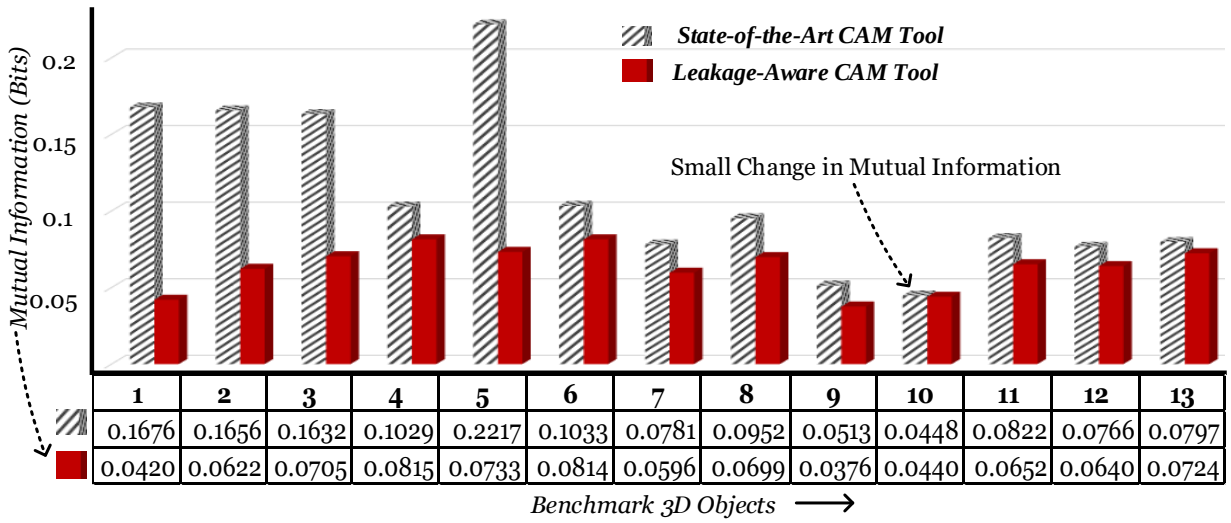



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.2174 | 0.1185 | 0.1263 | 0.1110 | 0.1597 | 0.0399 | 0.0243 | 0.0516 | 0.0280 | 0.0275 | 0.1034 | 0.0365 | 0.0311 |
| | 0.1017 | 0.0610 | 0.0921 | 0.1011 | 0.0992 | 0.0276 | 0.0176 | 0.0442 | 0.0210 | 0.0272 | 0.0537 | 0.0360 | 0.0281 |

Figure 3.8: Mutual information between G-codes and acoustic side-channel

However, for object number 10 and 12 (see Figure 3.7) there is only a small change in mutual information rather than large reduction when the leakage-aware slicing and tool-path generation algorithm is used. This may be due to the fact that objects 10 and 12 consists of geometric structures (spherical and circular) that do not benefit from design variable $\gamma$ (as the change in $\gamma$ does not introduce variation in the orientation of the spherical and the circular objects). However, real-world objects are not just spherical or circular, and the mutual information will decrease when leakage-aware CAM tool is used.

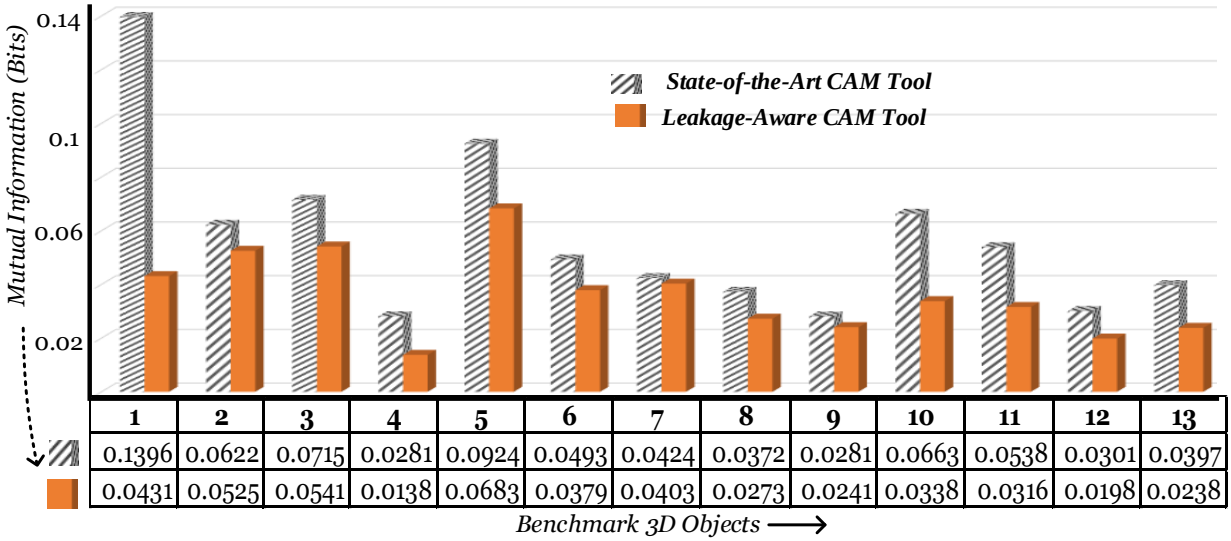
For power side-channel, the average drop of **32.91%** in mutual information is obtained across all the benchmark models. As shown in Figure 3.9, the mutual information dropped for all the benchmark models except object 10 (only a small change was observed). Unlike other side-channels, power side-channel measures data only in one dimensional (unlike acoustic, vibration, and magnetic that measure fluctuation in all X, Y, and Z-axis). The variation in the instantaneous power of the 3D printer occurs due to specific activation sequence of various components. For Spherical geometry like object 10, due to symmetric property, the variation in $\gamma$ does not result in a change in the sequence of activation drastically.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1676 | 0.1656 | 0.1632 | 0.1029 | 0.2217 | 0.1033 | 0.0781 | 0.0952 | 0.0513 | 0.0448 | 0.0822 | 0.0766 | 0.0797 |
| | 0.0420 | 0.0622 | 0.0705 | 0.0815 | 0.0733 | 0.0814 | 0.0596 | 0.0699 | 0.0376 | 0.0440 | 0.0652 | 0.0640 | 0.0724 |

Figure 3.9: Mutual information between G-codes and power side-channel

For magnetic side-channel, the average drop of **32.29%** in mutual information is obtained across all the benchmark models (see Figure 3.10). For all side-channels, the objects that benefited the most in reducing the mutual information were object 1 and object 5 (see Figure 3.7). This is due to the fact that these objects are stretched in either $X$ or $Y$ axis and hence changing the $\gamma$ drastically changes its orientation, thus affecting the analog emissions.

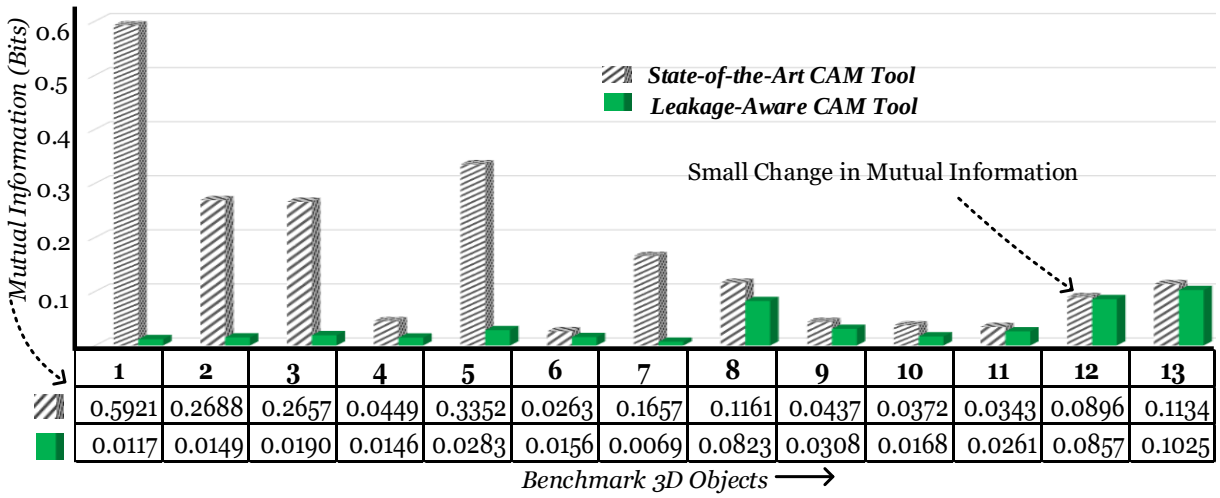



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.1396 | 0.0622 | 0.0715 | 0.0281 | 0.0924 | 0.0493 | 0.0424 | 0.0372 | 0.0281 | 0.0663 | 0.0538 | 0.0301 | 0.0397 |
| | 0.0431 | 0.0525 | 0.0541 | 0.0138 | 0.0683 | 0.0379 | 0.0403 | 0.0273 | 0.0241 | 0.0338 | 0.0316 | 0.0198 | 0.0238 |

Figure 3.10: Mutual information between G-codes and magnetic side-channel

For vibration side-channel, the average drop of **55.65%** in mutual information is obtained across all the benchmark models (see Figure 3.11). This drop in mutual information is highest among all the side-channels. This result corroborates the analysis presented in Section 3.3.1.

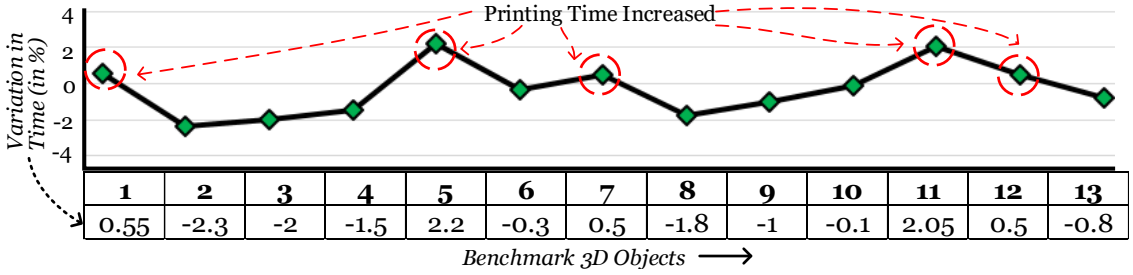
Optimizing the design variables $\gamma$ and $v$ affects the vibration side-channel the most. Provided the fact, there are more design variables and machine specific process parameters that may be tuned to reduce the mutual information, design space exploration of such variables may be performed to drastically reduce the mutual information from all the side-channels. Similar to acoustic side-channel, it also has a relatively small change in mutual information for object 12.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.5921 | 0.2688 | 0.2657 | 0.0449 | 0.3352 | 0.0263 | 0.1657 | 0.1161 | 0.0437 | 0.0372 | 0.0343 | 0.0896 | 0.1134 |
| | 0.0117 | 0.0149 | 0.0190 | 0.0146 | 0.0283 | 0.0156 | 0.0069 | 0.0823 | 0.0308 | 0.0168 | 0.0261 | 0.0857 | 0.1025 |

Figure 3.11: Mutual information between G-codes and vibration side-channel

Figure 3.12 shows the timing variation between the state-of-the-art CAM tool and the leakage-aware CAM tool expressed in terms of percentage increase or decrease in the printing time. It can be seen that for the given benchmark 3D objects, the maximum increase in the printing time was **2.2%**.




| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.55 | -2.3 | -2 | -1.5 | 2.2 | -0.3 | 0.5 | -1.8 | -1 | -0.1 | 2.05 | 0.5 | -0.8 |

Figure 3.12: Variation in printing time of leakage-aware CAM tool compared to the state-of-the-art

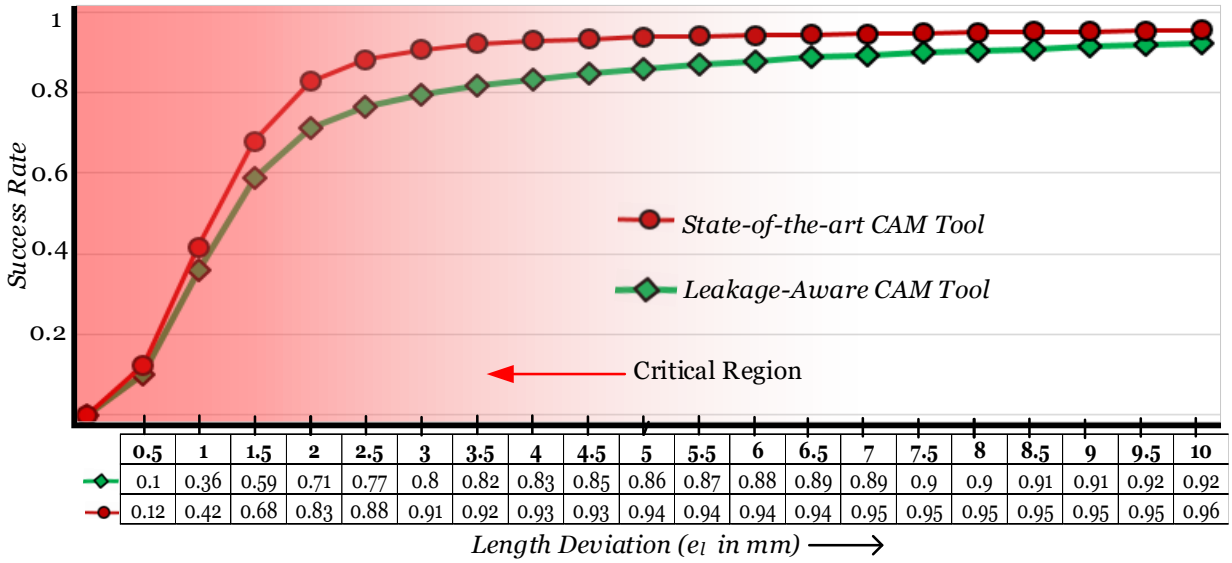
## 3.4 Case Study with an Attack Model

The mutual information provides the system designers about the amount of information that be deduced about the G/M-code from the analog emissions. However, the success rate in reconstructing the G/M-code also highly depends on the capability of an attacker. A strong attacker (with domain knowledge, large computing capability, and some high-level access to the digital process chain of the target 3D printer) may be able to achieve higher success rate with low mutual information. In our work, we do not provide an exhaustive analysis of the success rate based on the various capability of an attacker. Rather, we choose the attack model presented Section 3.2.2, which has also been used in previous works [77, 68].

### 3.4.1 Success Rate Calculation

In the attack model, an attacker will utilize leakage from all the size-channels to infer about various G-codes using a function $\hat{f}(.,\alpha)$. Moreover, each parameters of the G-code such as $\theta$ and $v$ is estimated using functions $\hat{\theta}_k = \hat{f}_\theta(l_{(1,k)}, l_{(2,k)}, l_{(3,k)}, l_{(4,k)})$ and $\hat{d}_k = \hat{f}_d(l_{(1,k)}, l_{(2,k)}, l_{(3,k)}, l_{(4,k)})$. Since G/M-code consists of various information about the line-segment, it becomes intuitive to have multiple functions estimating these parameters, and later combining them together to reconstruct the line-segment. As a test case study, a machine learning algorithm called *Random Forest* (previously used in [68]) is used to estimate the functions $\hat{f}_\theta(.,\alpha)$ and $\hat{f}_d(.,\alpha)$, and the success rate is calculated using Equation 3.7 for various values of $e_l$ ranging from 0 to 10 $mm$ with step size of 0.5 $mm$. As mentioned in Section 3.2.4, Equation 3.7, measures the average success rate in reconstructing the line-segment defined by the G/M-code. The error threshold $e_l$ corresponds to the combined error in predicting the length $d$ of the line segment and the angle $\theta$ made with the X-axis. For feature, an attacker extracts the power spectral density values from all the side-channels. These features are then fused together and passed to the random forest algorithms for esti-

mating the functions. By combining the features from all the side-channels, the estimated function effectively selects the channel that gives more information about the $d$ and $\theta$. During the training phase, 75% of the benchmark models are used to gather analog emissions from the side-channel, estimate the functions, and the rest of the benchmark model's G-code is predicted using these functions. Since an attacker has high-level access to the 3D printer, he/she may be able to acquire unlimited training data. However, for feasibility and variation in training data, we have selected the benchmark models and divided it into test and training to mimic the two phases of the attack model. The average success rate for the reconstruction




| | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 7.5 | 8 | 8.5 | 9 | 9.5 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ◆ | 0.1 | 0.36 | 0.59 | 0.71 | 0.77 | 0.8 | 0.82 | 0.83 | 0.85 | 0.86 | 0.87 | 0.88 | 0.89 | 0.89 | 0.9 | 0.9 | 0.91 | 0.91 | 0.92 | 0.92 |
| ● | 0.12 | 0.42 | 0.68 | 0.83 | 0.88 | 0.91 | 0.92 | 0.93 | 0.93 | 0.94 | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.96 |

Figure 3.13: Average success rate for G-code reconstruction with varying $e_l$

of the line-segments described by the G/M-code for varying ranges of $e_l$ is shown in Figure 3.13. It can be observed that compared to the state-of-the-art CAM tool the success rate of an attacker in reconstructing the line-segment is less in the leakage-aware CAM tool. The success rate reduction across the ranges of $e_l = 0$ *to* 10 mm is 8.74%. Moreover, the reduction in success rate is more than 10%, when $0\ mm < e_l \leq 5\ mm$. This is the critical region for $e_l$, as an attacker may acquire accurate geometry information easily without further post-processing in this range. The success rate calculated here is in reconstructing the individual line segments. This still does not explain the accuracy in overall 3D object reconstruction. For reconstructing the 3D object, an attacker will have to use post-processing to reconstruct

the 3D object layer-by-layer, using the estimated line-segments. This post-processing depends on the capability of an attacker and is out of the scope of this chapter. However, we argue that with a higher success rate in each of the line segment, an attacker will have to use less post-processing in reconstructing the 3D object, hence reduction of the success rate for each the line-segments still helps in obfuscating the intellectual property of the object.

### 3.4.2 Test Case with Reconstruction

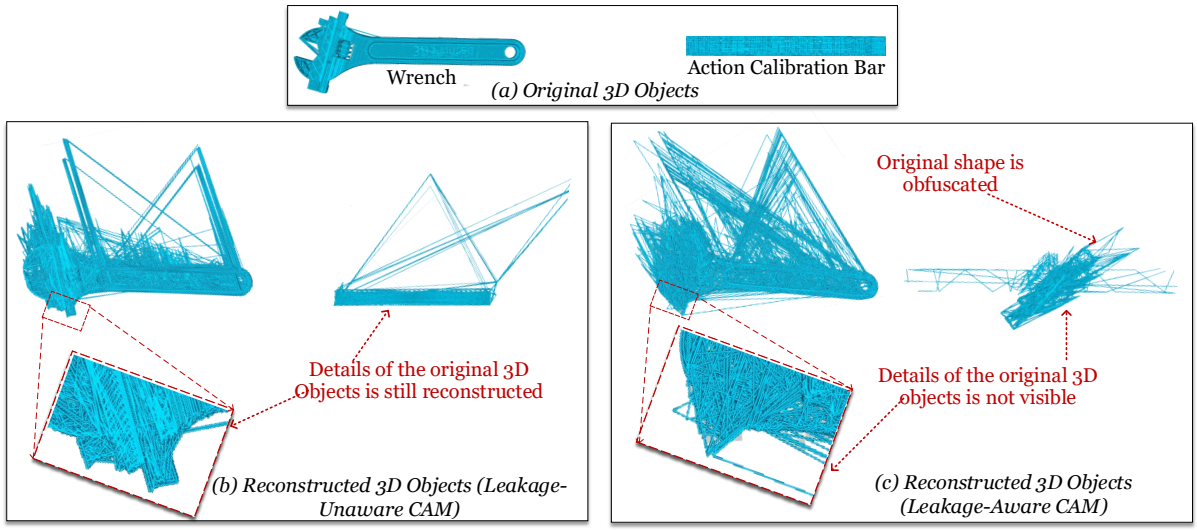



Figure 3.14: Reconstructed and traced G-codes of test case objects

In this section, simple post-processing is applied to visualize the effect of a decrease in mutual information and success rate on the reconstruction of the 3D objects (after the G-code prediction). The predicted line-segments described by G/M-code from the previous section is used for the 3D object reconstruction. This procedure is carried out for the G/M-codes produced by the state-of-the-art CAM tool (Cura), and the leakage-aware CAM tool. The two test objects chosen have relatively simple (for action calibration or bar) and complex (for a printable wrench) structures. The result of the reconstruction of the test objects for leakage-aware and leakage-unaware CAM tool is shown in Figure 3.14. The post-processing used involved using the same M-codes generated for the training objects on the

test objects as well. During a real attack, an attacker is assumed to have some information about the M-codes of the target 3D printer. As M-codes are machine specific parameters such as extruder and bed temperature, alignment of the printing bed, etc. These parameters are normally set to an optimal constant value for a given 3D printer. This makes sure the quality is maintained. And attacker, during the training phase, has access to the digital process chain of the profiling 3D printer, which is assumed to be similar in build to the target 3D printer. Hence, he/she can use the machine specific information as a post-processing step for reconstruction of the 3D objects.

From Figure 3.14, it can be observed that the objects reconstructed from emission captured from the side-channels, when state-of-the-art CAM tool is used, consists of a relatively higher amount of details. On the other hand, the geometry details of the objects are not reconstructed when the leakage-aware CAM tool is used. Without using the leakage-aware CAM tool, it can be observed that in *Printable Wrench*, the details of the outline can still be reconstructed. The obfuscation correlates to the amount of decrease in the mutual information and the success rate. For instance, the action calibration bar (object 5), which has a larger drop in mutual information (compared to object 9), caused the reconstruction to be distorted (making it look nothing like the original shape). This can limit the attacker's capability to steal the valuable intellectual property inherent in the detailed geometry design, thus preventing intellectual property theft.

## 3.5 Discussion

**Positioning of the Sensors:** In this work, the sensors position, number, and orientation have not been explored. An effective attacker may surreptitiously place multiple sensors for data acquisition which may aid in reconstruction. Moreover, to explore the position of sensors to acquire better emission signals may aid in improving the performance of leakage-aware

CAM tool as well. Currently, the sensors have been placed in a location that is non-intrusive to the printing process, such as the stationary base of the 3D printer for vibration sensor, stable ground for magnetic sensors, and X, Y, and Z-axes for acoustic sensors.

**3D Printer Variation:** Since the attack model relies on analog emissions that behave as side-channels, it highly depends on the mechanical structure of the 3D printer. In literature, we have found that the researcher has tested the effectiveness of the attack model using fused deposition modeling based 3D printers, especially the system with Cartesian movement. Since these printers have similar mechanical build, the same approach of the estimating the G/M-code may work, however, this needs to be tested for 3D printers with other mechanical build types (such as polar, delta, etc.) and technologies (such as Stereolithography, Selective Laser Sintering. Digital Light Processing, etc.)

**Optimization Constraints and Design Variables:** The design variables such as orientation $\gamma$ and speed $v$ do not increase the printing time of the 3D model. However, the test case reconstruction shows that some geometry information is still reconstructed even after the leakage-aware CAM tool is used. Hence, more design variables need to be explored in the additive manufacturing domain. These design variables, however, cannot increase various optimization constraints or affect the quality of the 3D objects. Moreover, the same design variables may be able to decrease the mutual information drastically provided the optimization constraints are relaxed. In this way, the trade-off between *confidentiality* and the optimization constraints (such as printing time) may be handled. Furthermore, we demonstrated the possibility of a reduction in mutual information and the success-rate in the reconstruction of the 3D object using just two design variables. We acknowledge, that 10% reduction in success rate may not be able to completely hide the geometry (specifically the larger outlines in figure 3.14) of the 3D objects. However, provided the fact that there are many design variables (both in object specific G-codes and machine specific M-codes), our methodology can easily be extended to achieve a higher level of obfuscation in geometry.

**Formulation of success rate for IP theft:** We presented a success rate based on the reconstruction of the individual G/M-code line-segments. However, for the success rate in stealing the intellectual property of the 3D object, a proper metric incorporating the capability of an attacker is necessary. Nonetheless, the propose leakage-aware CAM tool helps in reducing the mutual information and increasing the cost of reconstruction for an attacker.

**Other means of reduction of information leakage:** This chapter lacks a comparison of the results with other solutions. However, to the best of knowledge this is the first approach in reducing the information leakage from the side-channels in a cyber-physical additive manufacturing system, without additional system modification, and thus no additional cost. However, intrusive system modification (like the addition of noise generators) may achieve better results with added cost.

## 3.6 Summary

In this chapter, we present a novel data-driven methodology to improve the *confidentiality* of the cyber-physical manufacturing systems. We achieve this by incorporating leakage-aware computer-aided manufacturing tools such as *slicing algorithm* and *tool-path generation algorithm*. This cross-domain security solution takes the feedback from the physical domain, and optimizes the cyber-domain design variables (*orientation* and *speed*) to minimize the mutual information between the cyber-domain G/M-codes and the physical domain analog emissions. For various benchmark 3D models, our solution obtains an average mutual information reduction of **24.94%** in acoustic side-channel, **32.91%** in power side-channel, **32.29%** in magnetic side-channel, and **55.65%** in vibration side-channel of the fused-deposition modeling based Cartesian 3D printer. To validate the proposed methodology, a case study with an attack model is used to calculate the success rate. For the various threshold of G/M-code

reconstruction, the average drop in success rate obtained is **8.74%**. With this work, the capability of a leakage-aware cyber-domain computer aided manufacturing tool for maintaining the *confidentiality* of the system without physical system modification is highlighted.

# Chapter 4

# Part I: Data-Driven Defense: Kinetic Cyber Attack Detection

## 4.1 Introduction and Related Work

In chapter 3 we discussed a data-driven modeling approach to reduce the information leakage from the side-channels. In this chapter, we will present a data-driven modeling approach for providing defense against a new kind of attacks highly applicable to cyber-physical systems. This kind of attacks is known as kinetic cyber-attacks. These type of attacks originate from the cyber-domain but cause physical damage, injury, or even death [98]. In CPS, effects of kinetic cyber-attacks have been recently highlighted by incidents such as the Stuxnet malware [99], Maroochy water breach [100], German steel mill cyber-attack [101], and security breaches in automobiles [102]. In cyber-physical additive manufacturing systems, kinetic cyber-attack can find its way through the digital process chain to introduce various inconspicuous flaws in the 3D objects. If these objects are critical for the system, they can compromise the structural integrity and pose a severe safety risk. For example, an incon-

spicuous void (less than 1 $mm$ in dimension) placed in the 3D design of an American Society for Testing and Materials standard D638-10 tensile test specimen, reduced its mechanical strength to carry the load by 14% [103].

The security concerns in CPS are not new [104, 105], and in fact, various attack detection methods have been designed for the identification and detection of attacks [106]. However, cyber-physical additive manufacturing system has not received much attention for the research in attack detection methods. In [103], authors present the potential attack vectors for cyber-physical additive manufacturing systems' digital process chain and recommend securing the process chain by incorporating software checks, hashing, and process monitoring through side-channel. In [107], authors present a signature-based attack detection method leveraging the concept of integrated circuit Trojan detection from side-channel analysis and system health monitoring. However, it should be noted that signature-based technique requires acquiring the signature of a baseline structure every time it is created, which is counter-intuitive for rapid prototyping capability of the cyber-physical additive manufacturing system. In [108], authors list the threat surface and describe its effects on the manufacturing parameters. In the cyber-physical additive manufacturing system, any variation made in the design of the 3D object is manifested physically. Moreover, during the printing stage, the machine itself unintentionally emits *analog emissions* from the side-channels (such as acoustics, electromagnetic radiation, power, etc.). Thus, we propose a novel Kinetic Cyber-Attack Detection (KCAD) method that uses statistical data-driven modeling of the cyber-physical additive manufacturing system to detect the anomalous *analog emission* which can arise as a result of potential *zero-day kinetic cyber-attacks*.

### 4.1.1 Motivation

The fundamental motivation for KCAD method comes from the fact that in CPS, the information flow in the cyber-domain has at least one corresponding signal flow in the physical domain [109]. These signals in the physical domain actuate the physical processes, and this actuation converts energy from one form to the another. This phenomenon allows us to monitor the unintentionally leaked *analog emissions* which have high *mutual information* with the corresponding control signals. *Analog emissions* have been used in system health monitoring and prognostics to infer about the current state of the system, and also for quality control in manufacturing [110, 111]. However, traditional quality control system focus in measuring the key quality characteristics, and kinetic cyber-attack on various features may not be detected by such systems [107]. On the other hand these *analog emissions* are also used to breach the confidentiality of the system [112]. However, incorporating statistical method, the acquired *analog emissions* from the side-channel corresponding to the *control signals* can be used to model the behavior of the system. And this model may be used for detecting the intrusion in the system [113].

### 4.1.2 Problem and Research Challenges

The current challenge for securing the CPS is understanding its unique properties and vulnerabilities and preventing the possibility of an attack [114]. However, *zero-day vulnerabilities* of the system are hard to detect during design time. In such scenarios, detection of such attacks is the best possible defense. After detection, we can take measures such as halting the system, perform corrective actions, etc., to mitigate the effects of the attack. Details of the mitigation methods are out of the scope of this chapter. In the cyber-physical additive manufacturing system, the problem for designing a *zero-day kinetic cyber-attack* detection method poses the following key challenges:

1. Detecting the intrusion/attack that can occur at various points of the digital process chain of cyber-physical additive manufacturing system, and affect the dynamics of the system.

2. Making it non-intrusive so that it can be used in legacy cyber-physical additive manufacturing system systems.

3. Complementing the detection method with the physical and process knowledge from cyber-physical additive manufacturing system.

### 4.1.3 Our Novel Contributions

To address the above mentioned challenges, we propose a novel attack detection method for detecting the effects of possible *kinetic attacks* in the digital process chain of the additive manufacturing that employs:

1. **Modeling of an Adversary (Section 4.2)** to understand the various attack points in the digital process chain for effective implementation of the attack detection method.

2. **Statistical Estimation (Section 4.3)** to perform data-driven modeling of the behavior of the cyber-physical additive manufacturing system by analyzing the relationship between the *analog emissions* and the control signals.

3. **Analysis of Analog Emission (Section 4.3.3)** to use it as a parameter from the side-channel for estimating the relationship between the *analog emissions* and the control parameters using **mutual information** as the relation measurement metric.

## 4.2 Kinetic-Cyber Attack Adversary Model

In the cyber-physical additive manufacturing system, information (3D design specification) flows through the digital process chain and is finally converted as the control signals in the machine. In the adversary model, an attacker can infiltrate at various stages of the process chain (see Figure 4.1) to alter the ***integrity*** of the tools, algorithms, and firmware. Moreover, they may add exogenous inputs $e_1$, $e_2$ and $e_3$ during the transfer of information from one stage to the other in the digital process chain. Let us consider $y$ to be the control signals

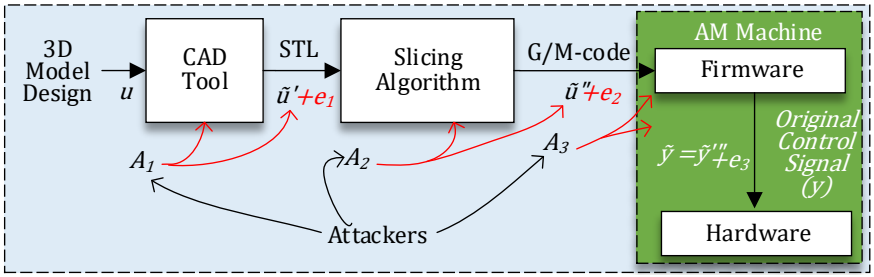



Figure 4.1: Adversary attack points in the digital process chain

to the physical components of the cyber-physical additive manufacturing system machine, provided $u$ is the true information. In this case, $\tilde{u}'$, $\tilde{u}''$ and $\tilde{y}'''$ are the false information produced in the process chain due to the compromised ***integrity*** of the CAD tool, slicing algorithm, and the firmware respectively. We can observe that the attacks on the digital process chain by different attackers $A_1$, $A_2$ and $A_3$ always results in the modification of the control signals $y$. This phenomenon is what separates the CPS from traditional information and technology systems. In FDM based cyber-physical additive manufacturing system, the outcome of kinetic attack will result in the variation of control signal $y$ to $\tilde{y}$, such that it modifies the initial 3D design of the object.

**Remark 1.** *In FDM based cyber-physical additive manufacturing system, the change in the information flow by an attacker will result in the change of control parameters* $y = [v, a, t, d]$ *responsible for controlling the dynamics of the machine.*

Where $v = [v_x, v_y, v_z, v_e]$ represents the speed of the nozzle in different axis, along with the speed of extrusion, and $v_{i\in\{x,y,z,e\}}\mathbb{R}_{\geqslant 0}$. $a = [a_x, a_y, a_z, a_{xy}, a_{xz}, a_{yz}, a_{xyz}]$ represents the axis of movement such that $a_{i\in\{x,y,z,xy,xz,yz,xyz\}} \in \{0, 1\}$, and $a = 1$ represents presence of movement in the given axis and $a = 0$ represents absence of movement. $t \in \mathbb{R}_{\geqslant 0}$ represents the temperature of the nozzle. $d = [d_x,\ d_y,\ d_z, d_e]$ represents the distance of the nozzle in different axis, along with the amount of extrusion, and $d_{i\in\{x,y,z,e\}} \in \mathbb{R}$. Hence, the kinetic cyber-attack in FDM based cyber-physical additive manufacturing system, will cause the information $u$ to be altered such that the final control parameters to the physical components are altered to $\tilde{y} = [\tilde{v}, \tilde{a}, \tilde{t}, \tilde{d}]$.

**Definition 1.** *With reference to remark 1, the kinetic cyber-attack in FDM can be defined as change in the information flow u in the digital process chain such that:*

$$\begin{bmatrix} v \\ a \\ d \\ t \end{bmatrix} \pm \begin{bmatrix} e_v \\ e_a \\ e_d \\ e_t \end{bmatrix} = \begin{bmatrix} \tilde{v} \\ \tilde{a} \\ \tilde{d} \\ \tilde{t} \end{bmatrix} \tag{4.1}$$

*and* $\tilde{y} \neq y$, *when* $\sum_{i\in\{v,a,d,t\}} e_i > 0$. *Here* $\{e_v, e_d, e_t\} \in \mathbb{R}_{\geqslant 0}$, $e_a \in \{0, 1\}$.

## 4.3 KCAD Method

Let $Y \rightarrow O$ be a side-channel, where $Y$ and $O$ represent random variables denoting control information parameters and observed *analog emission* respectively. Then, KCAD method leverages the fact that these variables have high mutual information.

### 4.3.1 Mutual Information

**Remark 2.** *Let the observed analog emissions be $o(t)$, then the control parameters, $y = [v, a, t, d]$, responsible for controlling the dynamics of the system, emit analog emissions such that the mutual information $\{I(V;O),\ I(A;O),\ I(T;O),\ I(D;O)\} > 0$. Where $(V, A, T, D)$ are random variables.*

The random variables $O$, $V$, $T$, and $D$ are continuous where as $A$ is discrete. Let $f(o)$, $f(v)$, $f(t)$, and $f(d)$ be probability density function (*pdf*) of continuous random variables $O$, $V$, $T$, and $D$ respectively. And for all $k, j \in \{o, v, t, d\}$, let $f_{k\neq j}(k, j) = f_{k\neq j}(j, k)$ be the joint *pdf*. The conditional *pdf* for these random variables is then defined as $f_{k\neq j}(k|j) = \frac{f_{k\neq j}(k,j)}{f(j)}$. Then, we can calculate the differential entropy of these random variables as follows:

$$h_{K\in\{O,V,T,D\}}(K) = -\int f(k)log(f(k))dk \tag{4.2}$$

Similarly, the conditional differential entropy of these random variables can be given as follows:

$$h_{K,J\in\{O,V,T,D\}}(K|J)_{K\neq J} = -\int\int f(k,j)log(f(k|j))dkdj \tag{4.3}$$

The mutual information between the observed *analog emission* and the control parameters can be given as follows:

$$I_{K\in\{V,T,D\}}(K;O) = h(K) - h(K|O) \tag{4.4}$$

Let $f(a)$ be a probability distribution function of the discrete random variable $A$. Then the entropy of $A$ can be calculated as follows:

$$H(A) = -\sum f(a)log(f(a)) \tag{4.5}$$

For calculating the mutual information between $O$ and $A$, we can divide the values of $O$ into bins of length $\varepsilon$. If $H(O_\varepsilon)$ be the entropy of $O$ after discretization, we have $h(O) = \lim_{\varepsilon \to 0} [H(O_\varepsilon) + log(\varepsilon)]$. And the mutual information can be calculated as:

$$I(A;O) = H(A) - H(A|O_\varepsilon) \tag{4.6}$$

The calculation of mutual information between two continuous random variables requires estimation of the probability density functions, which are then used in Equations 4.2 and 4.3. Kernel probability density estimation can be used for estimating the *pdf* based on the experimental data as:

$$\tilde{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{4.7}$$

Where $K$ is a real-valued integrable kernel function and $h$ the bandwidth. There are various kernel function that can be used for the estimation.

Our proposed KCAD method can be defined as a pipeline of deterministic algorithms which takes continuous observable *analog emissions* $o(t)$ as input along with the information flow $U$ in the form of G-codes $G_t = [g_1, g_2, g_3, ..., g_t]$ from which the control parameters $v, a, d$, and $t$ are parsed. The information ($U$) acquired by KCAD is assumed to be from a secure channel and free from any modification. KCAD method infers the *analog emission* $O$ based on the given control parameters. Given the presence of a kinetic cyber-attack, the control parameters are changed to $\tilde{y} = [\tilde{v}, \tilde{a}, \tilde{d}, \tilde{t}]$ with observed *analog emissions* being $\tilde{O}$, such that $|\tilde{O} - O| = e$. And for $e > e^T$, emission variation threshold, the output of the detection system is *True*, denoting presence of an attack.

**Definition 2.** *Attack Detectability: Given the input $O$ and $G$ with parsed variables $v, a, d$, and $t$, kinetic attack to the system, such that $\sum_{i \in \{v,a,d,t\}} e_i > 0$, is detected by KCAD method if its output is true.*

### 4.3.2 KCAD Architecture

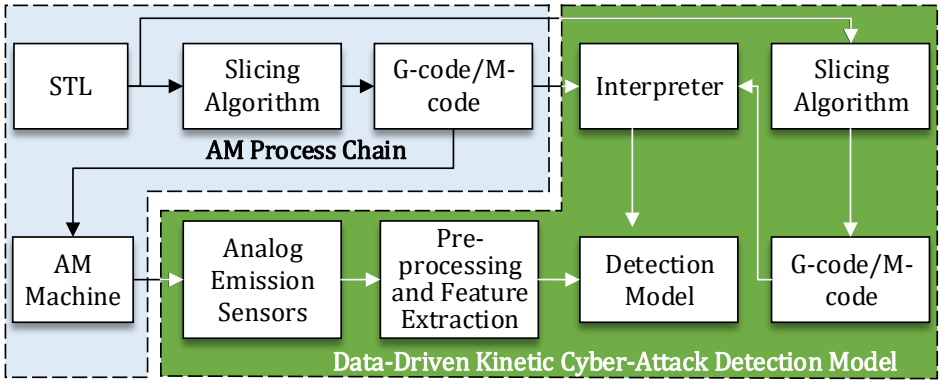



Figure 4.2: Architecture of data-driven kinetic cyber-attack detection method

The architecture of the simplified KCAD method is shown in Figure 4.2. One of the advantages of this method is that it can be placed to monitor the information flow in any of the stages of the digital process chain. With this, the attack on the integrity of any tools, firmware, and algorithms can be detected if it has the corresponding effect on the dynamics of the system. KCAD runs in parallel to the cyber-physical additive manufacturing system while it is printing, and non-intrusively and continuously acquires the observable *analog emissions*. The components of KCAD that work in parallel to the cyber-physical additive manufacturing system, depends on which point of the process chain we feed the information to it. For example, if we want to detect the attack on the integrity of the firmware of the cyber-physical additive manufacturing system, the input to the KCAD can just be the G-code/M-code. The slicing algorithm in it can thus be switched off. However, the channel through which the information passes to the KCAD method from different point of the digital process chain is assumed to be secure. This means that the KCAD method will always receive original/unmodified cyber data from the digital process chain.

*Analog Emission Sensors:* Various sensors (piezoelectric, current, electromagnetic, etc.) can be used to monitor the *analog emissions* from the cyber-physical additive manufacturing system system. Given the integrity attack that introduces values $e_v$, $e_a$, $e_d$, and $e_t$, in the

control parameters $(v, a, d, t)$, the sampling frequency $(F_s)$ and bandwidth $(B)$ should be such that it can measure corresponding changes $e_v$, $e_a$, $e_d$, and $e_t$ in the *analog emissions* $o(t)$. Moreover, distance and angle of placement of the sensors also affect the Signal to Noise Ratio (SNR) given as:

$$SNR_{dB} = 10log_{10}(\frac{P_{Signal}}{P_{Noise}}) \tag{4.8}$$

Hence, the choice and placement of the sensors depends on the choice of side-channel and the relation between the *analog emissions* and the control parameters. For choice of side-channel and the corresponding *analog emissions*, Equations 4.4 and 4.6 can be used as a measure of relation between the observable *analog emissions* and the control parameters. Based on this measurement, the observed values can either be incorporated or discarded from group of features to be used for estimating the behavior of the system in KCAD.

*Pre-processing and Feature Extraction:* Pre-processing is done to improve the SNR by removing the known noise signals from the *analog emissions* that are independent of the control parameters. If observable *analog emission*, $o(t) \neq f(y(t))$, where $y$ represents the control parameters, then the observed *analog emission* can be considered as noise. The signals acquired by sensors can be too large for the estimation algorithms used in the detection model. Hence, it is necessary to extract only the informative values from the signal to improve the processing time of the detection model. For each observed signal various values (features) are derived using the original signal.

$$O_t = \begin{bmatrix} o_1^{f^1} & o_1^{f^2} & o_1^{f^3} & \dots & o_1^{f^n} \\ o_2^{f^1} & o_2^{f^2} & o_2^{f^3} & \dots & o_2^{f^n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ o_t^{f^1} & o_t^{f^2} & o_t^{f^3} & \dots & o_t^{f^n} \end{bmatrix} \tag{4.9}$$

Where the column represents the number of features and the row represents the discretized

values of the signal $o(t)$. The type of features extracted is specific to the observed *analog emissions* (for acoustics see Section 4.2). For reducing the dimension of the extracted features, principle component analysis is used.

*Interpreter:* Each block (line) of instruction (G/M-code) sent to the cyber-physical additive manufacturing system consists of control parameters. These instructions or numerical control codes are converted to the canonical machining commands using interpreters such as NIST RS274NGC. In our KCAD method, a lighter version of Arduino G-code and NIST RS274NGC Interpreter is used to extract the control signals $v, a, d$, and $t$. These signals are then sent to the detection model.

*Detection Model:* The detection model uses the supervised learning approach to estimate the function $\hat{f}_i(O_t, \alpha_n)$ with $i = 1, 2, 3, 4$ using the training data-set of observed *analog emissions*. For each control parameters we estimate the function with respective parameter $\alpha$. Various predictive models can be estimated based on the initial training data-sets. The

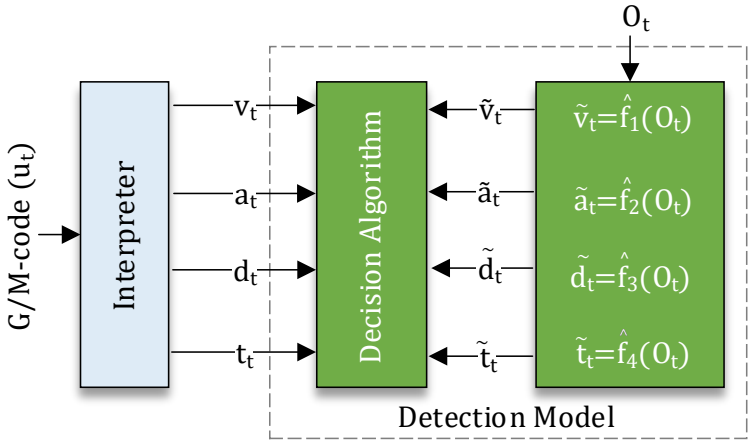



Figure 4.3: Data-driven detection model

training data-sets will require to balance the trade-off between bias and variance to find optimal parameter $\alpha$. $k-$fold cross-validation is used to perform data driven validation for the estimated function $\hat{f}_n(.)$. For the regression function estimation, learning algorithms such as *Gradient Boosting Regressor (GBR)*, *Ridge Regression*, *Stochastic Gradient Descent Regres-*

*sion (SGDR)*, *Bayesian Ridge Regression (BRidge)*, *Passive Aggressive Regression (PAR)*, *Decision Tree Regression (DTR)*, *Elastic Net Regression (ENet)*, *Linear Regression with Lasso*, and *k-Nearest Neighbor Regression (kNN)* is used. Each of these models is compared using metrics such as *explained variance*, *Mean Absolute Error (MAE)*, *Mean squared error (MSE)*, *Median Absolute Error*, and $R^2$ *Score*. For function that needs to be estimated for classification, classifiers such as *Support Vector Machine (SVC)* with *Linear* and *Radial Basis Function (RBF)* as kernels, *Logistic Regression*, *Stochastic Gradient Descent Classifiers*, *Ensemble of AdaBoost* and *Gradient Boosting* are used. They are compared using Receiver Operating Characteristic (ROC) curves, and the model with the least error is selected.

*Detection Algorithm:* The detection algorithm compares the real control parameters given by the interpreter and the values calculated by the estimated functions $\hat{f}_n(.)$. In Algorithm

**Algorithm 6:** Detection Algorithm.

**Input:** Real and Estimated Control Parameters $[v, a, d, t], [\tilde{v}, \tilde{a}, \tilde{d}, \tilde{t}]$
**Output:** Attack Flag $F_A$
1 Define Error Thresholds $e_v^T, e_a^T, e_d^T, e_t^T$
2 Initialize $f_v = 0, f_a = 0, f_d = 0, f_t = 0$
3 **for** *each* $i \in v, a, d, t$ **do**
4  **if** $|i - \tilde{i}| > e_i^T$ **then**
5  $f_i = 1$
6 **if** $\sum_{i \in \{v,a,d,t\}} f_i \geq 1$ **then**
7  $F_A = 1$
8 **return** $F_A$

6, in line 1, the error variation thresholds are defined. This value is based on the accuracy of the estimated functions $\hat{f}_n(.)$ during the training phase of the detection model. Lines 3 to 5 determines if the estimated control parameters and the real control parameters vary more than the error threshold. Lines 6-7 set the attack detection flag to high if any of the control parameters varies more than the error variation threshold. Finally in line 8 the attack flag is returned.

**Offline vs Online Model Estimation:** The model function $\hat{f}_n$ estimation can be done

either online or offline. This is necessary to consider because of the fact that cyber-physical additive manufacturing system machine will have varying observable *analog emission* over a long period of time due to wear and tear of the mechanical structures. Offline function estimation can have a shorter response time. However, an online estimation can be done for higher accuracy with a longer response time.

### 4.3.3 Acoustic Analog Emissions

Acoustic *analog emissions* is one of the observable emissions in the cyber-physical additive manufacturing system. The fundamental working principle behind KCAD method is that the *analog emissions* $o(t)$ must have high mutual information with the control parameters $y(t)$. As a proof of concept of the KCAD method, we will use acoustics as the observable *analog emissions* to detect the presence of kinetic cyber-attack on the cyber-physical additive manufacturing system. However, it is trivial to infer that the observed *analog emission* will have a weak relation with the control parameter temperature $t$. Therefore, only a kinetic cyber-attack affecting the control parameters $v, a$, and $d$ will be considered. The main source of acoustics in FDM based 3D Printers are the vibration of the stepper motors. These 3D Printers consists of at least one stepper motor to control the movement of the nozzle of the printer in each axis ($x$, $y$, and $z$ axis) [115]. These stepper motor consists of rotor (permanent magnet) and the stator (electromagnet). The varying radial electromagnetic force acting on the stator of the stepper motor produces vibration [116, 62, 65]. This vibration is the source of acoustics. The radial electromagnetic force is controlled by the control parameters, such as *speed* of the movement of the motor. However, the natural frequency of the stator is determined by the load, connected frame, and the structure of the stator [64]. Hence, resonance occurs when the vibration produced by the radial electromagnetic matches the harmonics of the natural frequency of the stator. This resonance frequency is different for different stepper motors responsible for moving the 3D Printer's nozzle in $x$, $y$, and $z$. This

will allow us to estimate functions to separate the movement in different axes. The *analog emission* sensors for capturing the acoustic emissions will require sampling frequency of more than 40 $kHz$ to capture the range of audible sound 20 $Hz$ to 20 $kHz$. During the pre-processing stage digital filter is used to remove, low and high frequency noise. Dynamic time

**Algorithm 7:** Dynamic Window Size Determination.

**Input:** Observed Analog Emission $o(t)$
**Output:** Dynamic Windows $w = [w_1, w_2, \ldots, w_n]$

1 Initialize $n = 1, i_{previous} = 1$
2 Extract Features $O_t$ // $n \rightarrow$Number of Features
3 **for** $i = 2$ *to* $t$ **do**
4  **if** $|\sqrt{(o_i^{f^1} - o_{i-1}^{f^1})^2 + \ldots + (o_i^{f^n} - o_{i-1}^{f^n})^2}| > d^T$ **then**
 // $d^T \rightarrow$Threshold Distance
5  $w_n = i - i_{previous} + 1$
6  $i_{previous} = i + 1$
7  $n = n + 1$
8 **return** $w$

warping is used to dynamically assign the window size $w$ for feature extraction. However, an initial fixed length window size (10-50 $ms$) will be used to extract features size as *Zero Crossing Rate*, *Energy Entropy*, *Spectral Entropy*, and *Mel Frequency Cepstral Coefficients (MFCCs)*. Using these features, euclidean distance is measured to define the dynamic window size for accurate feature extraction.

As shown in Algorithm 7, line 4 measures the euclidean distance between the features of the previous analog observation with the current *analog emission* in discrete time series (based on the sampling frequency). If this distance is greater than the threshold distance $d^T$, which is determined by measuring the distance between the *analog emission* of training data-sets. As windowing is done to extract the features from the observed *analog emissions* $o(t)$, we can determine the control parameter $v$ from $d$ and vice-versa, where $v = d/w$. Where $w$ is length of the window in seconds.

**Remark 3.** *For FDM based cyber-physical additive manufacturing system, KCAD method will be able to monitor any kinetic attacks modifying the control parameters* $v, a$, *and* $d$

Table 4.1: Mutual information in bits between features and control parameters

| | **Features** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $o_t^{f^1}$ | $o_t^{f^2}$ | $o_t^{f^3}$ | $o_t^{f^4}$ | $o_t^{f^5}$ | $o_t^{f^6}$ | $o_t^{f^7}$ | $o_t^{f^8}$ | $o_t^{f^9}$ | $o_t^{f^{10}}$ | $o_t^{f^{11}}$ | $o_t^{f^{12}}$ | $o_t^{f^{13}}$ | $o_t^{f^{14}}$ |
| $v_x$ | 0.79 | 0.95 | 0.16 | 0.81 | 0.74 | 0.76 | 0.35 | 1.18 | 1.51 | 0.31 | 0.55 | 0.69 | 0.75 | 0.71 |
| $v_y$ | 0.27 | 0.67 | 0.11 | 0.43 | 0.41 | 0.24 | 0.23 | 1.31 | 1.10 | 0.23 | 0.44 | 0.45 | 0.55 | 0.60 |
| $v_z$ | 0.16 | **0.07** | **0.0806** | **0.08** | **0.07** | **0.08** | **0.07** | **0.07** | **0.07** | **0.07** | **0.07** | **0.07** | **0.08** | **0.09** |
| $a$ | 0.58 | 0.82 | 0.18 | 0.69 | 0.69 | 0.53 | 0.44 | 1.91 | 1.17 | 0.64 | 0.74 | 0.22 | 0.40 | 0.39 |

*by analyzing the variation in the acoustic analog emissions to the corresponding control parameters $v$ and $a$.*

### 4.3.4 Performance Metrics

The performance of an attack detection method can be measured with two metrics *True Positive Rate (TPR)* and *True Negative Rate (TNR)*.

$$TPR = \frac{TP}{TP + FN} \tag{4.10}$$

where $True\ Positive$ $(TP)$ is the total number of positive detection when there is an attack in the system, and $False\ Negative$ $(FN)$ is the total number of negative detection during the presence of an attack. Similarly,

$$TNR = \frac{TN}{TN + FP} \tag{4.11}$$

where $True\ Negative$ $(TN)$ is the total number of positive detection when there isn't any attack to the system, and $False\ Positive$ $(FP)$ is the total number of positive detection when there isn't any attack to the system. Then, the accuracy of the system can be measured as follows:

$$Accuracy = \frac{TP + TN}{Total\ Sample} \tag{4.12}$$

## 4.4 Experimental Results

### 4.4.1 Experimental Setup

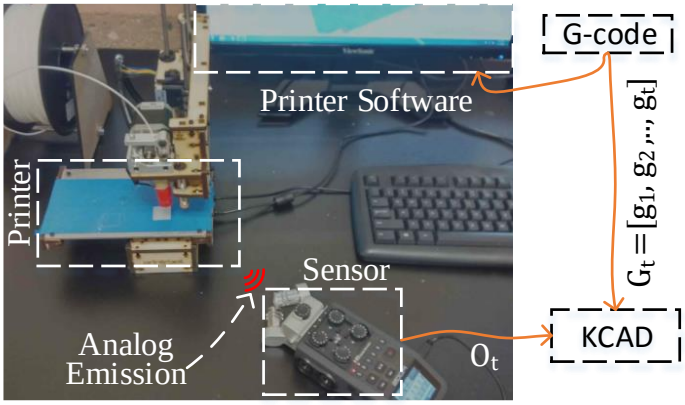



Figure 4.4: KCAD method experimental setup

The experimental setup for the KCAD method evaluation is shown in Figure 4.4. This evaluation tests KCAD performance against integrity attacks on the printer's firmware. In a simple scenario, a modified firmware is installed in the 3D Printer by an attacker. This attack modifies the control signal to the 3D Printer, which introduces variation in the geometry of the 3D object. The audio recorder is placed at an optimal distance to acquire the acoustics. In the experiment, the recorder is placed at 45° angle to the $x$ and $y$ axis to acquire the variation of the signal in both directions with a single recorder. During the training phase, G-codes are written to move the nozzle of the printer in various $x$ and $y$ axis directions with various printing speeds (400 $mm/min$ to 4500 $mm/min$ with 100 $mm/min$ step size). These speed ranges are machine specific. Using this training data, we estimated the model function for the control parameters $v_x$, $v_y$, and $a = [a_x, a_y, a_z, a_{xy}]$. Using a similar approach, model functions for parameters $v_e, v_z$, different directions, and $a = [a_{yz}, a_{xz}, a_{xyz}]$ can be estimated.

### 4.4.2 Mutual Information Calculation

To demonstrate the dependency of the acoustic *analog emissions* with the control parameters $v = [v_x, v_y, v_z]$ and, the $a = [a_x, a_y, a_z, a_{xy}]$ mutual information between them is calculated. For control parameters $a$, it is treated as a discrete random variable with different labels depending on the combination of axis movements. Table 4.1 shows the mutual information calculated for the control parameters and the features extracted from the observed *analog emissions*. We can see that different features have varying mutual information with the control parameters. Also, it can be observed that the mutual information between the speed in $z - axis$ and the *analog emission* is comparatively low. This is due to the reason that speed in $z - axis$ is almost constant in most of the 3D Printers. The estimation function utilizes different features on the basis of the mutual information and principal component analysis to select the features that are most relevant.

Table 4.2: Regression models comparison for $\hat{f}_v(.)$.

| Model | MSE | Variance | MAE | Median AE | $R^2$ |
|---|---|---|---|---|---|
| GBR | 0.0076 | 0.9923 | 0.0037 | 0.0167 | 0.9923 |
| Ridge | 0.0147 | 0.9854 | 0.0090 | 0.0775 | 0.9851 |
| SGDR | 0.0148 | 0.9852 | 0.0090 | 0.0785 | 0.9850 |
| BRidge | 0.0149 | 0.9852 | 0.0089 | 0.0772 | 0.9849 |
| PAR | 0.0183 | 0.9817 | 0.0095 | 0.0899 | 0.9815 |
| DTR | 0.0258 | 0.9741 | 0.0090 | 0.0582 | 0.9740 |
| ENet | 0.0786 | 0.9212 | 0.0210 | 0.1990 | 0.9208 |
| Lasso | 0.1015 | 0.8980 | 0.0241 | 0.2331 | 0.8976 |
| kNN | 0.0025 | 0.4997 | 0.0014 | 0.0182 | 0.4997 |

### 4.4.3 Model Function Estimation

The function, $\hat{f}_i(O_t, \alpha_n)$, estimation is the fundamental step in our KCAD method. The parameter $[\alpha_1, \alpha_2, \ldots, \alpha_n]$ are responsible for minimizing the cost functions used by the learning algorithms. This estimation is done in the training phase. Based on the relation

between the control parameters and the features extracted from observed *analog emissions*, the estimated functions can be used for regression or classification. The relation between the control parameter $v$ and observed *analog emissions* $o(t)$ both being continuous random variable can be estimated using regression algorithms. Where as, the parameter $a$ is a discrete random variables and and we have to use classifiers for estimating the function $\hat{f}(.)$. We will use various algorithms and perform the comparison between them to select the function $\hat{f}(.)$, that gives the least error.

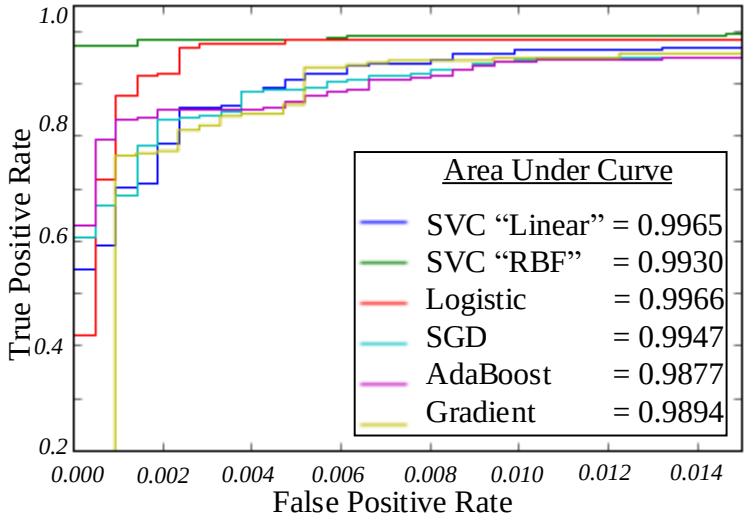



Figure 4.5: Receiver operating characteristics curve of classifiers

*Regression:* Function $\hat{f}_{[v_x,v_y]}(O_t, \alpha_n)$ is estimated for the control parameter $v$ based on $O_t$. From Table 4.2, it is clear that *Gradient Boosting Regression*, outperforms rest of the regression models in terms of the error metrics. Hence, it is selected to estimate the function for the control parameters $(v_x, v_y)$. Function for $v_z$ is not estimated as speed in z-axis is constant.

*Classification:* For the parameter $a$, various functions estimated along with their performance is compared in Figure 4.5. The estimation function is treated as one versus the rest classification and the ROC is calculated as an average of ROC of all the classes i.e various movement axis, $[a_x, a_y, a_z, a_{xy}]$. From Figure 4.5, *logistic regression* classifier is selected as the optimal function for the classification. In Figure 4.6, the ROC curve of various classes

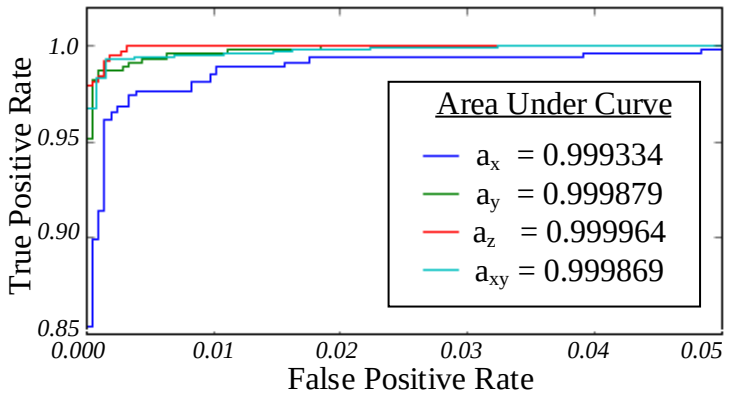



Figure 4.6: Receiver operating characteristics curve of classification with One versus rest calculated as a one versus rest classification using *Logistic Regression* classifier is presented. Since the area under the curve for the $z - axis$ movement class is higher, the detection of the presence of nozzle movement in $z - axis$ is easiest compared to the other classes. With these functions estimated for various control parameters, KCAD can effectively detect the variation in the *analog emissions* with the expected emissions in the presence of kinetic cyber-attacks on the firmware of the 3D Printer.

Table 4.3: True positives for speed variation

| $\delta v$ | TP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| mm/min | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | TP |
| 1000 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 144 |
| 500 | 16 | 16 | 16 | 16 | 16 | 14 | 15 | 16 | 14 | 139 |
| 300 | 16 | 16 | 16 | 13 | 10 | 7 | 7 | 8 | 7 | 92 |
| 200 | 13 | 16 | 10 | 9 | 8 | 6 | 8 | 8 | 6 | 84 |
| TPR | | | | | | | | | | 0.7968 |

### 4.4.4 Results for Detection of Kinetic Attack

We developed a zero-day kinetic attack on the 3D Printer's firmware to test the KCAD method. Our attack modifies the nozzle speed in the $x$ and $y$ direction while printing, thus effectively changing the dimension of the object. Additionally, the axis values are changed resulting in the deformation of the object. For detecting the variation of speed on $x - axis$ and $y - axis$, the speed is varied from 600 $mm/min$ to 3500 $mm/min$, while the attack is

assumed to introduce range of variation in the original speed i.e. from 50 $mm/min$ to 1000 $mm/min$. From the function estimation, error threshold for speed $e_v^T$ is set as 200 $mm/min$.

Table 4.4: False positives for speed variation

| $\delta v$ | FP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| mm/min | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | FP |
| 1000 | 5 | 4 | 3 | 6 | 5 | 7 | 7 | 6 | 8 | 51 |
| 500 | 4 | 3 | 2 | 5 | 5 | 8 | 6 | 8 | 8 | 54 |
| 300 | 2 | 2 | 3 | 5 | 5 | 7 | 6 | 6 | 7 | 47 |
| 200 | 3 | 1 | 3 | 5 | 2 | 8 | 7 | 8 | 7 | 44 |
| FPR | | | | | | | | | | 0.3402 |

Table 4.3 and 4.4 show that for higher speed and lower speed variation ($\delta v$), the true positives are lower compared to the low speed and high speed variation. However, the false positives are higher for higher speeds.

Table 4.5: True positives for distance variation

| $\delta d$ | TP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| mm | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | TP |
| 20 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 14 | 16 | 142 |
| 10 | 16 | 15 | 16 | 16 | 16 | 15 | 14 | 13 | 14 | 135 |
| 5 | 16 | 13 | 14 | 10 | 12 | 14 | 13 | 12 | 15 | 119 |
| 3 | 14 | 10 | 12 | 10 | 11 | 10 | 12 | 11 | 13 | 103 |
| TPR | | | | | | | | | | 0.8663 |

Intuitively, this is due to the fact that feature extracted prominently consists of MFCC, which focuses on extracting more features from lower frequency range rather than higher frequencies causing poor function estimation for higher speeds. KCAD accuracy for detection of attack on control parameter $v$ can be calculated using Equation 4.12 as 72.83%.

For testing the performance of KCAD method for detecting the change in the control parameter $d$ introduced by the modified firmware, distance is varied from 3 $mm$ to 20 $mm$. The error threshold for the distance is selected ad 3 $mm$ based on the error in the estimated function.

Table 4.6: False positives for distance variation

| $\delta d$ | FP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| mm | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | FP |
| 20 | 4 | 4 | 4 | 6 | 4 | 3 | 3 | 5 | 4 | 37 |
| 10 | 3 | 4 | 4 | 5 | 4 | 4 | 4 | 5 | 5 | 38 |
| 5 | 2 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 7 | 42 |
| 3 | 2 | 4 | 7 | 7 | 5 | 2 | 5 | 5 | 8 | 45 |
| FPR | | | | | | | | | | 0.2812 |

Table 4.5 shows that the number of true positive is decreasing with the increasing speed and lower distance variation $\delta d$. Moreover, the number of false positive is increasing with the increasing speed. Using Equation 4.12, the accuracy for detection of attack on control parameter $d$ is calculated to be 79.25%. We modified the movement in $x$, $y$, $z$, and $xy$ axis to

Table 4.7: True positives for axis variation

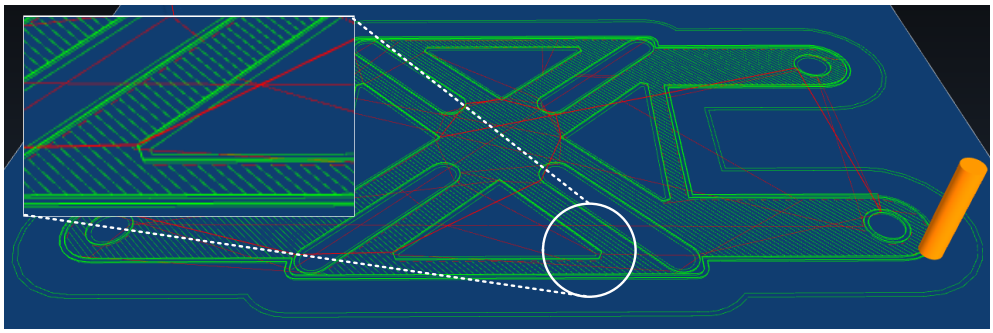
| $Axis, Total$ | TP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| TP | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | TP |
| $a_x, 32$ | 32 | 32 | 28 | 28 | 24 | 21 | 21 | 20 | 19 | 225 |
| $a_y, 32$ | 31 | 32 | 27 | 25 | 23 | 19 | 19 | 19 | 18 | 213 |
| $a_{xy}, 24$ | 20 | 21 | 21 | 19 | 19 | 20 | 19 | 18 | 17 | 174 |
| $a_z, 24$ | 24 | 20 | 19 | 18 | 18 | 19 | 18 | 19 | 18 | 173 |
| TPR | | | | | | | | | | 0.7787 |

measure the KCAD performance against firmware modification attacks that vary the control parameter $a$, which determine the movement axis. Table 4.7 shows that the true positive rate is decreasing with increase in speed and false positive rate in increasing with increase in the speed. The accuracy for detection of attack on control parameter $a$ is 79.07%.

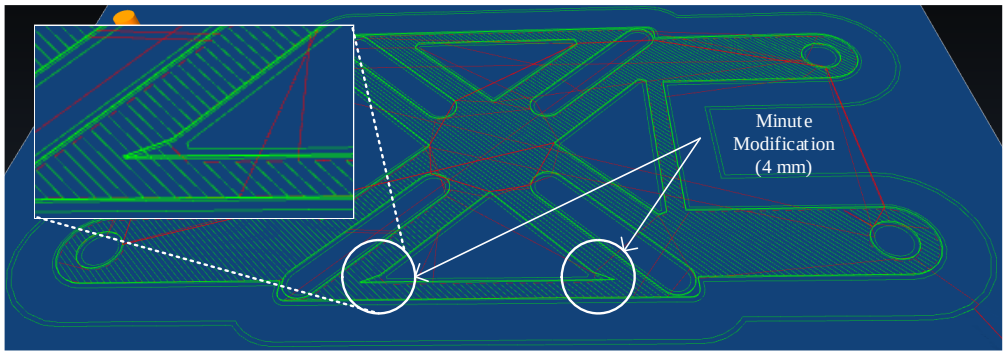
Table 4.8: False positives for axis variation

| $Axis, Total$ | FP for Speed (*100 mm/min) | | | | | | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| TN | 7 | 9 | 12 | 15 | 17 | 20 | 25 | 30 | 35 | FP |
| $a_x, 32$ | 2 | 2 | 2 | 3 | 4 | 6 | 6 | 7 | 9 | 41 |
| $a_y, 32$ | 2 | 3 | 4 | 2 | 4 | 5 | 4 | 9 | 8 | 41 |
| $a_{xy}, 24$ | 4 | 4 | 6 | 6 | 7 | 5 | 6 | 7 | 8 | 53 |
| $a_z, 24$ | 5 | 6 | 7 | 8 | 7 | 8 | 7 | 8 | 8 | 64 |
| FPR | | | | | | | | | | 0.1974 |

Our KCAD method relies on the fact that any *zero-day kinetic cyber-attack* results in variation of control parameters $v, a$, and $d$. Hence, treating these parameters as synthetic benchmarks, the accuracy of the KCAD method, for measuring the variation of various control parameters $v, a$, and $d$ is 77.45%.



a) Original G-code Trace.




b) G-code Trace after Kinetic Attack.

Figure 4.7: Attack on base plate of a quad copter

### 4.4.5 Test Case: Base Plate of a Quad Copter

As a test case, we present an analysis on a flight controller base plate [117], which is a part of the quadcopter that can be printed using a 3D Printer. We assume that the modified firmware, as a result of *zero-day kinetic cyber-attack*, introduces variation in the certain part of the code by adding 4 $mm$ to the $x - axis$ movement distance. Such small changes in the design of an object can compromise their structural integrity during use, and lead to a catastrophic failure. The original design of the base plate is shown in Figure 4.7 (a). As

a result of an attack, minute modification introduced in the design is shown in Figure 4.7 (b). This modification might not be visible to human eyes, however, it compromises the structural integrity of the base plate. KCAD method detected the variation in the x-axis introduced in all three layers by the firmware modification attack.

## 4.5 Limitations

In this section, we provide an open discussion on the limitations of our KCAD method: **1. Printer Variation:** KCAD is machine specific. Hence, for implementation, the function estimation (training) has to be conducted before it can be implemented. Different FDM based 3D Printers will emit different acoustic signature based on the type of motors used and the structure of the frame. This has to be studied before KCAD can be implemented. Different *analog emissions* and their respective features have to be analyzed for model function estimation. However, this has to be done only once before the implementation.

**2. Complex Attacks:** In the experimental section, we have tested the KCAD with firmware modification attacks introducing simple variation in the $x$, $y$ axis. However, more complex variation can be introduced by the attack, with attacks modifying both the $x$ and $y$ axis together. In such scenarios, more function has to be estimated for a combination of control parameters.

**3. Sensor Placement:** The *analog emissions* sensors placement should not obstruct the printing process. However, it must be able to capture the emissions with high SNR. This requires analyzing various sensors and their optimal position. It might have to be placed inside the system for better SNR. However, this can only be done if it does not obstruct the printing process.

## 4.6 Summary

This chapter presented a novel data-driven kinetic cyber-attack detection method, which can be placed non-intrusively and can monitor the system during run-time. We have performed the analysis of acoustic *analog emissions* to measure the feasibility of such a system, and demonstrated that acoustics have high mutual information with the control parameters parsed from the cyber-domain data. We have tested our system on an FDM based 3D Printer, assuming that integrity attack on the printer firmware eventually modifies the control parameters to the physical components. We have tested the performance of our method with a variety of speed, distance, and axis that can be an outcome of the kinetic cyber attack in the digital process chain of the additive manufacturing. KCAD method achieves a high accuracy of 77.45%. This provides a good starting point and proof of concept for the proposed attack detection method.

# Chapter 5

# Part I: Security Analysis using Generative Adversarial Networks

## 5.1 Introduction and Related Work

In this chapter, we will present a generative adversarial networks based modeling approach for performing security analysis in cyber-physical systems. The fourth industrial revolution will encompass a large scale use of cyber-physical systems. This manufacturing system in combination with CPS is also known as Cyber-Physical Production Systems (CPPS) [118]. CPPS is an integration of sub-systems from multiple cyber and physical domains interconnected through communication networks. Using CPPS will aid in making the factory units smarter and adaptive; however, due to the tight interactions between the cyber and physical domains, there may exist various cross-domain vulnerabilities in CPPS.

There are two types of exploits of cross-domain vulnerabilities that we are primarily interested in: *kinetic-cyber attacks* and *side-channel attacks* as presented in chapter 4 and 2. Both types of attacks may even be composed of smaller exploits that target confidentiality,

integrity, or availability. Existing CPPS modeling tools were created for the purpose of analyzing the system-level performance, reliability, energy efficiency, and quality of controls. Security is ignored by these tools and left as an afterthought of system design. However, most of the existing CPS security research work only target known vulnerabilities in specific domains. Afterward, they suggest an ad-hoc repair such as patching software and/or replacing hardware components without showing that the repaired system is free of further vulnerabilities [114, 119]. Some of these CPS also end up being a part of the overall CPPS.

Since these solutions cannot analyze and explore all the potential vulnerabilities, they can not answer the questions such as, "How secure is this CPPS against a confidentiality attack by a specific attacker?" or "Can we detect an integrity or availability attack on a CPPS?" To address these questions and improve the CPPS security research, we hypothesize that it is necessary to have a new modeling approach that takes into consideration the signal and energy flows of a system. We propose a security model that abstracts the relationship between the energy flows and signals flows to answer questions like the aforementioned ones.

### 5.1.1 Research Challenges

Notice that since the attack models mostly describe the capability of attackers rather than the system, one might assume that the existing attack models may be directly applied in CPPS security analysis. However, there exist the following research challenges to create a system-level model for CPS security analysis:

1. The existing security properties are mostly only proposed for analyzing attacks in the cyber domain [114]. Thus, analyzing the cross-domain attacks in CPPS requires new types of security properties applicable in both the cyber and physical domains.

2. The existing system-level behavior models in CPPS require various Models of Com-

putation (MoCs)[1], for the cyber domain and physical domain (an MoC cannot be cross-domain) [9]. In order to analyze cross-domain security, a unified system behavior of interest for CPPS is required.

3. In the CPPS environment, there are multiple sub-systems interacting with each other; therefore, information leakage or attack detection needs to be performed across multiple sub-systems.

To address these challenges, researchers have proposed to use information flow as the basis of a unified behavior model of cyber and physical domains for security analysis in [120]. However, [120] only focused on the application related to commodity flows, and proposed a coarse-grained information flow that models the commodity flow changes as an event. It fails to provide the capability of a quantified analysis of cross-domain attack and detection capability.

Modeling information flows requires a statistical method. However, there is no guarantee that collected data during both design time and run-time is sufficient to create an accurate security model. In this work, we use generative adversarial networks (see Section 5.1.2) to acquire a better estimate of the distribution (conditional in this paper) of the data. One hand if there is a large amount of data available, the discriminator of the generative adversarial network is able to estimate the data distribution, on the other, the generator, since it never sees the real data estimates the distribution without overfitting on the limited data, thus providing better distribution estimation. For security analysis, we consider abstracting a CPPS by the signal and energy flows and deriving the conditional densities among flow pairs using Conditional Generative Adversarial Network (CGAN).

[1]An MoC is a set of allowable operations used in computation and their respective costs (e.g., timing, performance, and memory overhead)

### 5.1.2 Preliminaries

First, let us present some preliminary definitions about the energy and signal flows as well as GANs before explaining the details about our proposed CGAN-based security model of the CPPS.

***Signal Flow***: A signal flow is modeled as a discrete signal may be defined as a random variable $F_S$ which have $n$ number of possible values, $F_S \in \{f_1, f_2, ..., f_n\}$. We define a set of $n$ events $E = \{E_1, E_2, ..., E_n\}$, where $E_i = 1$ when $F_S = f_i$, and $E_i = 0$ otherwise. We assume the probabilities for $E_i$ are known as $Pr(E_i)$.

***Energy Flow***: We define energy flow as a continuous-domain time dependent variable $F_E$. Given a feature construction function $f_X(\cdot)$, we may construct a set of feature vectors $X = f_X(F_E)$. Next, given a feature extraction and selection function $f_Y(\cdot)$, we may extract a set of more relevant feature vectors $Y = f_Y(X)$. Assume $Y = \{Y^1, Y^2, ...Y^m\}$, where each feature vector $Y^i$ may have $n_i$ number of possible values $Y^i \in \{y_1^i, ..., y_{n_i}^i\}$. We define a set of $n_i$ events $E_i = \{E_1, E_2, ..., E_{n_i}\}$, where any event $E_{i_j}$ is true if $Y^i = y_j^i$ is met. We assume the probability for $E_{i_j}$ is known as $Pr(E_{i_j})$.

***Generative Adversarial Network***: A Generative Adversarial Network (GAN) consists of a generative model $G$ that captures the probability distribution of the flow's data $Pr(F_1)$, and a discriminator model $D$ that estimates the probability that a sample of data x came from the training data rather than the generative model, $Pr(x \in F_1)$ [121]. GAN is a form of a maximum likelihood estimator that is trained to estimate the true probability distribution of a random variable by utilizing the empirical distribution derived from the training data. It achieves this by minimizing the *Kullback-Leibler divergence* between the empirical distribution and the generator's distribution, as follows [121]:

$$\theta^* = \arg\min_{\theta} D_{KL}(Pr_{data}(F_1) \;||\; Pr_G(F_1; \theta)) \tag{5.1}$$

where $\theta$ are the model parameters. The models used in these networks are generally deep convolutional neural networks. Some advantages of using GAN over other methods is that it does not require Markov chains (where the chance of state explosion is high) and they can generate samples in parallel [121]. A Conditional Generative Adversarial Network (CGAN) is a variation of the GAN, where the generator and the discriminator both receive a conditional variable $F_2$ (see Figure 5.2). The primary objective of the generative model $G$ is to learn the conditional probability distribution $Pr(F_1|F_2)$. In order to do this, it will require the labeled data $(f_{1_1}, f_{2_1})$, $(f_{1_2}, f_{2_2}), \ldots (f_{1_n}, f_{2_n})$. Here $\{f_{1_1}, f_{1_2}, \ldots, f_{1_n}\}$, $\{f_{2_1}, f_{2_2}, \ldots, f_{2_n}\}$ are the values taken by the random variable $F_1$ and $F_2$, respectively. $G$ and $D$ have a common objective function based on the two-player mini-max game, as follows [122]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{F_1 \backsim Pr_{data}(F_1)}[log(D(F_1|F_2))] + \mathbb{E}_{Z \backsim Pr_Z(Z)}[log(1 - D(G(Z|F_2)))] \quad (5.2)$$

where $Z$ is a noise random variable that is provided along with $F_1$ and $F_2$ to $G$ for training.

In Equation 5.2, the model $G$ is trained to minimize $log(1 - D(G(Z|F_2)))$ so that it may generate samples more similar to the training data. $D$ is trained to maximize $log(D(F_1|F_2))$ so that it may accurately differentiate between the real data and the generated data. Given enough time and capacity, the end result will be that $G$ is trained to accurately estimate the $Pr(F_1|F_2)$ and $D$ cannot differentiate $G$'s output from real or generated data. This is an important concept for our security model because using $Pr(F_1|F_2)$, one can estimate a $(signal/energy)$ flow based on another flow.

### 5.1.3 Novel Contributions

To resolve the challenges of CPPS security modeling, our novel contributions are as follows:

1. **Conditional Generative Adversarial Network-based Model for Security Anal-**

**ysis of CPPS (Section 5.2)** which models the conditional probability distribution between the various information flows (cyber-domain data and physical domain energy flow).

2. **Automatic Model Generation Algorithm (Section 5.3)** which includes a graph search and pruning algorithm to reduce the complexity of the model and simulation/ experimental methods to estimate and generate the proposed behavior model.

3. **A Case Study (Section 5.4)** for analyzing information leakage from multiple physical emissions in a single sub-system (additive manufacturing) to demonstrate the applicability of the proposed modeling technique in security analysis.

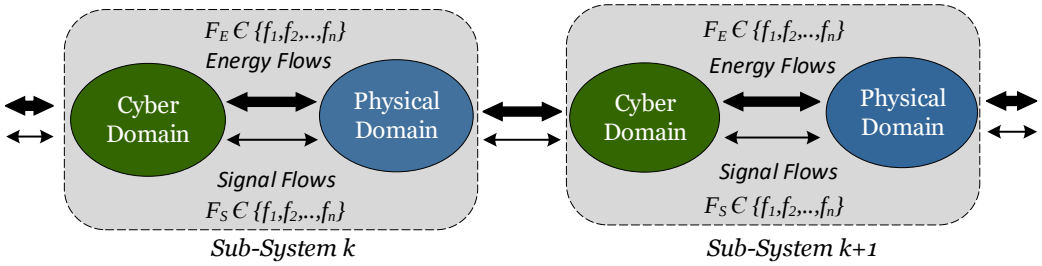



Figure 5.1: Cyber-physical production system with multiple sub-systems

## 5.2 CGAN-based CPPS Security Model

A typical CPPS model (see Figure 5.1) includes multiple sub-systems $(1, 2, \ldots, n)$. In each sub-system, there are cyber and physical domain components with energy and signal flows connecting them. Moreover, the energy and signal flows may occur among sub-systems as well. In this section, we propose to abstract the relationship between the various flows (*energy-energy*, *signal-energy*, and *signal-signal*), either in a single sub-system, or across various sub-systems. This will be achieved by using the Conditional Generative Adversarial Model (CGAN). We propose to model the system behavior using the relationship between the various flows as shown in Figure 5.2. Based on the particular system design of CPPS,

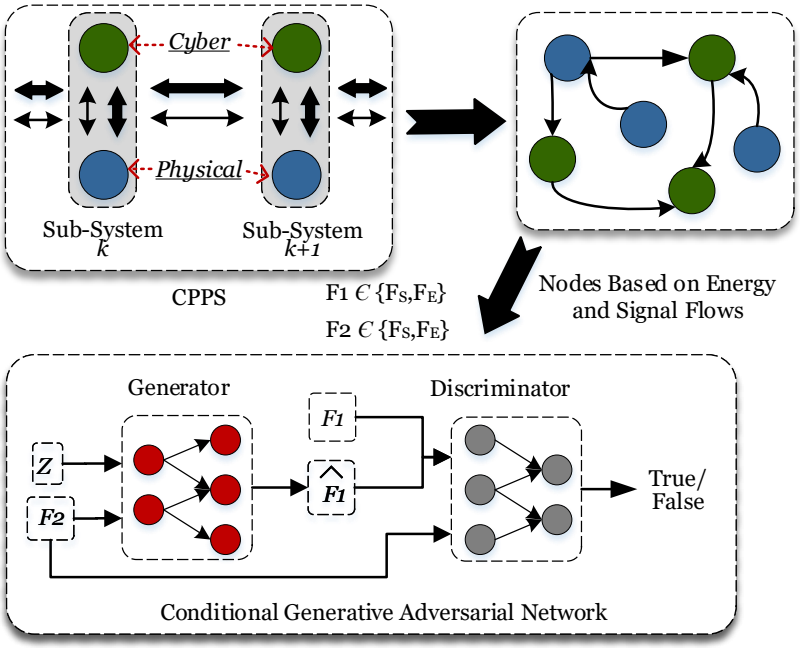



Figure 5.2: CGAN based modeling of CPPS

we can acquire all the signal and energy flows during design time. One can then construct a graph where the various nodes represent the cyber and physical domain components, and the edges represent the signal and energy flow between them (see Figure 5.3).

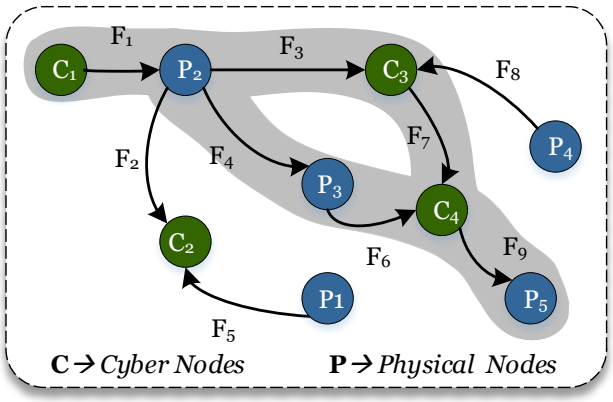



Figure 5.3: Decomposition of CPPS in terms of components and flows

The graph created in Figure 5.3 is then used to list all the flows $F = \{F_1, F_2, \ldots, F_n\}$ with $F \in \{F_S, F_E\}$ that are available in the CPPS system. From this, we extract the flow pairs $(F_i, F_j)$. Each pair is then supplied to the CGAN to model $Pr(F_i|F_j = f_j)$ or $Pr(F_j|F_i = f_i)$. We can infer that a high value of conditional probability indicates a strong relationship between two flows (given knowledge of one of the flows). Based on these distributions, we

can therefore analyze the relationship between various flows from a security perspective. For example, using $Pr(F_1|F_9)$ (see Figure 5.3), one can answer questions similar to the following: Is data in $F_1$ (cyber domain) being leaked from $F_9$ (physical domain)? Can $F_9$ be used to monitor any attacks in the integrity of the flow path from node $C_1$ to $P_5$? Various other metrics may also be created using the conditional probability values (e.g., mutual information metrics of side channel attacks).

**In summary, the proposed CGAN-based security model provides a theoretical foundation to enable a system-level methodology for the design and analysis of CPPS against cross-domain attacks. To the best of our knowledge, this is the first effort towards building a security analysis methodology for CPPS based on CGAN.**

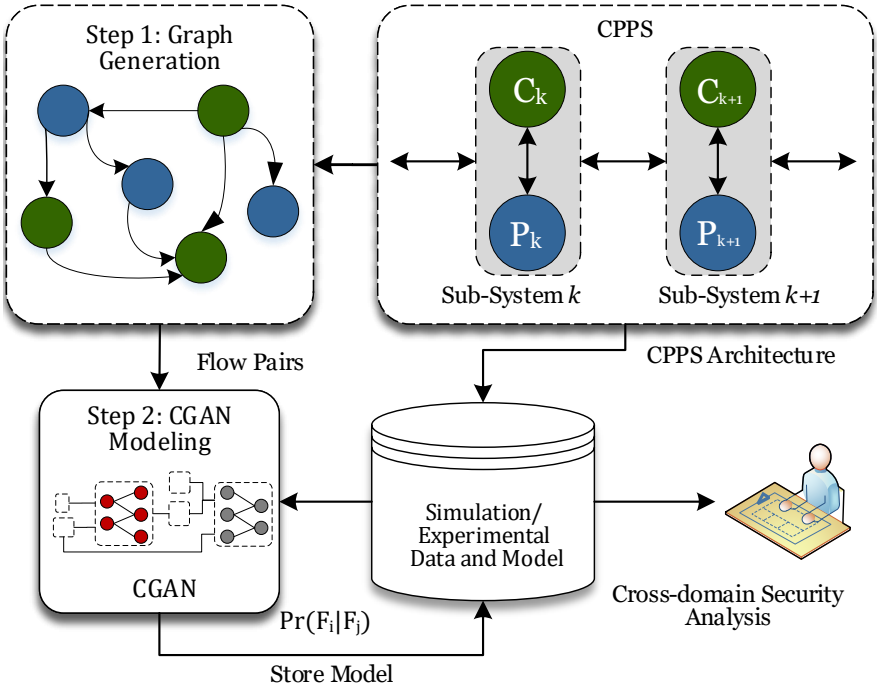



Figure 5.4: Automatic model generation method

## 5.3 CGAN Model Generation

To address the need for a design-time security analysis tool for CPPS, we propose to use a two-step method which generates the proposed CGAN-based security model from a given CPPS model, as shown in Figure 5.4. The two steps include 1) Graph Generation and 2) CGAN-based Security Model Generation. In order to develop the graph from the existing CPPS, the graph generation algorithm takes the design time CPPS architecture as input with information about all the sub-systems $\{Sub_1, Sub_2, \ldots, Sub_n\}$, cyber $C$ and physical $P$ domain components with each sub-system, and the corresponding energy flow $F_E$ and signal flow $F_S$ data among each sub-system.

**Step 1:** *CPPS graph generation based on signal and energy flows.* First, it is necessary to have a list of all the signal and energy flows between cyber and physical components and also between sub-systems. The design-time CPPS architecture specifications will consist of this information and we will convert it into a graph-based model. A graph would allow us to select the energy and signal flow pairs necessary to create the CGAN model in the next step. We will denote the corresponding graph generated in this step as $G_{CPPS}$ consisting of nodes made of cyber and physical components.

Building the graph, $G_{CPPS}$, given the design-time architecture of the CPPS, is straightforward and presented in Algorithm 8. After building $G_{CPPS}$, in Lines 12 to 19 we reduce the graph to find only relevant pairs that can be modeled using CGAN.

**Step 2:** *CGAN model generation.* We assume that the given $G_{CPPS}$ is a black box with only input/output flows. There may exist historical data of the CPPS that we can directly use (e.g., testing/runtime data of a system). The $Pr(F_1|F_2)$ learned by the CGAN relies on the training data. In CPPS, this training data is bound by either the architecture or the production capability. Hence, during design-time security analysis, a large amount of training data is required. This may be a drawback of the CGAN-based modeling, but this problem

**Algorithm 8:** CPPS graph and flow pairs generation

**Input:** CPPS Architecture Data: $Sub, C, FP, F_S, F_E$
**Input:** Historical Data: $Data$
**Output:** Flow Pair List: $FP_T$

```
1  Initialize G_CPPS with V nodes and its adjacency list, E
2  Initialize FP_F as flow pairs (F_i, F_j) list
3  Initialize FP_T as flow pairs (F_i, F_j) list
4  Remove all the feedback loops so that signal and energy flows are directed
5  foreach Subsystem S ∈ Sub do
6      Node list Q=(Add all P_i ∈ S and C_i ∈ S)
7  foreach Node v ∈ Q do
8      Add v in G_CPPS
9      foreach Node u ∈ Q \ v do
10         if F_S or F_E exists between v and u then
11             Add u into adjacency list of v, E[v]
12 foreach Flow F_1i ∈ E do
13     foreach Flow F_2j ∈ (E \ F_1i) do
14         if head of F_2j is reachable from tail of F_1i using DFS then
15             FP_i,j = (F_1, F_2)
16             FP_F = FP_F ∪ FP_i,j
17 foreach FP_i,j = (F_1i, F_2j) ∈ FP_F do
18     if FP_i,j ∈ Data then
19         FP_T = FP_T ∪ (FP_i,j)
20 return FP_T
```

persists in other data collection methods as well. In CPPS, there are some limitations to the access of the high-level data which needs to be protected, such as product specification, due to the mechanical structure of the sub-systems. This allows us to gather training data within the specified bound, and estimate $Pr(F_1|F_2)$ using our model. The amount of data given for training can also be modified according to the attacker capability or attacks detection models resources, such as sub-system type, money or time. Algorithm 9 is used to train the CGAN. The algorithm takes the flow pair list $FP_T$ generated in Algorithm 8 and the data corresponding to the flows in the CPPS.

For each of the flow pairs available in $FP_T$, Lines 1 to 10, which are based on [121, 122], iteratively take sample flows from $F_i$ and $F_j$, and train $D$ by stochastic gradient ascent (Line

**Algorithm 9:** CGAN model generation and storage for security analysis

**Input:** Flow Pairs: $FP_T$
**Input:** Historical Data: $Data$
**Input:** CGAN Training Parameters: Batch Size $n$, Step Size $k$, Number of Iterations $Iter$
**Output:** Generator, Discriminator: $G$, $D$

1 **foreach** $FP_j \in FP_T$ **do**
2  Extract flows corresponding to $(F_1, F_2)$ of $FP_j \in Data$
3  **for** *Iter steps* **do**
4  **for** $k$ *steps* **do**
5  Acquire $n$ mini-batch noise samples $\{z_1, z_2, \ldots, z_n\}$ from $Pr(Z)$
6  Acquire $n$ mini-batch noise samples $\{f_{1_1}, f_{1_2}, \ldots, f_{1_n}\}$ from $Pr_{data}(F_1)$
7  Corresponding to $\{f_{1_1}, f_{1_2}, \ldots, f_{1_n}\}$, acquire $\{f_{2_1}, f_{2_2}, \ldots, f_{2_n}\}$ from $Pr_{data}(F_2)$
8  Update $D$, by ascending its stochastic gradient: $\nabla_{\theta_d} \frac{1}{n} \sum_{i=1}^{n} [logD(f_{1_i}|f_{2_i}) + log(1 - D(G(z_i|f_{2_i}))]$
9  Acquire $n$ mini-batch noise samples $\{z_1, z_2, \ldots, z_n\}$ from $Pr(Z)$ and the $\{f_{2_1}, f_{2_2}, \ldots, f_{2_n}\}$ used in the line 17
10  Update $G$, by descending its stochastic gradient: $\nabla_{\theta_g} \frac{1}{n} \sum_{i=1}^{n} [log(1 - D(G(z_i|f_{2_i}))]$
11 **return** $G$ that estimates $Pr(F_i|F_j)$, and $D$

8), and train $G$ by stochastic gradient descent (Line 10). The number of steps and the iterations to be performed depends on the assumptions about the attacker and can be easily modified accordingly. At the end, $G$ learned for each flow pair is returned and stored. This model effectively will learn the conditional probability $Pr(F_i|F_j)$ [121, 122].

The convergence of $G$ to the historical data has been extensively proven in [121, 122]. Then based on the assumption of how much actual data an attacker can acquire, or how much data an attack detection model can acquire, $Pr(F_i|F_j)$ can be estimated accordingly by adjusting $D$ in the training phase.

## 5.4 Case Study and Analysis

As a case study, we provide the CGAN-based modeling for security analysis of a sub-system of a CPPS. The CPPS sub-system selected is a fused modeling deposition-based additive manufacturing system, also known as a 3D printer. Additive manufacturing has been predicted to be one of the enabling technologies for the next generation of CPPS due to its rapid prototyping and distributed manufacturing capability. However, many emerging threats to this CPPS subsystem has also been highlighted [103, 123]. Hence, we will demonstrate how some security analysis may be performed in this sub-system using the proposed CGAN-based modeling.


(a) Ultimaker 3D Printer (b) Acoustic Sensor

Figure 5.5: Additive manufacturing as a sub-system for security analysis

The state-of-the-art Cartesian 3D printer model consists of four stepper motors as shown in Figure 6.6. Three of them are used to provide printer nozzle movement in the X, Y, and Z directions, while the fourth motor is used to extrude the filament while printing. The speed and direction of all the stepper motors are controlled by cyber domain instructions written with G-code, a programming language widely used in industrial systems to tell a tool what to do and how to do it, along with M-code, auxiliary commands for miscellaneous machine functions. Our experimental setup is enclosed in a makeshift anechoic chamber to isolate the

noise from the environment and other components in the lab. A contact microphone (C411 L) is attached to the back of the 3D printer to acquire the acoustic energy flow dissipated to the environment.

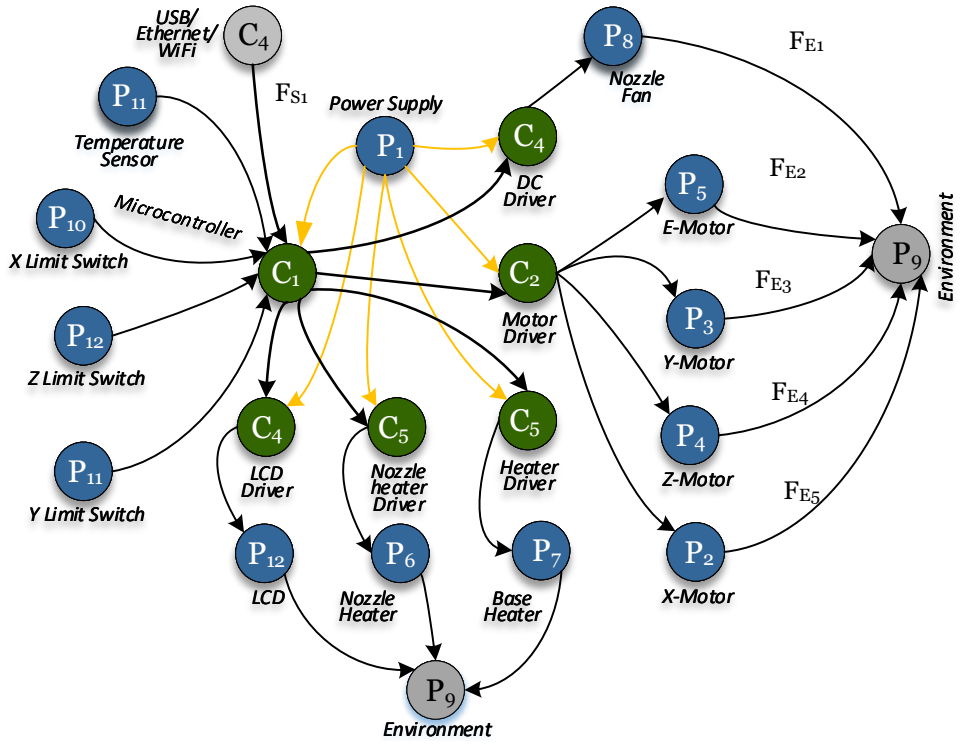



Figure 5.6: Graph generation for cyber-physical additive manufacturing system

### 5.4.1 $G_{CPPS}$ Generation

Using Algorithm 8, the additive manufacturing CPPS can be decomposed to its corresponding energy and signal flows in the form of a graph, $G_{CPPS}$ (see Figure 5.6). One may notice that vertices $C_4$ and $P_9$ are not components of the 3D printer. $C_4$ represents the external signal flows from other sub-systems into the 3D printer. The node $P_9$, on the other hand, represents the physical environment. Various energy flows that are either intentional or unintentional passing to the environment are encompassed by the edges going towards the node $P_9$. From this graph, flow pairs $P_T$ is extracted using Algorithm 8.

### 5.4.2 Experimental Data Collection

In our experiment, we have only selected cross-domain flow pairs for security analysis. Specifically, we analyze the acoustic/vibration energy in the physical domain and the G/M-code instructions passed to the 3D printer subsystem from an external node. The G/M-code instructions are the signal flows entering the sub-system from node $C_4$. In our analysis, we have monitored the energy flows between nodes $P_2$, $P_3$, $P_4$, $P_5$, $P_8$ and the node $P_9$. For ease of training the CGAN, we convert the time-domain acoustic energy flows values into frequency domain values using continuous-wavelet transforms, which preserves the high-frequency resolution in time-domain as well. We obtain a non-uniformly distributed 100 bins $Freq = [freq_1, freq_2, \ldots, freq_{100}]$ between 50 and 5000 Hz (this range may be changed for further security analysis purposes). In this case-study analysis, for simplicity, we extract G/M-codes from 3D objects that only move one stepper motor at a time. The G/M code is one-hot encoded based on presence of instructions that run stepper motors X ($[1, 0, 0]$), Y($[0, 1, 0]$) and Z($[0, 0, 1]$), respectively. This encoding is done based on G/M-codes $G_t$ and $G_{t-1}$. For example, if $G_{t-1}$ is $[G_1\ F1200\ X5\ Y5\ Z5]$ and $G_t$ is $[G_1\ F1200\ X10\ Y5\ Z5]$ then encoding for $G_t$ will be $Cond = [1, 0, 0]$ as only stepper motor X will run. For more thorough security analysis, the one-hot encoding can be extended to consider a combination of signal and energy flows. For example, for three physical components and their combination, the one-hot encoding can be of size $2^3 = 8$.

### 5.4.3 CGAN Modeling

Using the one-hot encoding obtained in Section 9.5 as conditions $Cond$, we have trained CGAN to estimate $Pr(Freq|Cond)$. In doing so, we can derive a trained conditional density function to find the probability of a particular value of frequency component $f_{req}$ given which stepper motor is running. These conditions are generated from the signal flows coming from

node $C_4$, and frequencies extracted from the acoustic energy flows going to the environment node $P_9$ from the various nodes. Hence, the conditional density function estimates the relation between the signal flow from node $C_4$ to $C_1$ and energy flows from nodes $P_2$, $P_3$, $P_4$, $P_5$, $P_8$ to the node $P_9$.

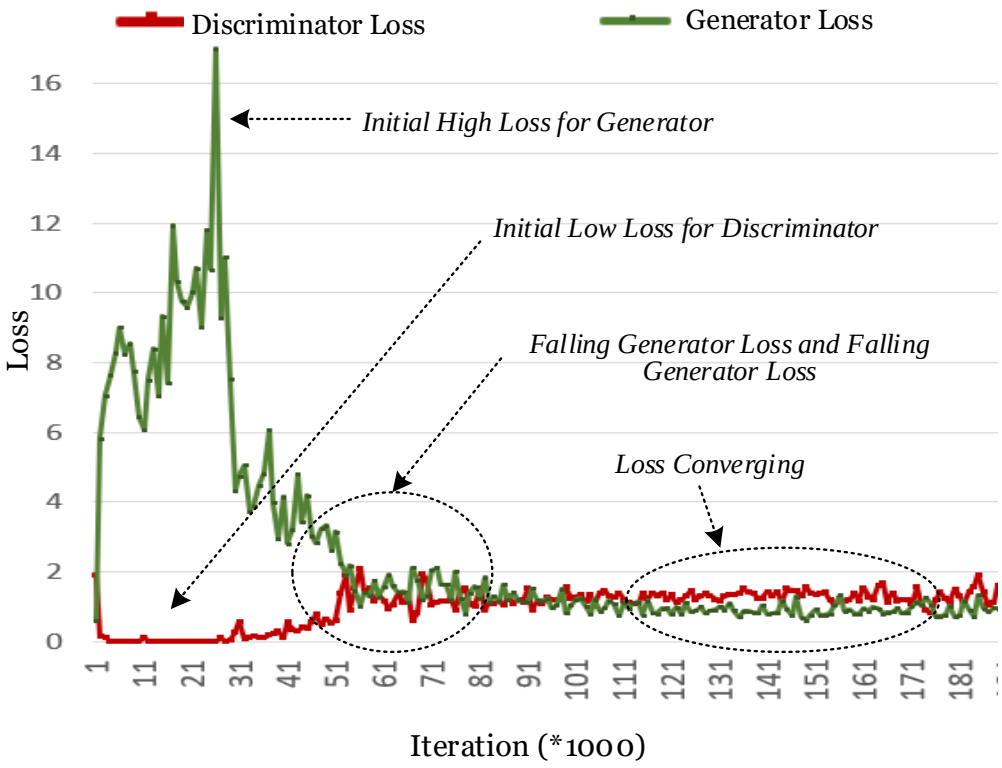



Figure 5.7: Training results for the CGAN

Figure 5.7 shows the training results for the CGAN. On the *X-axis*, the iteration number is increasing. With the increasing iteration, however, the more signal and energy pair data are also incorporated. We can observe that initially, $G$'s loss is high, whereas $D$'s loss is low. However, over more iterations and data, the $G$'s loss decreases, making it difficult for $D$ to know whether the data generated is real or fake, and hence increasing the loss of $D$.

### 5.4.4 Security Analysis Results

In our experiment we will demonstrate how the CGAN modeling may be used for security analysis concerned with *confidentiality* breach through the side-channels, and design of *integrity* and *availability* attack detection on the physical components of the 3D printer subsystem using the physical domain (or the same side-channels). Both of these analyses have one thing common, the conditional relationship between the energy flows given the signal flows. In this section, we will show how the CGAN model (Discriminator $D$, Generator $G$, and Noise $Z$) can be used to check this conditional relation for security analysis. The preliminary algorithm used for security analysis shown in Algorithm 10 (this may be changed for more complex signal flow analysis but can still use the same CGAN).

Algorithm 10 computes security metrics based on the average likelihood of test samples. If a test sample $x$ has a high likelihood for a specific conditional distribution $Pr(x|Cond_i)$ and its actual label is $Cond_i$ then it means that there is a strong relationship between $Cond_i$ and the $x$. This type of metric can be used to evaluate the system's confidentiality vulnerability level or perhaps to detect an integrity or availability attack. For each condition label $Cond_i$, we first generate samples $X_G$ with $G(Z|Cond_i)$. Then for a given Parzen window size $h$, and current feature index $FtIdx$, we create an estimated conditional distribution $FtDistr = Pr(X_G^{FtIdx}|Cond_i)$ via the Parzen Gaussian Window method (Line 10). For each frequency feature, we derive the corresponding test samples from our test batch $X_{test}$ (Line 8), and for each test sample, we update two metrics based on the likelihood.

The first metric $CorLike$ is the total likeliness that this test sample belongs to $FtDistr$ and the other $IncorLike$ refers to the total likeliness that it does not belong to $FtDistr$. We then average the two metrics according to the number of test samples per feature (Lines 19-20). In the outermost loop based on the conditions, we update two sets $AvgCorLikes$ and $AvgIncLikes$, with the corresponding sets of averaged metric values created in the inner

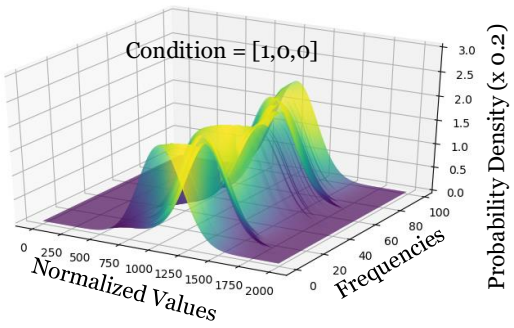
**Algorithm 10:** Algorithm for the security analysis

**Input:** $D$,$G$, $Z$,$Cond$, $GSize$, $X_{test} \in \mathbb{R}^{\mathrm{LxM}}$
**Input:** Frequency Feature Indices: $FtIndices \in \mathbb{R}^{\mathrm{K}}$
**Input:** Parzen Window Width: $h$
**Output:** Average Likelihood Metric Sets: $AvgCorLikes$, $AvgIncLikes$

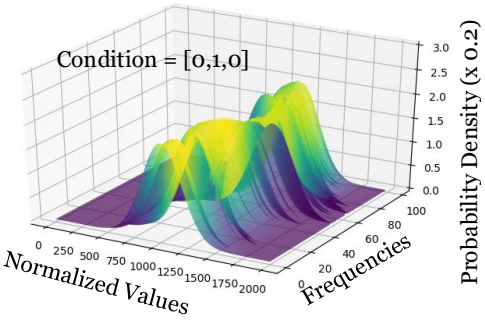
Initialize $AvgCorLikes$ as $\mathbb{R}^{\mathrm{NxK}}$ matrix
Initialize $AvgIncLikes$ as $\mathbb{R}^{\mathrm{NxK}}$ matrix
**foreach** $C_i \in Cond$ **do**
    $CorLike = 0$, $CorNum = 0$
    $IncLike = 0$, $IncNum = 0$
    **foreach** $FtIdx \in FtIndices$ **do**
        $X_G$ = generated $GSize$ samples from $G(Z|C_i)$
        **foreach** $X_{test}^{l,FtIdx}$, *where* $l \in [1, \ldots, L]$ **do**
            $FtDistr = ParzenWindowDensity$
            $(X_G^{FtIdx},\ 'Gaussian',\ h,\ )$
            $LogLike = FtDistr.score(X_{test}^{l,FtIdx})$
            $Like = exp(LogLike) * h$
            **if** $Label(X_{test}^{l,FtIdx}) == Condi$ **then**
                $CorLike$ += $Like$
                $CorNum$ += 1
            **else**
                $IncLike$ += $Like$
                $IncNum$ += 1
        $CurrAvgCorLikes$=$CurrAvgCorLikes \cup \{CorLike\ /\ CorNum\}$
        $CurrAvgIncLikes$=$CurrAvgIncLikes \cup \{IncLike\ /\ IncNum\}$
    $AvgCorLikes[i] = CurrAvgCorLikes$
    $AvgIncLike[i] = CurrAvgIncLikes$
    Reset $CurrAvgCorLikes$ and $CurrAvgIncLikes$
**return** $AvgCorLikes$, $AvgIncLikes$

loops. Higher values in the first set mean that our model has learned a better relationship between data (e.g., features) and their correct conditions (e.g., running motors). However, higher values in the second set mean that our model has learned unexpected and potentially unrealistic relationships between data and other conditions.

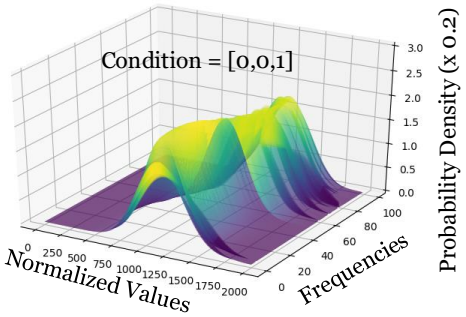
The conditional density functions estimated by the CGAN is shown in Figure 5.8. The frequency magnitudes for the $Freq = [freq_1, freq_2, \ldots, freq_{100}]$ are scaled between 0 and 1. The parzen window width $h$ value used for the Gaussian kernel density estimation is 0.2.




(a) Conditional Probability for Acoustic Emissions for Stepper Motor X




(b) Conditional Probability for Acoustic Emissions for Stepper Motor Y




(c) Conditional Probability for Acoustic Emissions for Stepper Motor Z

Figure 5.8: Conditional probability distribution for the acoustic energy signal (h=0.2) Hence, the actual probability of the frequency values is obtained by multiplying it by 0.2.

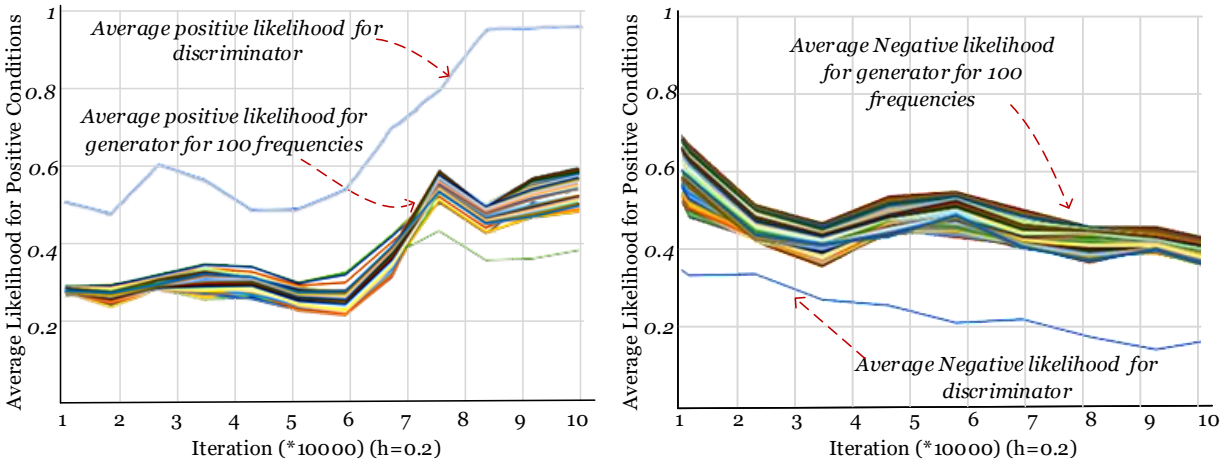



Figure 5.9: Average correct and incorrect likelihood values for the conditions

In Figure 5.9, the average correct and incorrect likelihood values are presented for the condition ($[1, 0, 0]$). As can be seen, over increasing iterations, the positive likelihood averages improve. This shows that the generator is able to accurately learn the conditional distribu-

tion of the acoustic emissions according to the signal flows.

Table 5.1: Average and incorrect likelihood of acoustic energy flows

| | h=0.2 | | h=0.4 | | h=0.6 | | h=0.8 | | h=1 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Cor** | **Inc** | **Cor** | **Inc** | **Cor** | **Inc** | **Cor** | **Inc** | **Cor** | **Inc** |
| **Cond1** | 0.6000 | 0.2245 | 0.6000 | 0.3247 | 0.6069 | 0.3634 | 0.6293 | 0.3783 | 0.6437 | 0.3856 |
| **Cond2** | 0.5750 | 0.3887 | 0.5750 | 0.3961 | 0.5750 | 0.3974 | 0.5750 | 0.3982 | 0.5532 | 0.3978 |
| **Cond3** | 0.6556 | 0.3876 | 0.6556 | 0.3956 | 0.6556 | 0.3979 | 0.6601 | 0.3983 | 0.6556 | 0.3985 |

Based on the density function estimated using the generator of the CGAN, we have used the security analysis algorithm 10 to calculated the average correct and incorrect likelihoods of the emissions given the three conditions (presence of X, Y or Z motor movement in the G/M-code). Using this, a CPPS designer can estimate if an attacker is able to estimate the G/M-code based on acoustic emissions. For example, in table 5.1, we have presented the average correct and incorrect likelihoods of a single feature in the frequency domain based on the G/M-code related to the X, Y, or stepper motor. The table shows that an attacker can estimate condition 3, which is the presence of the Z-motor movement in the G/M-code, better than the other conditions (presence of X or Y motor movement). Moreover, if a designer needs to create an integrity and availability attack detection model to detect attacks on individual components (X, Y or Z motor) using the side-channels, he/she will be able to estimate the performance of such a model using the CGAN model.

## 5.5 Summary

In this chapter, we have presented GAN-Sec, a Conditional Generative Adversarial Network (CGAN) based modeling approach for the security analysis of Cyber-Physical Production Systems (CPPS). To do this, GAN-Sec abstracts the system in terms of its flows and estimates the conditional distribution between them using a CGAN model. We have used GAN-Sec to analyze the security of an additive manufacturing CPPS sub-system. The promising

results indicate that GAN-Sec is applicable in analyzing the cross-domain security of CPPS.

# Chapter 6

# Part II: Dynamic Data-Driven Digital Twin Modeling

## 6.1 Introduction and Related Work

Cyber-physical manufacturing systems have various cyber processes interacting with the physical domain components through a communication network [124]. The processes in the cyber domain can be monitored for each clock cycle of the computing and the communicating component. This allows us to accurately predict the behavior of the cyber domain components. However, in the physical domain, the same is not true. Monitoring physical domain components, and predicting the physical system behavior are faced with a myriad of challenges [125, 20]. One of the major challenges arises due to the fact that the physical domain continuously interacts with the environment, humans, etc., which changes the states of the physical components [126]. Predicting this interaction and the effects of the environment has been one of the major research topics in the manufacturing industry [127, 128, 129, 30]. Being able to infer about this interaction is crucial as it affects the process parameters which

eventually determine the quality of the products.

To overcome the problem of process control, among many others, digitization of the physical domain can be one of the solutions [130, 131, 125, 132]. This real-time digital representation of the physical domain, also known as "The Digital Twin" [133, 20, 134], will make it is easier for manufacturers to accurately predict the future system performance and plan accordingly as well [135, 125, 136]. The future trend will be in using the capability of the Internet of Things in producing a massive amount of data to create a Digital Twin that interacts with the cyber domain [137, 138]. The concept of Digital Twin was first used by NASA, whereby they wanted a Digital Twin replica of the physical system used for the space exploration [139]. Since then, various works have been done to create the Digital Twin of the Physical Twin [133, 140]. Authors in [141] have created Digital Twin, an ultrahigh fidelity model of individual aircraft, to predict the structural integrity and the life of the aircraft structure. A company like IBM are providing sensor systems for creating the Digital Twin of buildings [142]. In fact, technology solutions are hitting the market for enabling the creation of Digital Twin. Ansys [143], released their technology solution for building the twin, with an example for creating the Digital Twin for pumps. Honeywell has recently presented a connected plant concept with a solution to bring all process domain knowledge to create a digital twin in the cloud [144]. Digital Twin technology is also being used for life-cycle engineering asset management in [145]. Authors in [146, 147] provide analysis for research needs and current status for the building blocks of the first generation of the Digital Twin for additive manufacturing (or a 3D printer) system.

### 6.1.1 Research Challenges

In summary, the benefits of building and updating a Digital Twin of a physical system has recently been acknowledged both by researchers and industries. However, the Digital Twin

technology is still in its infancy, and it faces various research challenges such as:

- Understanding what variables in the physical domain can be extracted.
- Knowing how to select the number and position of sensors either during design time or during the usage time (for legacy systems).
- Understanding how a Digital Twin model can be developed given the constraints of resources (such as sensors).
- Figuring out when to update the Digital Twins (as lightweight as possible to meet the resource constraints), to make sure that they can accurately predict the system performance.

### 6.1.2 Our Contributions

In order to solve these research challenges, in this chapter, we provide the following various solutions:

- Present analysis of physical domain signals that can be extracted from cyber-physical manufacturing systems (Section6.5.1).
- Perform feature engineering on signals and data-driven modeling for creating the Digital Twin (Section 6.4.2).
- Use dynamic data-driven application systems for keeping the digital twin up-to-date and lightweight (by re-ranking and selecting the most prominent features for building the Digital Twin) (Section 6.4).

The major contribution of this chapter is in providing a methodology for keeping the Digital Twin alive using the concept of Dynamic Data-Driven Application Systems (DDDAS) [148].

The DDDAS concept is used in influencing the data-driven models (in our case the Digital Twins), by providing dynamic feedback in updating the models based on the real-time data from the physical domain. The key technical challenge of modeling and updating a living Digital Twin of a Physical Twin is solved with the use of dynamically steered sensor data processing (selection and fusion), which uses concepts of DDDAS.

### 6.1.3 Digital Twin Model

The major contribution presented in this chapter is the creation of the Digital Twin based on the dynamic feature selection using side-channel emissions. Before building the Digital Twin, we need to set up an expected outcome or a use case for the Digital Twin. An optimistic version of the Digital Twin would be able to demonstrate and predict every possible physical state of the system. However, to narrow down the scope, we set two objectives for the Digital Twin:

- Based on the status of the physical components, predict the KPIs such as surface texture and dimension of the 3D object that will be printed in the future.
- Be able to know when the digital representation of the physical components is not up-to-date for the KPI predictions.

## 6.2 Digital Twin of Cyber-Physical Additive Manufacturing System

With these narrowed down scopes, let us explain how the Digital Twin can be modeled in a cyber-physical additive manufacturing systems. In Figure 6.1, we present the Digital

Twin representation. It takes into consideration the various environmental interactions and aging phenomena into consideration (represented by random variables $(B_1, B_2, \ldots, B_n)$, the process (random variable $\alpha$) and design parameters (random variable $\beta$), the historical and current analog emissions from the systems (represented by random variable: Acoustics $(A_1)$, Vibration $(A_2)$, Power $(A_3)$, Magnetic $(A_4)$, etc.), and predict the KPI, such as surface texture (represented by a random variable $K_1$) and dimension (represented by a random variable $K_2$).

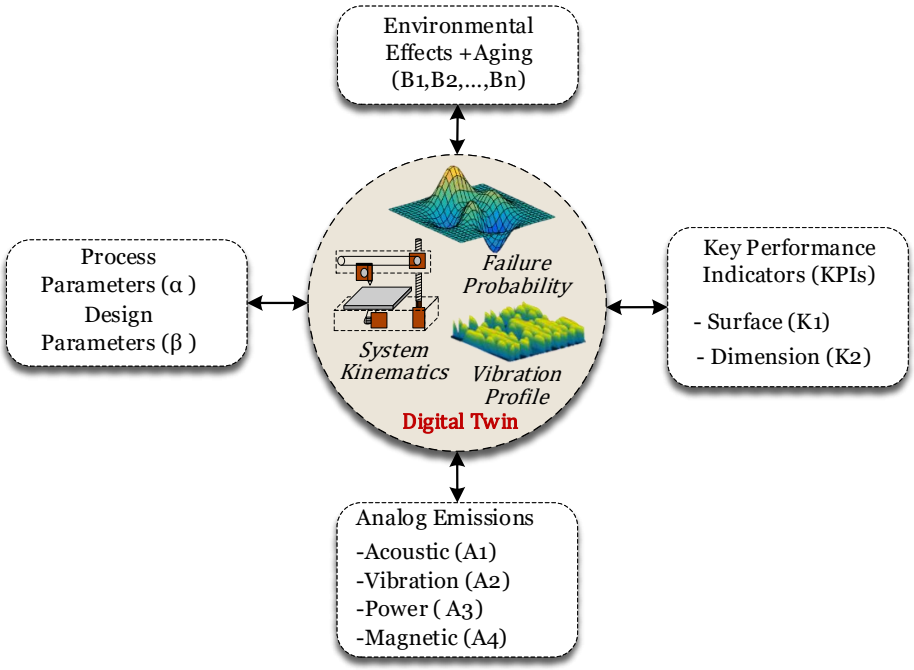



Figure 6.1: Digital twin of cyber-physical additive manufacturing system

In doing so, it explains the effect of environmental parameters $(B_1, B_2, \ldots, B_n)$ on process and design parameters $(\alpha, \beta)$ which in return affects the KPIs. With this backdrop, the Digital Twin can be modeled using a machine learning algorithm that explains the relation between analog emissions, process parameters, environmental factors, and the KPIs.

### 6.2.1 Key Performance Indicators (KPIs)

The two KPIs, used in measuring the performance of the additive manufacturing systems are as follows:

#### Surface Texture ($K_1$)

The quality of the surface texture of the printed object is one of the most sensitive performance indicators that is affected by different design and process parameters. For instance, a slightly higher temperature of the base plate or extruder can cause distortion on the surface by affecting the property of the filament being deposited. Moreover, the slice thickness of the 3D object and the road width of the segment being printed on the $XY$-plane also cause the surface texture to vary in quality. For quantifying the surface quality, we explored an innovative metric called *dispersion of directionality* ($K_1$). It is a heuristic metric calculated by performing various image processing algorithms on the surface image. *We would like to emphasize that other standard metric can be used with our methodology (as the methodology is independent of the metric used to measure the KPI), and in fact may improve the accuracy of the Digital Twin.* The steps to calculate $K_1$ are as follows:


(a) Measurement enclosure to prevent external light from entering

(b) Internal view of the enclosure

Figure 6.2: Constant lighting environment for measuring the quality of surface texture

1. **Create a constant lighting environment:** As shown in Figure 6.2, we use an empty box with a small opening for fitting a DSLR camera on the top (this camera can be replaced with a low-cost camera as well). This enclosure prevents the time-varying external light from entering and affecting the surface texture measurement. To maintain constant lighting, we used two similar light sources, around 10 *lumen* of luminous flux, placed directly opposite to each other to provide homogeneous lighting on the surface of the 3D object. Moreover, guidelines are drawn to always place the 3D object on the same location for the surface texture measurement.

2. **Remove the background:** Since the 3D objects have a distinct color (green) compared to the background (brown), we first transform the image taken by the camera from the RGB to the Lab color space [149]. Lab color space consists of three dimensions: $L$ for the lightness of the image, and $a$ and $b$ for the color opponents green-red and blue-yellow, respectively. After this transform, we choose the value of $a$ which eliminates the shadows and the brown colored background. Then, we perform a constant threshold to create a mask, which matches the green colored object. This mask is then applied to the image to eliminate the background.

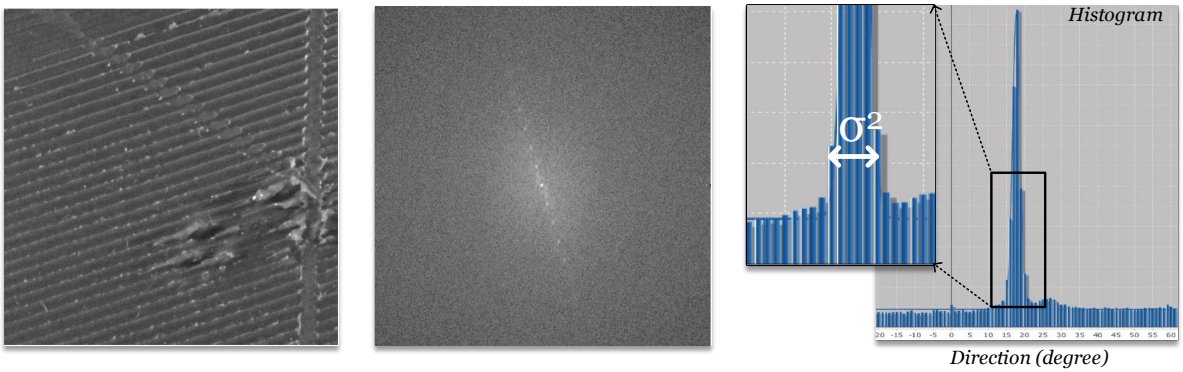
3. **Surface division:** The image is divided into either 4 by 4 or 16 by 16 equal parts for aiding the process of mapping the surface texture $K_1$ value to its corresponding analog emissions.

4. **2D Discrete Fourier Transform:** The discrete transform ($F[k,l]$) is calculated for the image of the 3D object's surface ($f[m,n]$) (see Figure 6.3 (b)), using Equation 6.1, where, $M$ and $N$ are the height and the width of the image calculated in the step 3. The maximum values for $k$ and $l$ are $M$ and $N$ as well.

$$F[k,l] = \frac{1}{MN} \sum_{m=0}^{M-1} f[m,n] e^{-j2\pi(\frac{k}{M}m + \frac{l}{N}n)} \tag{6.1}$$

5. **Directionality histogram calculation:** Based on the value of $F[k, l]$, calculated in the previous step, a directionaility histogram is calculated using the approach mentioned in [150] for faster calculations (see second image in Figure 6.3).

6. **Fit Normal Distribution:** A normal curve is then fitted to the histogram (see Figure 6.3 (c)) obtained in the previous step, and the corresponding distribution parameters are calculated. Out of these parameters, standard deviation $\sigma$ is the dispersion metric we use to measure the directionality and hence the surface texture $K_1$.




(a) Surface Texture Image (b) Discrete Fourier transform of the image (c ) Distribution parameter estimation

Figure 6.3: Last three steps (left to right) for measuring the quality of surface texture

The heuristic dispersion metric used for surface texture measurement $K_1$ does not have a linear relation with a number of the lines in the image or the amount of disorientation in the image. In order to prove the effectiveness of the dispersion metric for surface texture measurement, we asked two people to sort 28 surface images of the 3D objects printed by the fused deposition modeling based additive manufacturing system. These surface textures were varied randomly with various level of surface quality. This sorting was then compared to the sorting order created using the values of *dispersion of directionality* calculated using our algorithm. The result was a similar sorting order. However, due to the subjective nature of quality perception, further work is needed to compare the result of the standard metric and the *dispersion of directionality* for surface texture measurement. Nonetheless, in this work, *dispersion of directionality* is used as a KPI for surface texture measurement, due to

its ability to represent the varying road thickness of the filament deposited on the $XY$-plane, presence of non-directional surface pattern, etc. Moreover, the use of standard metric will only make the Digital Twin modeled using our approach more accurate.

**Dimension ($K_2$)**

The dimensional accuracy is affected by various process and design parameters. The process parameters that affect $K_2$ are *build environment temperature*, *filament feed-rate*, *nozzle temperature*, etc. Whereas, the design parameters that affect the $K_2$ are *road width*, *slice thickness*, *air gap*, *build orientation*, *raster patterns*, etc. Various environmental factors such as *room temperature*, *humidity*, *vibrations*, etc. can influence these process and design parameters, which in return affect $K_2$. Since the resolution of the 3D printer used in our experiment is about 12.5 *micron*, 12.5 *micron*, and 2.5 *micron* in X, Y, and Z directions, respectively, micrometer is used to measure the dimension of the 3D objects.

## 6.3 Keeping Digital Twin Updated

The Digital Twin model explains the relationship between KPIs, analog emissions, environmental factors, process parameters, and design parameters. In order to keep the Digital Twin updated, these models will have to be continuously updated based on the run time difference ($\delta$) between the predicted KPIs and the measured KPIs (as shown in Figure 6.4). This $\delta$ will give the Digital Twin an intuition of liveliness of the Digital Twin and enable it to have the cognitive ability. If the error between the predicted and the real KPI value is large then the Digital Twin will be assumed to be outdated. This in return will trigger feedback to the dynamic feature selection algorithm to re-rank and fuse the feature values to measure the relation between the most recent texture value and the analog emission values

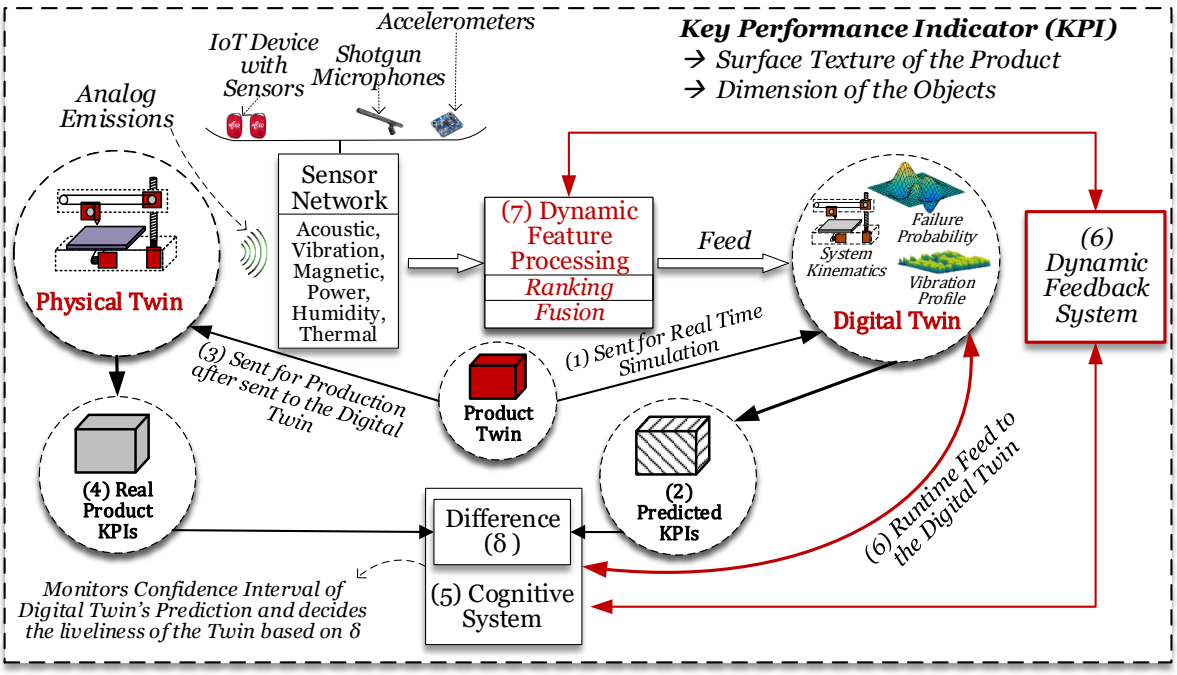



Figure 6.4: Dynamic data-driven application systems enabled digital twin modeling

calculated, while only selecting the features that are necessary (thus keeping the Digital Twin model lightweight). The steps for updating the Digital Twin (as shown in Figure 6.4) are as follows:

1. The product Digital Twin (3D object) is given to the Digital Twin of the 3D printer.
2. Digital Twin of the 3D printer predicts the KPIs ($K_1$ and $K_2$).
3. The 3D Object is sent to the physical twin for production only when the predicted KPIs are within a tolerable range.
4. Actual KPIs are measured from the printed 3D Object if it is printed.
5. The cognitive algorithm measures the difference between the actual and predicted KPIs.
6. If the $\delta$ is large, feedback is sent to the Digital Twin and the feature processing algorithms.

7. Dynamic feature processing algorithm re-ranks and fuses the features to get the most up-to-date Digital Twin model.

The feedback to the data-driven Digital Twin model is activated by the analog emission data gathered from the sensors. This data represents the current physical state of the system (for example, mechanical degradation may result in variation of the analog emission pattern).

## 6.4 Building Digital Twin

### 6.4.1 Sensor/ Emission Modality Selection

The 3D printer consists of a set of actuators, mechanical moving parts, heating elements, and a controller board. An actuator under the load consumes energy, vibrates, emits electromagnetic waves, and produces audible sound. The heating elements of the 3D printer, which are embedded inside the nozzle(s) and the base plate, consume electrical energy and convert it into thermal energy. Based on these facts, we decided to monitor the physical domain by acquiring acoustic, electromagnetic, vibration, power, humidity, and temperature data and analyzed them for the amount of information revealed about the cyber domain and physical domain states of the 3D printer. The validation for the analog emission modality selection is experimentally validated through the accuracy of the Digital Twin models.

### 6.4.2 Feature Engineering

After selecting the emissions to be monitored in the physical domain, various features have to be extracted to reduce the size of the raw data collected, and to improve the performance of the machine learning models (or the data-driven models) used to create the Digital Twin.

In this section we will briefly describe the various features extracted in various domains.

### Time Domain

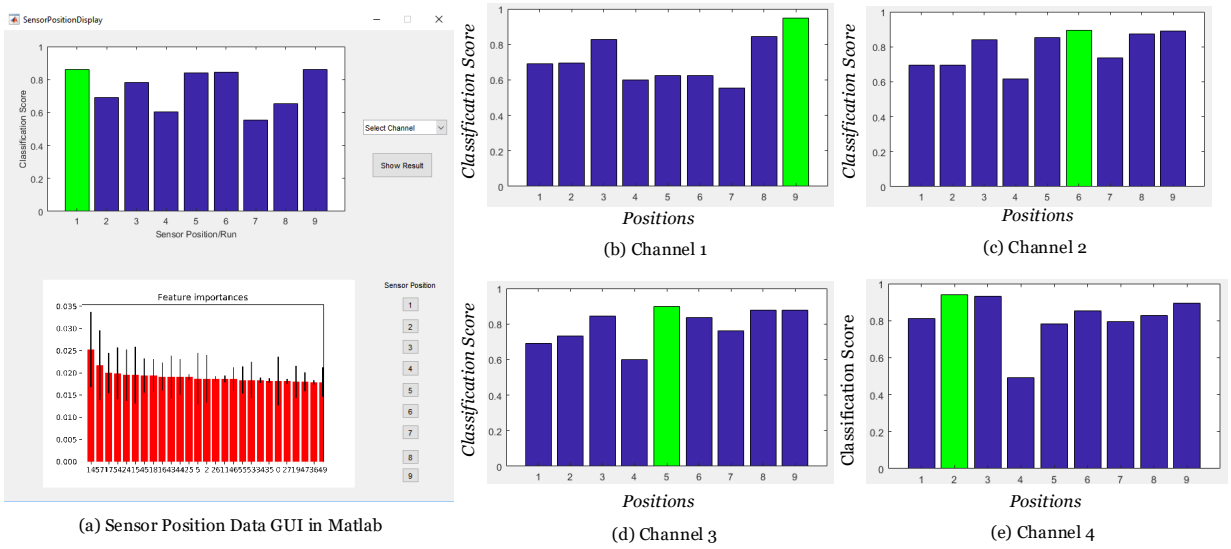
The various features extracted in time domain are *Energy*, *Energy Entropy*, *Mean Amplitude*, *Maximum Amplitude*, *Minimum Amplitude*, *Median Amplitude*, *Mode of Amplitude*, *Peak to Peak features* (*highest peaks*, *peak widths*, *peak prominence*, etc.), *Root Mean Square values*, *Skewness*, *Standard Deviation*, *Zero Crossing Rate*, *Kurtosis*, etc (Total 114). These are extracted for each emission. Each of these features capture various properties of the signal, explaining each one of them is out of scope of this report, and will be left for the future work.

### Frequency Domain

Various analog emissions are monitored in the physical domain. Each of them has a different frequency range and characteristics. To capture all of the characteristics for all the signal, various frequency domain signals are analyzed.

- **Frequency Characteristics and Short Term Fourier Transform (STFT):** The frequency characteristics analyzed are based on the short term windows and various characteristics of the frequency domain such as: *Mean Frequency*, *Median Frequency*, *Signal to Noise Ratio*, *Power Bandwidth*, *Spectral Centroid*, *Spectral Entropy*, *Spectral Flux*, *Spectral Roll Off*, etc (82 in Total).
- **Frequency Characteristics and Continuous Wavelet Transform (CWT):** The challenge with short term Fourier Transform based features is that the tradeoff between time and frequency. The window frame (in the time domain) and the resolution of frequency domain features are highly dependent, and one has to be compromised

for the other. Instead of compromising these we have also analyzed continuous wavelet transform in the frequency domain, and analyzed various characteristics of the transform (in Total 58). Based on the result of the Digital Twin prediction and feature ranking, in the future, the continuous wavelet transform will be used to calculate the discrete wavelet transform with specific approximation and detailed coefficients.




(a) Sensor Position Data GUI in Matlab

(b) Channel 1

(c) Channel 2

(d) Channel 3

(e) Channel 4

Figure 6.5: Classification scores for various sensor position

### 6.4.3 Sensor Positioning

In order for us to determine the position of the sensors selected to acquire the data from the physical domain, we performed profiling of the analog emissions from each modality. For this, a classifier is modeled to classify various cyber domain data (G/M-codes), such as movement in each axis. Then, various location around the 3D printer (9 locations for each modality except power, humidity, and temperature) was selected for measuring the analog emission, and the corresponding classification scores. A Matlab based graphical user interface is created to analyze the classification scores and the corresponding feature ranking scores for analyzing and finalizing the sensor positions. In Figure 6.5, the result of the sensor position GUI and the four channel's classification scores are presented. It can be seen from

the figure that for channel 1, position 9 gives the highest classification accuracy, whereas, for channel 2, channel 3 and channel 4, the position most favorable for higher accuracy are 6, 5, and 2, respectively.

### 6.4.4 Data-Driven Models

For creating the data-driven models of the Digital Twin, we explored various machine learning algorithms such as Gradient Boosted Regressor [151], Decision Tree regressor [152], K nearest Neighbor Regressor [153], AdaBoost Regressor [154], etc. These models were used to model the relationship between the design and process parameters and the analog emissions, and the KPIs.

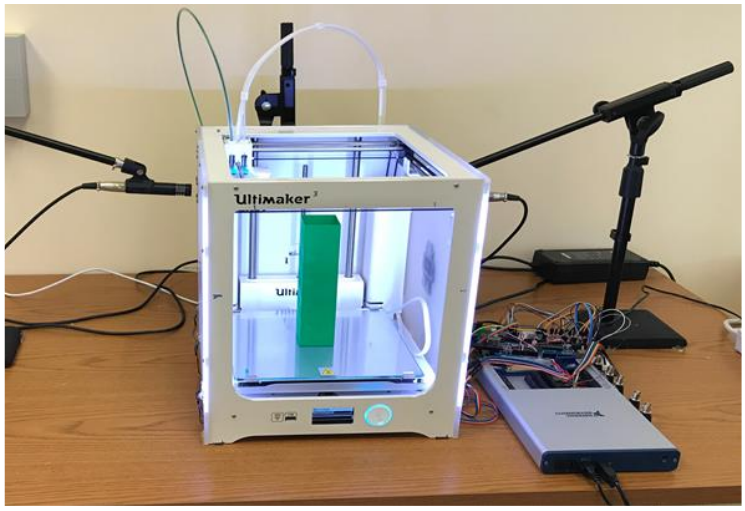



Figure 6.6: Experimental setup for the digital twin modeling using DDDAS

## 6.5 Experimental Setup

The experimental setup for modeling and updating the Digital Twin is shown in Figure 6.6. As a test case, we have selected fused deposition modeling based cyber-physical additive manufacturing systems. The various components of the experiments are explained as follows:

Table 6.1: List of sensors used for monitoring the 3D printer and its surrounding environment

| Quantity | Modality | Sensor Model | Interface | Outputs | Sampling Rate (kS/s) | Sensitivity |
|---|---|---|---|---|---|---|
| 3 | Vibration | ADXL326 | Analog | 3 | 1.6 (X,Y), 0.55 (Z) | 57 mV/g |
| 3 | Magnetic | HMC5883L | Digital (I2C) | 3 | 0.16 | 2 mG |
| 1 | Current | PICO TA018 | Analog | 1 | 20 | 100 mV/Amp |
| 3 | Acoustic | AT2021 | Analog | 1 | 20 | 11.2 mV/Pa |
| 1 | Acoustic/Vibration | AKG C411 III | Analog | 1 | 18 | 2 mV/Pa |
| 1 | Temperature | LM35 | Analog | 1 | 1.5 | 10 mV/°C |
| 1 | Humidity | AM2001 | Analog | 1 | 0.0005 | 0.1%RH |

### 6.5.1 The Test-bed

The test-bed consists of an *Ultimaker 3* 3D printer, a set of sensors with analog and digital interface, a Data Acquisition (DAQ) device, two *Arduino Uno* microcontroller boards coupled with MCP4725 for digital to analog conversion (DAC) purpose, and a personal computer for managing the acquired data. The details of the test-bed are as follows:

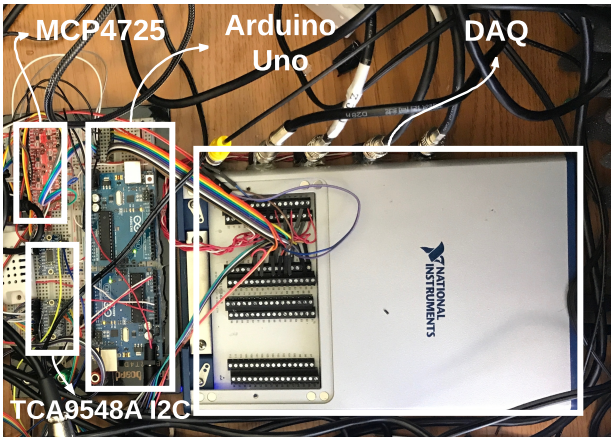



Figure 6.7: Data acquisition setup for the experimental analysis of digital twin models

#### Sensors

In this chapter, as presented in Table 6.1, we monitor the 3D printer and its surrounding environment using a total of 25 sensors. The maximum sampling rate, sensitivity, and cost

are the important factors that determine the type of sensor selected for each modality. We have used three accelerometers for measuring the vibration in the system. These accelerometers have a sampling rate of 1 $kHz$. A contact microphone is used to measure the acoustic noise and the high-frequency vibration from the printer. Three microphones with a sampling rate of 20 $kHz$ are used to measure the acoustic emissions in the audible range from the 3D printer. ZOOM TAC-8 phantom power supply and amplifier are used to condition the acoustic signal before feeding it to the DAQ. The electromagnetic field intensity variation caused by the stepper motors of the 3D printer cannot be captured without using high precision EM sensors. Instead, we use three compass sensors, which are designed to sense the Earth's magnetic field. By using these magnetic sensors, we measure the fluctuation in the magnetic field of Earth caused by the moving metallic parts of the 3D printer. The sampling rate of the compass is around 270 $Hz$. The humidity and the temperature of the room change slowly over time. Moreover, the KPIs do not change drastically within the ±5% fluctuation of humidity, and ±1°C change in the temperature. Hence, any sensor satisfying these properties would be sufficient for monitoring the analog emissions from these modalities.

**Data Acquisition**

*NI USB-6343 OEM* is used for data acquisition (DAQ), see Figure 6.7. It has 32 analog inputs. For DAQ, with the increasing number analog inputs the overall sampling rate decreases. This limits the number of analog signals that can be monitored with a high sampling rate. However, for the 25 analog inputs used in our experiment, the resolution of the DAQ is 16 bits for the data with 20,000 Samples/Second sampling rate. These resolutions are sufficient for the acoustic emissions and surpass the requirements for other emissions. Since the DAQ takes an analog signal, an Arduino board coupled with MCP4725 is used to convert the sensor's digital data to analog form. This is done to synchronize all the 25 channels and maintain coherent sampling and data resolution. The Arduino board reads the data from

the sensor using the I2C interface and sends it to MCP4725 over the I2C for conversion to analog form. This conversion is necessary for the magnetic sensors. There are three magnetic sensors, each measuring magnetic fluctuation in X, Y, and Z direction. This results in a total of nine signals given by the three magnetic sensors. Since the MCP4725 boards share the same I2C addresses, two TCA9548A I2C Multiplexers are used to access them separately. According to our measurements, this setup can convert more than 6×275 digital samples to analog signal every second, which is more than enough for converting the 3×170 samples generated by the compass sensors.

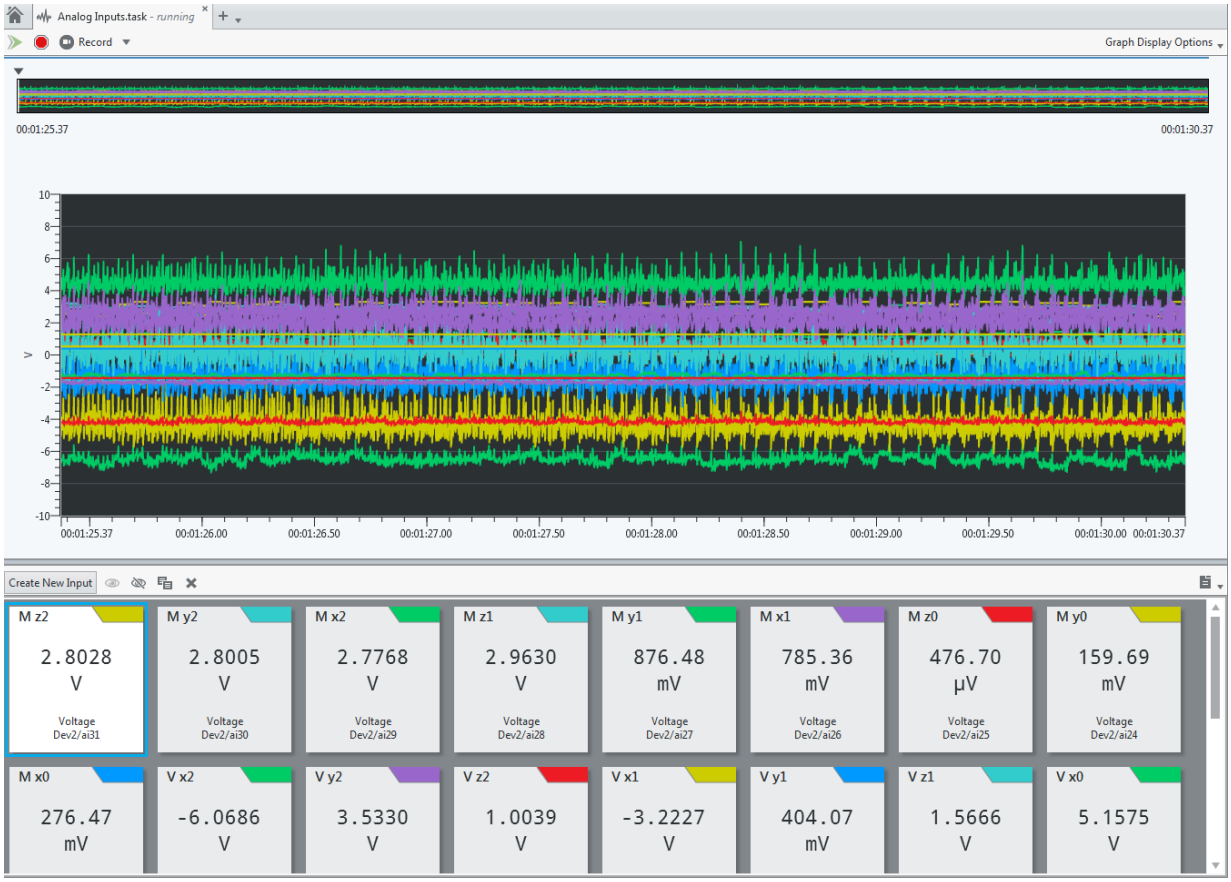



Figure 6.8: A sample snapshot of the data collected from the sensors

### Data Synchronization

The DAQ used in the experiment assures synchronization of all the sensors' data with each other (see Figure 6.8 for a snapshot of data collected from the DAQ). However, the analog signals collected from the sensors should be mapped with G-code for building the Digital Twin. For being able to segment the G-code, the 3D printer firmware is modified to send every G-code (along with the time-stamp with accuracy in the range of milliseconds) right before execution to the host's IP address. The port used for communication is 5000. A Python code running on the host side (the desktop computer) is made to continuously listen at port 5000, waiting for the printer to start printing. Once the host receives the first G-code, it first saves the 3D printer's clock data (which allows us to synchronize the G-code with the DAQ data). It then starts saving the sensors data (in *.tdms* format) sent from the DAQ in chunks of $\sim 55\ MB$.

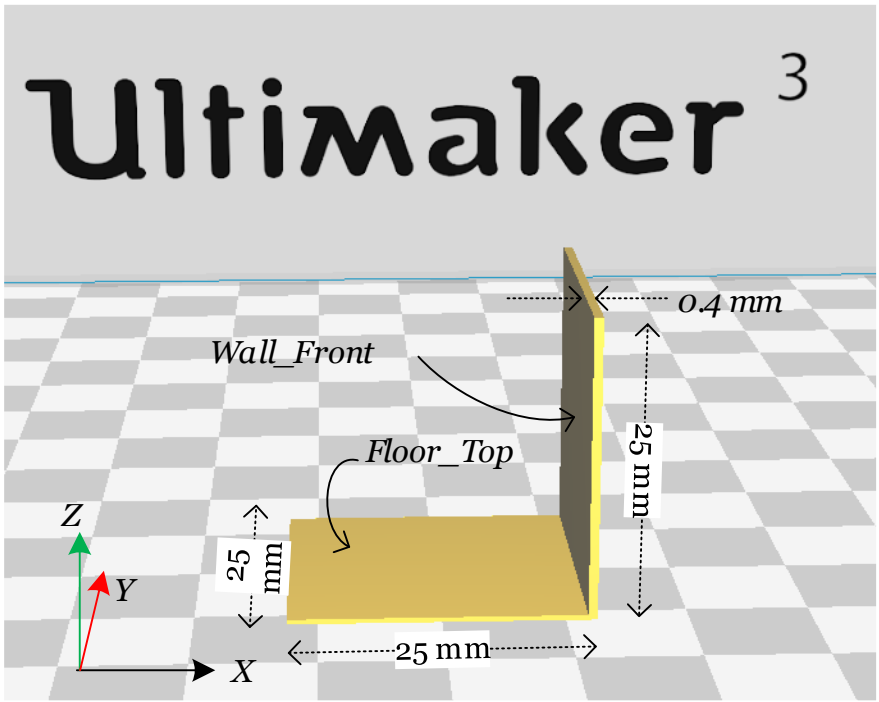



Figure 6.9: Test 3D object for digital twin experiment

### 6.5.2 Test 3D Objects

For modeling and testing the aliveness of the Digital Twin, we have customized a benchmark model (see Figure 6.9). Using this benchmark model, we measure the surface texture ($K_1$) and dimension variation ($K_2$), map these KPIs with the corresponding analog emissions from the side-channels, and build the Digital Twin model. Our benchmark model has four surfaces: top and bottom for the floor and front and back for the wall. Hence, we measure the surface texture of all four surfaces. For dimension, we measure the width and the breadth of the floor, the width and the height of the wall, and the thickness of both the floor and the wall (see Figure 6.9).

Table 6.2: Summary of environmental and aging degradation parameters

| Degradation Parameter | Corresponding $\alpha$ and $\beta$ | Environmental+Aging Effects | effects |
|---|---|---|---|
| $B_1$ | Nozzle Temperature ($\alpha$) | (1) Sensor Malfunction, (2) Extreme environmental temperature, (3) Heating element malfunction degradation | $K_1, K_2$ |
| $B_2$ | Filament Feedrate ($\alpha$) | (1) Slippage, (2) Worn out rollers due to mechanical degradation, (3) Variation of filament thickness | $K_1, K_2$ |
| $B_3$ | Acceleration of Stepper Motors ($\alpha$) | (1) Rust, (2)Vibration due to loose components, (3) Vibration due to mechanical degradation | $K_1, K_2$ |
| $B_4$ | Power Outage ($\alpha$) | (1) Short circuit of electronic components, (2) PCB failure, (3) Power supply failure | $K_1, K_2$ |
| $B_5$ | Printer Alignment ($\alpha$) | (1) Shockwave, (2) Earthquake | $K_1, K_2$ |
| $B_6$ | Humidity ($\alpha$) | (1) Faulty HVAC | $K_1, K_2$ |
| $B_7$ | Slicing Thickness ($\beta$) | (1) Faulty Z-motor, (2) Erosion of translation screw | $K_1, K_2$ |

### 6.5.3 Data Collection

The main objective of the Digital Twin is to be able to predict the KPIs based on the environmental effects and aging. Collecting analog emissions, within a short period of time, that encompasses the interaction between the system and the environment is a challenging task. Moreover, collecting the variation in analog emissions due to aging requires collecting data for a long period of time. In order to carry out the experiment within a short period of time, we have performed the following tasks:

- First of all, a summary of environmental and aging effects and their corresponding impacts on the design and process parameters is listed (presented in Table 6.2).
- Analysis of how each design and process parameter gets affected is performed.
- Then design and process parameters are varied to reflect the impact of environmental and aging effects
- The KPIs, corresponding to the design and process parameter variation are measured.

From Table 6.2, we were able to analyze which design and process parameters get affected by which environmental factors and aging degradation. For the purpose of the experiment, where we want to validate the DDDAS enabled Digital Twin, we selected the degradation parameter ($B_2$) which can be reflected in terms of varying filament flow rate. The flow rate of the system is changed to reflect the effects of the environment and aging. To do this, the flow rate is changed from 20% to 200% of the optimal flow rate of the 3D printer with the step-size of 10%. In the 3D printer, the optimal range for the flow rate lies in the range of 80% to 120%, The flow rate is the process parameter which is calculated using the equation:

$$W * H = A = \frac{Q}{v_{feed}} \tag{6.2}$$

Where $W$ is the width and $H$ is the height of the line-segment being printed on the XY-plane, $Q$ is the constant volumetric flow rate of the material. $Q$ is estimated based on die swelling ration, pressure drop value and buckling pressure of the filament. $v_{feed}$ is the feed velocity of the filament and is calculated as:

$$v_{feed} = \omega_r * R_r \tag{6.3}$$

Where, $\omega_r$ is the angular velocity of the pinch rollers, and $R_r$ is the radius of the pinch rollers. Based on these values, the pressure drop is calculated as follows:

$$P_{motor} = \frac{1}{2} \triangle P * Q \tag{6.4}$$

Where, $P_{motor}$ is the pressure applied by the stepper motors, $\triangle P$ is the pressure drop. Hence, the pressure applied by the motor needs to be maintained for the constant volumetric flow rate. However, this pressure needs to be less than buckling pressure calculated as follows:

$$P_{cr} = \frac{\pi^2 * E * d_f^2}{16 * L_f^2} \tag{6.5}$$

Where $E$ is the elastic modulus of the filament, $d_f$ is the diameter of the filament, and $L_f$ is the length of the filament from the roller to the entrance of the liquefier present in the nozzle. It is evident from these equations that maintaining a constant flow rate depends on various parameters, and environmental or aging factors affecting any of these parameters will change the flow rate, causing changes in the KPIs.

*While creating a Digital Twin in a manufacturing plant, this analysis need not be performed, and data from various analog emissions that have the likelihood of behaving as side-channels can be collected. This data can then be mapped to the KPIs to be able to model a Digital Twin that predicts the KPIs based on the varying process parameters. The variation in the process parameters is due to the environmental effects and degradation due to aging.*

### 6.5.4 Data Segmentation

The foremost task in any data-driven modeling is acquiring the labeled data for training the machine learning algorithms. In our case, we want to be able to map the KPIs with the corresponding $\alpha$ and $\beta$, and the analog emissions. In order to achieve this, we segmented the floor and wall of the test objects into various segments, mapped it to the corresponding G-codes of the object and acquired the timing data necessary to segment the analog emissions.

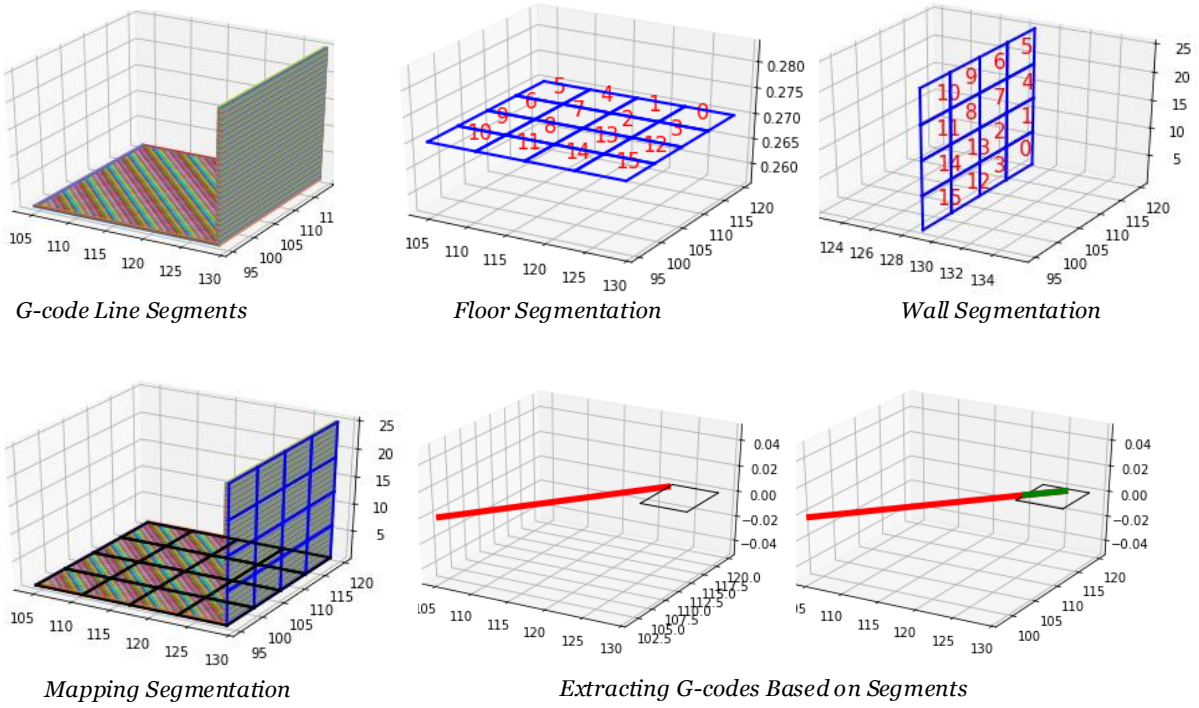



Figure 6.10: Segmentation of test 3D object for digital twin experiment

For experimental purposes, both the wall and floor region are segmented into 4 by 4 matrix segments. After segmenting the data, the various segments are grouped together for the purpose of mapping the corresponding KPIs with analog emissions. For instance, the Floor and Wall segments numbers 3, 7, 8, and 13 (see Figure 6.10), are grouped to measure the surface texture and thickness, as they lie on the center region and have homogeneous surface texture and dimension. The reason for searching homogeneous segments is made clear through Figure 6.12. Due to the custom synchronizing code running inside the 3D-

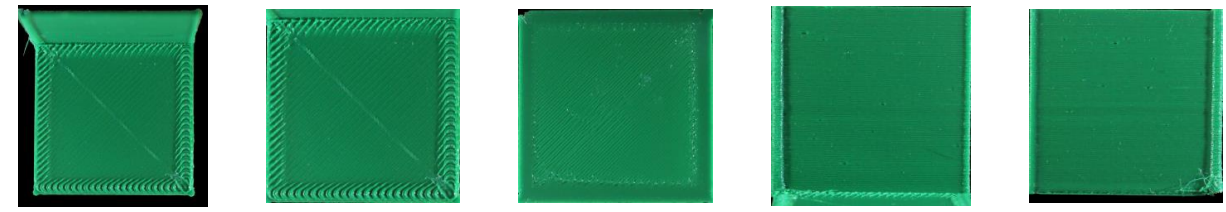
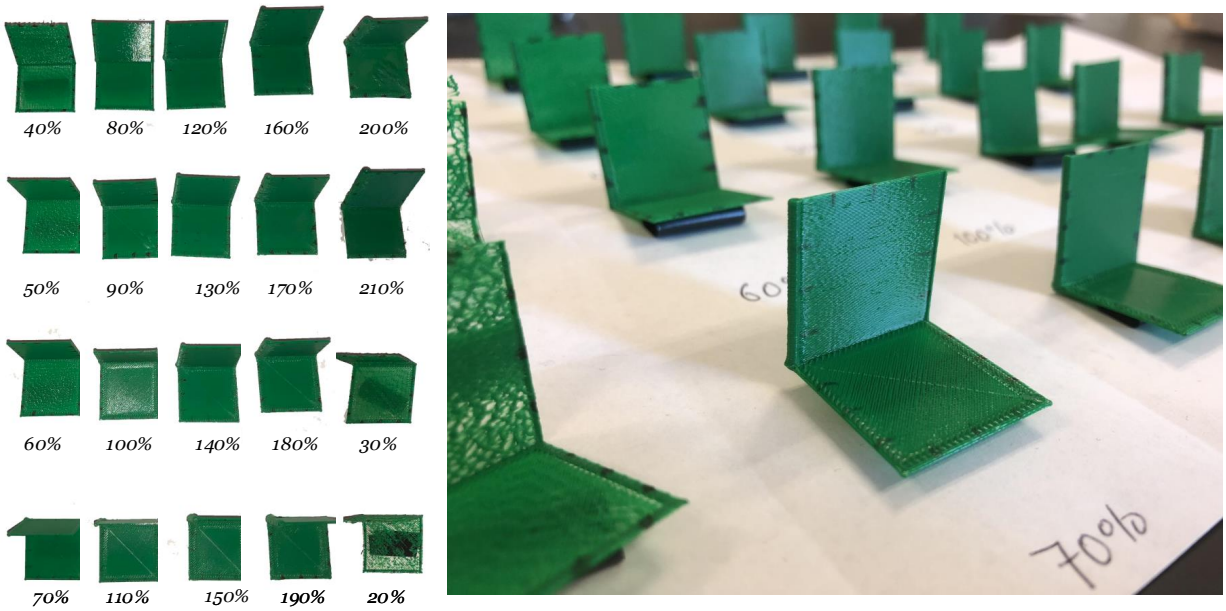



Figure 6.11: Printed test objects

printer firmware, various non-homogeneous changes are created in the printed 3D object. These changes are more prominent on the outline of the object. Hence for the experimental verification purpose, in the work, we have discarded the non-homogeneous outer segments.




(a) Top View

(b) Perspective View

Figure 6.12: Various test objects printed with varying flow-rates

For each of the test objects, the total number of channels from which the data collected is 25. And for each channel, the total number of features extracted is $114 + 82 + 58 = 254$. In our training model, all the channel's features are fused together into a single matrix. Hence, the dimension of training data is $X_{samples \times 25*254} = X_{samples \times 6350}$, where samples represent the total number of observations.

## 6.6 Simulation and Results for Digital Twin Models

### 6.6.1 Digital Twin Models

With the objective of predicting the dimension and surface texture quality, we have created the following Digital Twin models.

- Two models, predicting the thickness of the wall and the floor.
- Four models predicting the surface texture of floor top, floor bottom, wall front, and wall back, respectively.

All of these models are created using gradient boosting based regressors [151, 155]. These are an ensemble of decision tree based regression models. By generating a new tree against the negative gradient of the loss function, these algorithms combine weak learners to control over-fitting. It is chosen due to its robustness against outliers and better predictive power against other regression algorithms. In this section, we will present some accuracy measures of the regression algorithm while training it for optimal values of flow rate (80% to 120%), and slowly incorporating degradation along the positive direction, i.e. $(130\%, 140\%, \ldots, 200\%)$.

### 6.6.2 Aliveness

In order to check the aliveness of the Digital Twin, we devised the experiment as follows:

1. Train the Digital Twin with the optimal values of the flow rates (80% to 120%).
2. Assume an environmental degradation has caused the flow rate to vary to 200%. Predict the KPIs, with the Digital Twin trained with optimal flow rates.

3. Gradually incorporates the flow rates degradation from 130%, 140%, etc., all the way to 190%. This incorporation demonstrates a gradual update of the Digital Twin along with the run-time performance of the 3D printer.

Through this experiment we will be able to measure two things: (1) the predicting capability of the Digital Twin modeled from side-channel emissions, and (2) cognitive capability enabled for DDDAS by checking the aliveness of the Digital Twin. The prediction capability is demonstrated in terms of the accuracy of each of the models. By gradually adding the analog emissions in modeling the Digital Twin, we assume that we are adding the emissions that represent the physical status of the system that is closer to the current state. By doing so, we expect to see improvement in the prediction capability of the Digital Twin and lower $\delta$ value. The lower value of $\delta$ will signify that it is possible to monitor the $\delta$ and infer about the aliveness of the Digital Twin model.

Since we have used the gradient boosted trees, the feature re-ranking is performed based on the relative importance of each of the features. It is calculated as follows [151]:

$$\hat{I}_j^2(T) = \sum_{t=1}^{J-1} \hat{i}_j^2 1(v_t = j) \tag{6.6}$$

Where, we first define a $J$-terminal node tree $T$, and sum the result over the non-terminal nodes $t$. $v_t$ is defined as the splitting variable, and it is associated with each of the node $t$. The indicator function $1(.)$ has value 1 if its argument is true, and zero otherwise. And $\hat{i}_j^2$ is defined as the estimated empirical improvement in squared error in prediction as a result of split using the particular feature, and it is calculated as:

$$i^2(R_1, R_2) = \frac{w_l w_r}{w_l + w_r}(\bar{y}_l + \bar{y}_r)^2 \tag{6.7}$$

Where, $\bar{y}_l, \bar{y}_r$ correspond to the left and right daughter response means for the node respectively, and $w_l, w_r$ are the corresponding sums of the weights. For collection of decision trees

$\{T_m\}_1^M$ obtained through boosting, Equation 6.6 can be generalized with an average over all the trees as follows:

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m) \tag{6.8}$$

Hence, using Equation 6.8, the feature importance is calculated for the boosted trees, and this is used as a metric for re-ranking the features for virtual sensor placement using dynamic data-driven application systems.

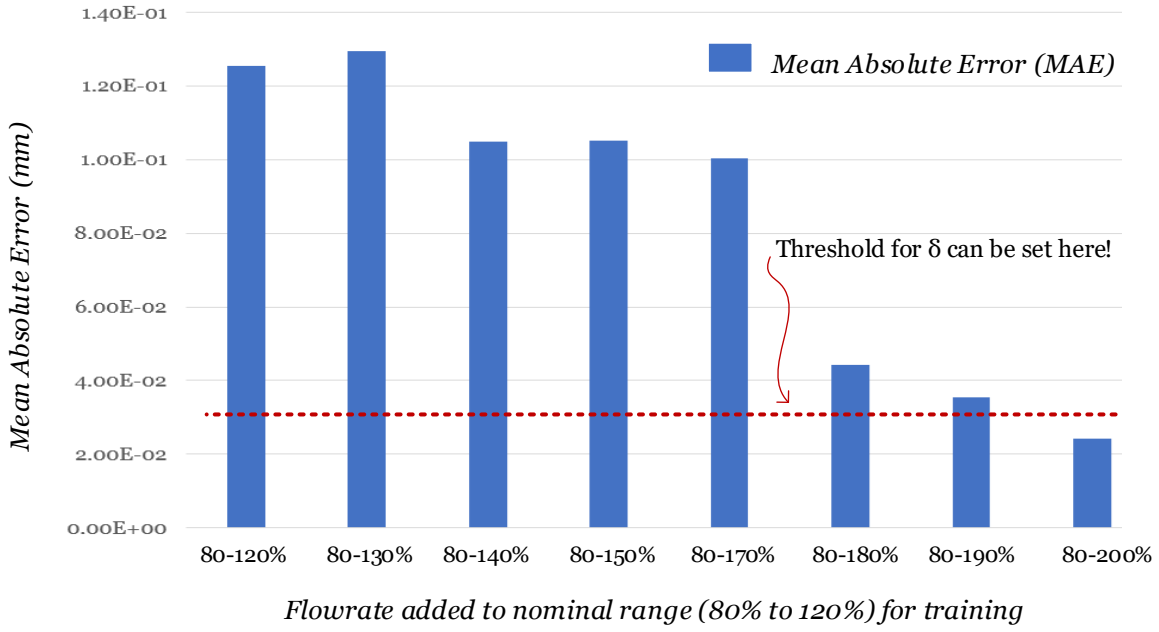



Figure 6.13: Aliveness test result for digital twin model predicting thickness of the floor

The results for modeling the Digital Twin for predicting the KPIs thickness of the floor and the wall are shown in Figure 6.13 and 6.14, respectively. The mean absolute error actually represents the mean absolute error for $\delta$. First of all, the accuracy of the Digital Twin for the optimal flow rate value is around 0.12 $mm$, which is around the resolution of the 3D printer. This proves that the Digital Twin can be modeled using the analog emissions which behave as the Digital Twin. It can also be seen that, as more recent degradation emissions are incorporated in the training sample, the delta value gets smaller and smaller for the

Digital Twin model predicting thickness values for both the surface and the wall.

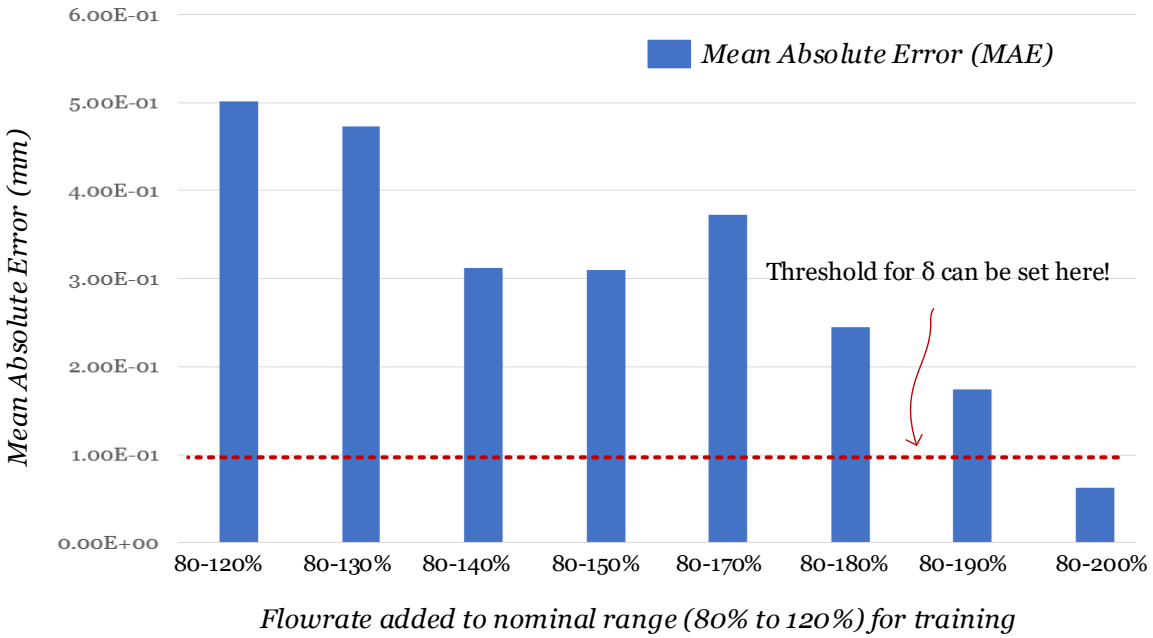



Figure 6.14: Aliveness test result for digital twin model predicting thickness of the wall

The delta threshold that can be selected for checking whether the Digital Twin can be set along $\pm$ 0.03 $mm$ of the previous delta value for the Digital Twin predicting the floor thickness. Whereas, for the Digital Twin predicting the wall thickness, the delta value threshold can be set at $\pm$ 0.1 $mm$. This shows the DDDAS system is capable of improving the Digital Twin model by providing feedback about the aliveness of the model.

The feature re-ranking results for checking whether the Digital Twin is up-to-date or not for large and small delta value is shown in Figure 6.15 and 6.16, respectively. From the figure, it can be seen that the feature importance for the Digital Twin has drastically changed when the delta was large compared to the one with the small delta. This means the DDDAS system is re-ranking the features dynamically and updating the Digital Twin models. Notice that the feature names are presented as $[< Emission\ Name > \_ < Axis\ of\ Measurement > \_ < Signal\ Domain) > \_ < Feature\ Name >]$ for sensors measuring in the three axes, and $[< Emission\ Name > \_ < Signal\ Domain) > \_ < Feature\ Name >]$ for sensors measuring one-dimensional data (such as temperature, humidity, and power).

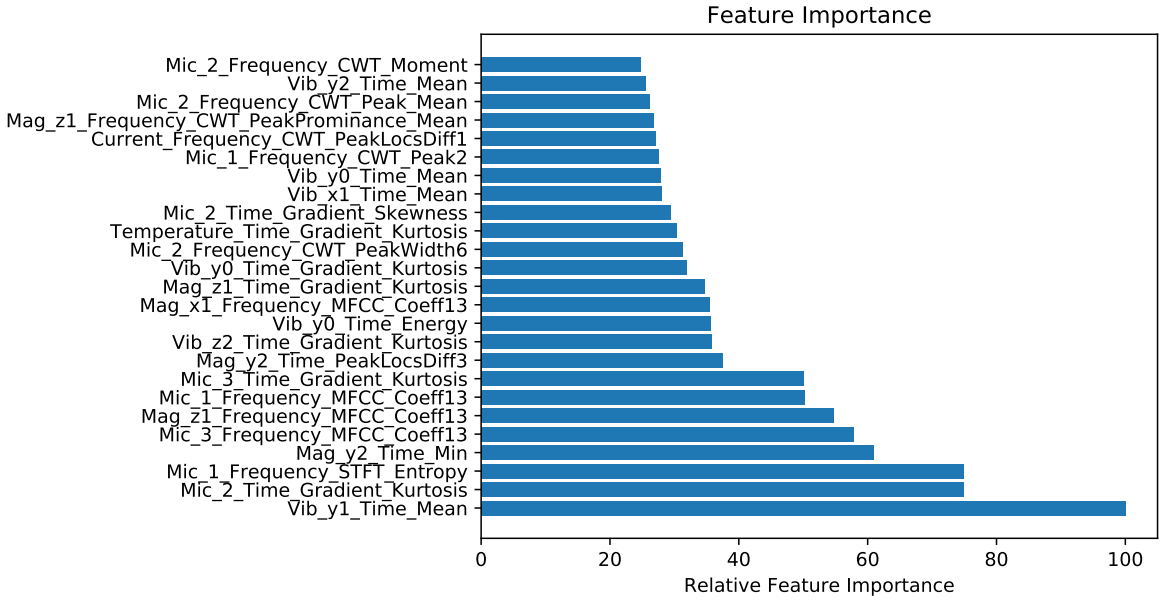



Figure 6.15: Feature importance for predicting floor thickness with largest delta

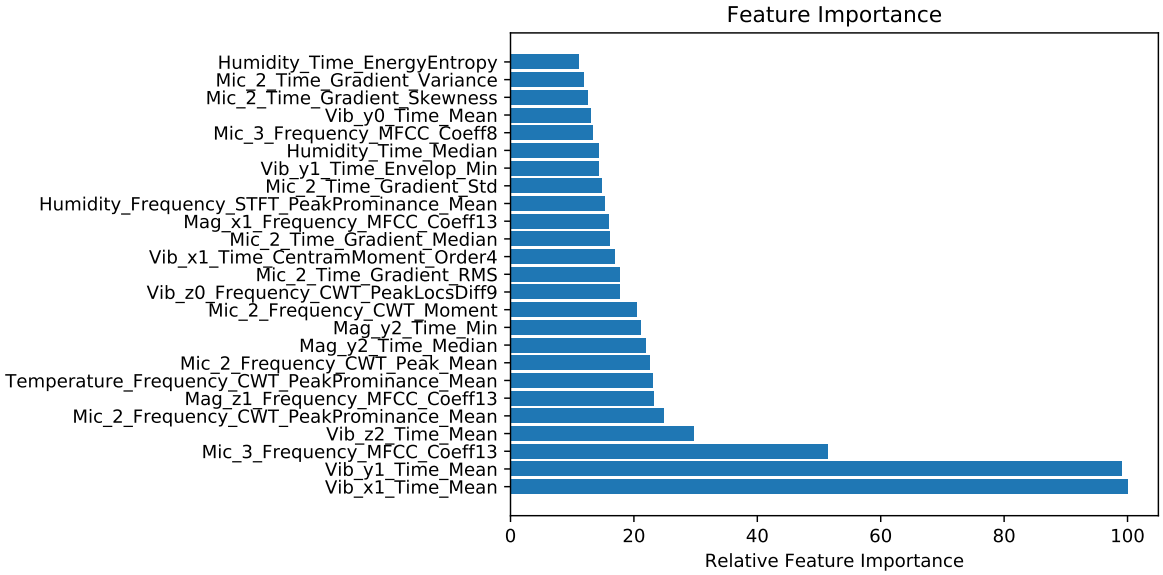
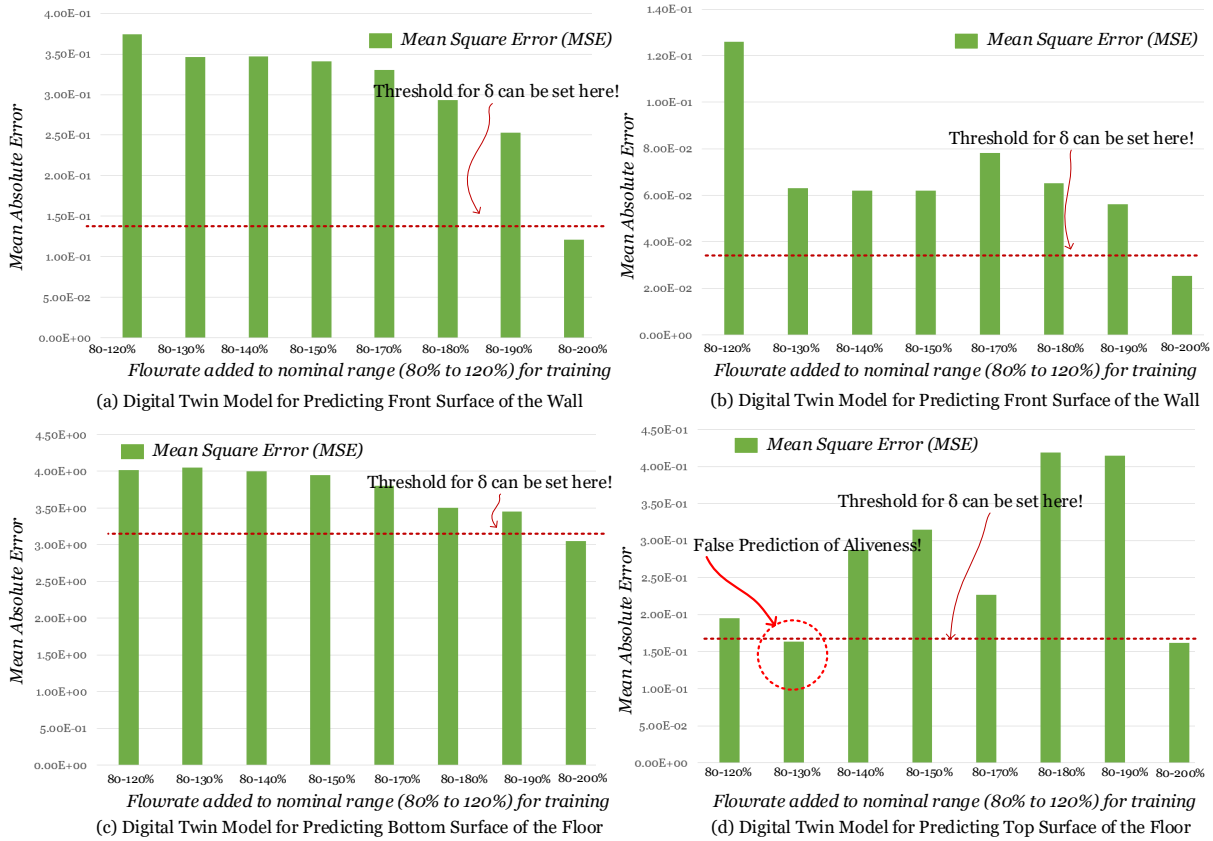



Figure 6.16: Feature importance for predicting floor thickness with smallest delta

The result for checking the results for modeling the Digital Twin that predict the KPIs surface texture of the floor and the wall is shown in Figure **??** (a), (b), (c), and (d) respectively. As predicted the delta value for the Digital Twin model predicting the surface texture for the bottom surface of the floor, the front surface of the wall, and back surface of the wall are decreasing with the incorporation of more recent analog emissions. The delta threshold that can be selected for checking whether the Digital Twin can be set along $\pm$ 0.17 of the

previous delta value for the Digital Twin predicting the top surface's texture of the floor.




(a) Digital Twin Model for Predicting Front Surface of the Wall

(b) Digital Twin Model for Predicting Front Surface of the Wall

(c) Digital Twin Model for Predicting Bottom Surface of the Floor

(d) Digital Twin Model for Predicting Top Surface of the Floor

Figure 6.17: Aliveness test while predicting surface quality

With this threshold, however, we see that there is one false positive that the Digital Twin is alive when the flow rate 130% is incorporated. The delta threshold can be set along ± 3.5 of the previous delta value for predicting the bottom surface's texture of the floor. The delta threshold can be set along ± 0.03 of the previous delta value for predicting the front surface's texture of the wall. The delta threshold can be set along ± 0.2 of the previous delta value for predicting the back surface's texture of the wall.

As seen in Figure 6.17 (d), checking the aliveness of the Digital Twin was met with unexpected variation in the delta values. On narrowing down the surface textures, we were able to find the reason for this unexpected behavior. The analysis is presented in Figure 6.18. In the figure, segment 2's surface of the top surface of the floor is presented for various flow rates. As we can observe that there is a trench-like structure on the surface which

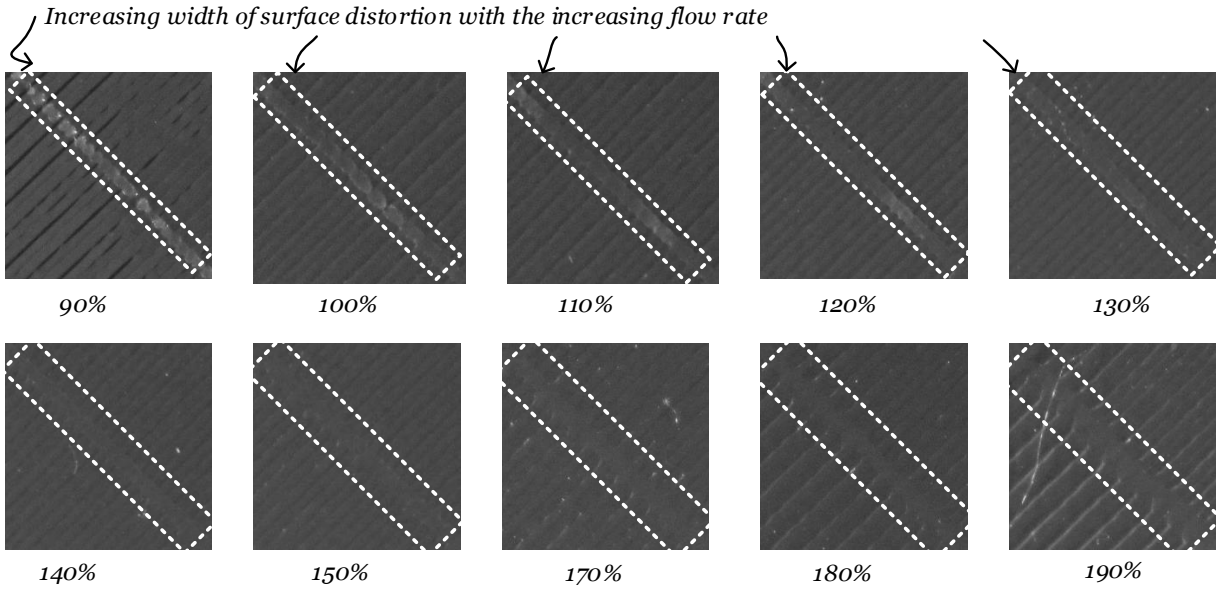



Figure 6.18: Surface textures with flow rate variation and the case of unaccounted trench has a varying amount of the width for various flow rates (in fact it increases with the flow rate). This is due to the heated nozzle passing very close to the surface of the object. This effect, however, happens on the next layer after the surface has been completed. In our algorithm, we have not incorporated the analog emissions for the layers (G-codes) that are not contributing towards making the surface. Hence, the training algorithms are not able to incorporate these changes, causing it to have unexpected performance.

## 6.7 Summary

In this chapter, we have presented how dynamic data-driven application systems concepts can be used to re-rank, and re-train the Digital Twin model. This model update is based on measurement of varying $\delta$, which measure the difference in predicted and real key performance indicators (for example surface texture and dimension). We showed how monitoring $\delta$ can enable the Digital Twin to have a cognitive capability of being aware of its aliveness. This methodology is scalable to create Digital Twin for multiple key performance indicator predictions, and towards other manufacturing systems as well. To the best of our knowl-

edge, this is the first work that demonstrates how dynamic data-driven application systems enabled feature re-ranking method can help in keeping the Digital Twin up-to-date.

# Chapter 7

# Part II: IoT-enabled Living Digital Twin Modeling

## 7.1 Introduction

In chapter 7, we presented a digital twin modeling methodology inspired by dynamic data-driven application systems. In this chapter, we will delve deeper into utilizing the side-channels to update the model. Digital twin consists of large historical context and performance data and utilizes the direct (through inbuilt sensors) and indirect (through latent variable analysis) sensing to provide the near real-time representation of the physical system. Moreover, it consists of various models (for simulation, monitoring, control, optimization, etc.,) in a hierarchical manner (consisting of a representation of the system, process, component, etc.,) which can provide the blueprint of the whole system [17].

Cyber-physical additive manufacturing technologies are still susceptible to defects due to the large diversity in the structure and properties of printed components [156]. Digital twin models, in this situation, could alleviate the cost of manufacturing by providing tools to

simulate and infer quality deviation in the virtual domain. In this work, we narrow down the scope towards Fused-Deposition Modeling (FDM) technology based 3D printers, which print 3D objects using thermoplastic filaments such as acrylonitrile butadiene styrene (ABS) or polylactic acid (PLA).

A key enabler for creating a *digital twin* is the availability of a large number of built-in sensors and their historical data. However, current FDM based additive manufacturing printers lack a large number of these sensors [157]. It mainly consists of sensor necessary for basic control (such as a temperature sensor, micro-switch, etc.,). Lack of sensor arrays makes it a difficult task for sensing the current system states, which is vital for digital twin models. Moreover, the task of building digital twins becomes even harder once the system has been manufactured as placement and selection of sensors for direct observation of system states can be challenging [158].

Previously, due to the lack of cheap sensors and high-speed and reliable network, acquiring the data for the *digital twin* was costly. Today, thanks to the availability and affordability of IoT sensors, it is becoming easier and cheaper for system operators to acquire large amounts of data on-the-go from their physical systems [159, 160]. However, for 3D printers that do not have built-in sensors and lack historical data, building the *digital twin* with IoT sensors is still a challenge.

### 7.1.1 Research challenges

The work in this chapter is motivated by three important research questions that apply to legacy FDM technology based additive manufacturing system without built-in sensors and access to historical data:

- Is it possible to build a *digital twin* of a FDM based additive manufacturing system by

indirectly monitoring the side-channels?

- Can this indirect side-channel based digital twin model faithfully captures the interaction between the environmental factors, process parameters of the system, and the design parameters of the product to explain the impact of such interaction on products?
- Can we retrofit low-end IoT sensors to maintain the digital twin up-to-date and use it to predict the product quality (localize faults and infer tolerance deviation) of the next product being produced?

### 7.1.2 Our contribution

To address these research challenges, this chapter provides a study on the limits of various sensors modalities (such as acoustic, magnetic, power, vibration, etc.) and their contributions towards building and maintaining a *living digital twin*. The key insight of our work is that manufacturing machines generate *unintended* side-channel emissions that carry valuable information about *the machine itself, the product they are producing, and the environment*. Our methodology uses IoT sensors to capture these side-channel information and build a *living digital twin* (see **Section 7.3.6**). Our main contribution is a novel methodology to monitor production machine degradation, build their *living digital twin*, and use this *living digital twin* to provide product quality inference (see **Section 7.3.7**) while localizing faults (see **Section 7.3.5**).

***Motivational case study for multisensor data analysis:***
In this work, we analyze the data collected from multiple sensors commonly available in IoT devices. These sensors (such as an accelerometer, Hall-effect magnetic sensors, microphone, etc.) are commonly used in the state-of-the-art IoT devices. Moreover, recently more and more sensors (such as current, humidity, temperature, etc.) are added in IoT devices. For building the system *digital twin*, we propose to capture the interaction between the cyber-

domain data (such as G/M-codes carrying geometry and process information), the physical domain input of the system (such as raw materials and energy), and the environment. G/M-codes consists of G and M code. The *digital twin* of the product (the 3D object being created) is initially described using a Computer-Aided Design (CAD) tools. The CAD tool then produces StereoLithography (STL) files which consist of geometry description of the object in coordinate space. Then a Computer-Aided Manufacturing (CAM) tool takes the STL file and slices it into multiple layers and finds a trace to be followed to print the object in each layer. The output of the CAM layer is the G/M-code.

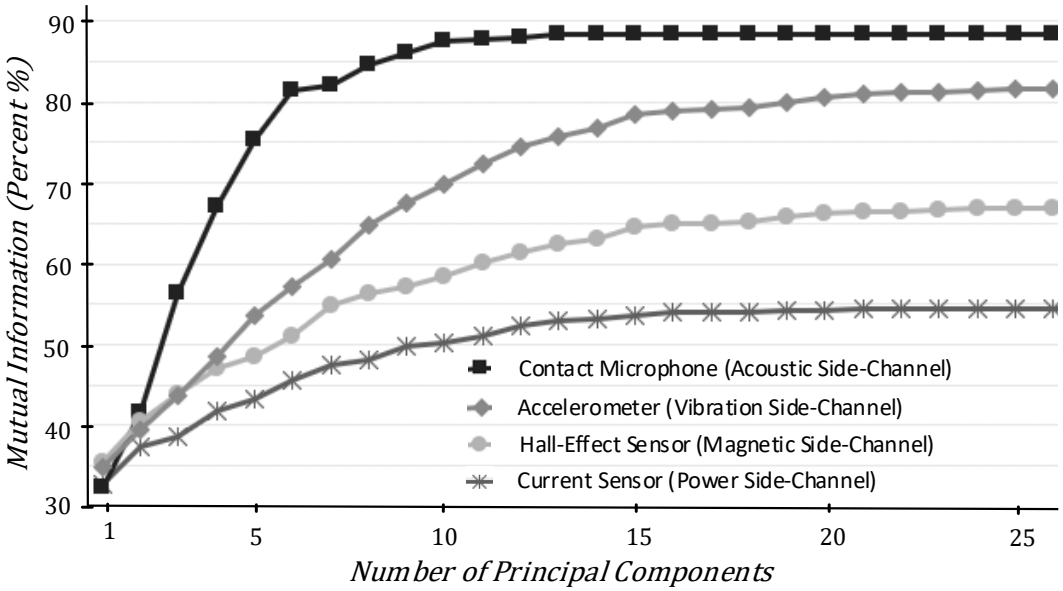



Figure 7.1: Mutual information analysis on side-channels

In our experiment, we consider that the *digital twin* of the product is described using the G/M-code. G-codes are responsible for controlling the motion (XYZ-axes and extrusion rate of filaments) of the machine for constructing certain geometry shape. Whereas, M-codes are responsible for controlling the machine parameters (process parameters such as temperature, acceleration of stepper motors, etc.). Recently, researchers in [161, 77] have demonstrated that various signals (such as acoustic, magnetic, etc.) collected from the 3D printer behave as side-channels and reveal information about the cyber-domain. Motivated by these work, we performed a preliminary study to measure the mutual information between the four

types of sensor data (*acoustic*, *magnetic*, *vibration*, and *power*) and the G/M-codes in a 3D printer. A variation of angle ($0^o$ to $90^o$ with step size of $9^o$) and speed (700 mm/minute to 3300 mm/minute with step size of 100 mm/minute) for printing line-segments is encoded in G/M-codes (with total of 297 G/M-codes) and the principal components various time and frequency domain features extracted from the side-channels were used to calculate the mutual information.

Figure 7.1 shows that various data collected from different sensors. The percentage of mutual information in the y-axis represents how much of the total Shannon entropy of the G/M-code ($log_2(297) \approx 8.21$ *bits*) can be explained by the analog emissions. We have assumed the distribution of G/M-codes to be uniform for calculating the Shannon entropy. The figure shows that different sensors have varying level of mutual information with the cyber-domain G/M-codes. This means that they behave as side-channels and provide information about the cyber-domain. However, the data collected from these sensors not only allow us to infer the cyber-domain data, but it also captures the current system status (mechanical degradation, system vibration, effects of the environment on the system, etc.). Hence, this work leverages the side-channel data to build a living *digital twin*.

### 7.1.3 Related work

Since the concept origination, and the onset of emerging technologies, there have been various efforts to model the *digital twin* of a manufacturing system. Knapp and Mukherjee in [147] provide building blocks for modeling *digital twin* for laser-based directed energy deposition additive manufacturing. They use the *digital twin* to estimate the effects of the process variables on cooling rates, single layer deposit geometry, and other structural features. Debroy and Zhang in [146] surveyed the state-of-the-art and motivate the need for more building blocks to create *digital twins* of additive manufacturing systems. Boschert

and Rosen in [162] highlight the simulation aspect of the *digital twin* and its use in the product life-cycle. Alam and Saddik in [163] provide the reference model for the cloud-based cyber-physical system with the implementation of a Bayesian belief network for dynamically updating system based on current contexts. Schroeder and Steinmetz in [164] provide a methodology to model the attributes related to the *digital twin* for providing easier data exchange mechanism between the *digital twins*. Cerrone and Hochhalter in [165] present finite element models of as-manufactured models to predict the crack path for each specimen. Authors in [166] provide a semantic layer which provides a mechanism to pass control feedback and evolve the build parameters on-the-fly for compensating the tolerance.

In summary, all the work have focused on either building the digital twin using simulation of the first principle based equations [167, 168] or just placing expensive sensors for in-situ [169, 170] process monitoring. There is work that uses low-end sensors for in-situ process monitoring [171, 172, 173, 36], however, they do not consider keeping the model up-to-date, using the indirect side-channels, which is the fundamental requirement for the *digital twin*. To this end, we propose a methodology for building the system *digital twin* and keeping it alive using low-cost sensors available in off-the-shelf IoT devices. We use the fact that some of the physical emissions act as side-channels, revealing information about cyber-domain, and that for every control signals in cyber-domain, there is a corresponding physical fingerprint in the physical domain. As it uses the side-channels, this methodology is different compared to the existing methods. Using the proposed methodology, we may be able to find new emissions (that may not have been considered during design time) that are able to better represent the cyber and physical states of the system during run-time.

## 7.2 Background

### 7.2.1 Concept definition

As briefly explained earlier, in a manufacturing environment, we have Digital and *physical twin* of product and system $DT_{product}$, $DT_{system}$, $PT_{product}$, and $PT_{system}$, respectively (see figure 7.2). The *digital twin* of the product $DT_{product}$ starts its life-cyle in the design phase, where computer aided design and computer aided manufacturing tools are used to represent the product in the cyber-domain. These product *digital twins* from design phase (with an instance represented using $X_i$) then go through the production, where the *physical twin* of the system $PT_{system}$ takes raw materials, energy, and the $DT_{product}$ to create its corresponding *physical twin*s (with an instance represented using $Y_i$). The *physical twin* of the system $PT_{system}$ consists of actual physical components that are used for manufacturing. The $PT_{system}$ is influenced by the manufacturing environment in a stochastic manner.

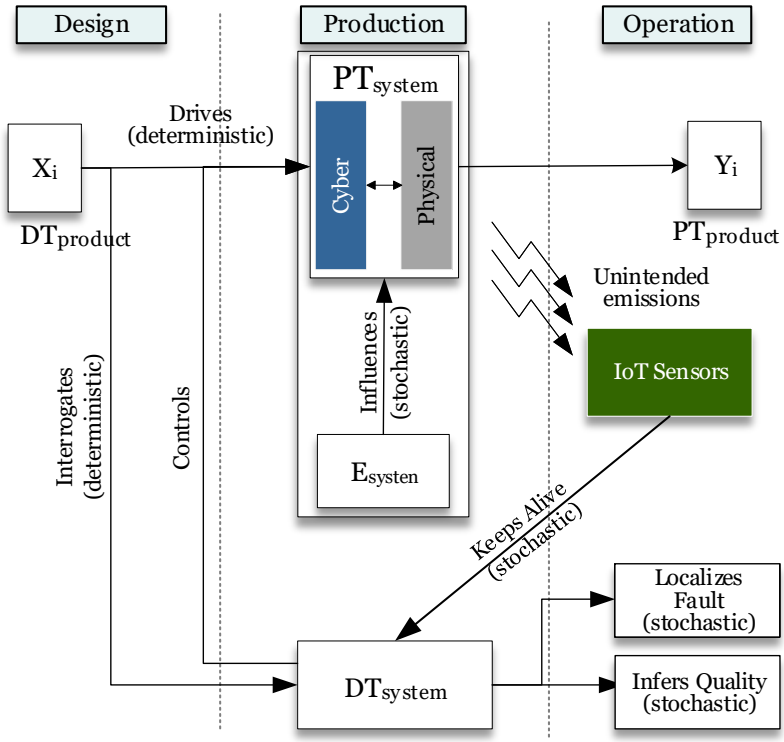



Figure 7.2: Digital twin concept for manufacturing

Let $DT_{product} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} : m \in \mathbb{Z}_{>0}, \alpha \in \mathbb{R}$ represent the parameters that define the *digital twin* of the product (such as dimension, surface roughness, mechanical strength, etc.), $PT_{system} = \{\beta_1, \beta_2, \ldots, \beta_n\} : n \in \mathbb{Z}_{>0}, \beta \in \mathbb{R}$ represent the parameters of the *physical twin* of the manufacturing system ( such as flow rate, acceleration values for motors, nozzle temperature, etc.), and let $E_{system} = \{\gamma_1, \gamma_2, \ldots, \gamma_p\} : p \in \mathbb{Z}_{>0}, \gamma \in \mathbb{R}$ represent the environmental factors affecting the manufacturing system (such as temperature, humidity, pressure, etc.). Here m, n and p represent the total product, system and environmental parameters that maybe considered for the modeling purpose. We propose to capture the interaction between these parameters using IoT sensors. Using the data collected from multiple modalities (acoustic, vibration, magnetic, power, etc.), we propose to model a stochastic function $\hat{f}(.)$, that performs three tasks: (1) localize the deviation in the $DT_{product}$ parameter from its *physical twin* $PT_{product}$, (2) make sure that the $DT_{system}$ is up-to-date (alive), and (3) infer the quality deviation for the $DT_{product}$ before creating the $PT_{product}$. Moreover, the $DT_{product}$ may interrogate the $DT_{system}$ to infer the quality deviation due to the current status of the $PT_{system}$.

### 7.2.2 IoT sensor data as side-channels

Manufacturing systems consists of cyber and the physical domain. The computing components in the cyber-domain have processes that communicate with the physical domain. A cross-domain signal that is passed from the cyber-domain to the physical-domain have the possibility of impacting the physical domain characteristics. This phenomenon is more prominent in manufacturing system where the *digital twin* of the product causes the *physical twin* of the system to behave in a certain deterministic manner. However, due to these characteristics there exists physical emissions (such as acoustic, vibration, magnetic, etc.) which also leak information about the *digital twin* of the product. We denote these emissions as side-channels, as they indirectly reveal the information about the cyber-domain interactions

due to the particular physical implementation of the system. For building the *digital twin* of the manufacturing system that captures the interaction between the product *physical twin*, the environment, and the system's *physical twin*, these side-channels play a crucial role in providing the necessary information. As show in [23, 25], there are various components of the system that reveal information about its internal states through the side-channels. In this chapter, we propose to utilize those indirect side-channel information for fault localization, quality inference and for updating the digital twin models. In this work, we analyze four such analog emissions which potentially behave as side-channels. Let $s_a(t)$, $s_v(t)$, $s_p(t)$, and $s_m(t)$ represent acoustic, vibration, power and magnetic emissions from the manufacturing system. Then we define each of these signals as:

$$s_a(t) = \hat{\delta}_a(\alpha_i, \beta_j) + \gamma_k : i <= m, j <= n, k <= p \tag{7.1}$$

$$s_v(t) = \hat{\delta}_v(\alpha_i, \beta_j) + \gamma_k : i <= m, j <= n, k <= p \tag{7.2}$$

$$s_p(t) = \hat{\delta}_p(\alpha_i, \beta_j) + \gamma_k : i <= m, j <= n, k <= p \tag{7.3}$$

$$s_m(t) = \hat{\delta}_m(\alpha_i, \beta_j) + \gamma_k : i <= m, j <= n, k <= p \tag{7.4}$$

Equations 7.1-7.4 represent the analog emissions as a result of the deterministic function $\hat{\delta}(.)$ which is influenced by the *digital twin* parameters of the product, $DT_{product}$, and the *physical twin* parameters of the System $PT_{system}$, and the non-deterministic environmental parameters $E_{system}$. Moreover, for each of the analog emissions, the total number of parameters $(\alpha, \beta, \gamma)$ may not be same. Traditionally, non-trivial simulation based approach such as finite element analysis is used to model the deterministic part and explore relation between the $DT_{product}$, $PT_{product}$, and $PT_{system}$. However, the $PT_{system}$ parameters vary over time, and $E_{system}$ parameters affect the $PT_{product}$ in a stochastic manner. Hence, we explore the possibility of using IoT sensors to model and maintain a live $DT_{system}$ for product quality inference.

### 7.2.3 Metric for quality measurement

The *digital twin* can be used for various purposes. However, one of the most fundamental uses of *digital twin* is in predicting the Key Performance Indicators (KPIs). Although the ultimate goal of the *digital twin* will be in predicting a variety of KPIs [174], we select quality as one of the KPIs. We will demonstrate that by maintaining a living *digital twin* we can infer the possible deviation in the quality of the product. One of the quality metrics that is used is the dimension ($Q_d$) of the product.

## 7.3 Building the digital twin

As mentioned earlier, we need to build the *digital twin* from the IoT sensor data to perform three tasks: run-time localization of faults, regularly update of the system *digital twin* and infer the quality of the product *digital twin*. Hence, in this chapter, the digital twin model consists of algorithms and models associated with fault localization, fingerprint generation, and quality inference. For run-time localization, we propose to create and maintain an active fingerprint library of the individual IoT sensor data corresponding the $DT_{product}$ parameters. This fingerprint also captures the $PT_{system}$ and $E_{system}$ parameters during run-time. Then for localizing the faults, the deviation of the run-time IoT sensor data is compared with the fingerprint. For updating the *digital twin*, a voting scheme is used to check if the majority of the fingerprints are deviating corresponding to a few fingerprints. To infer the deviation in quality, we have proposed to estimate a function $Q_d = \hat{f}(\alpha, \beta, \gamma, s_a(t), s_v(t), s_m(t)), s_p(t))$, where the $Q_d$ is a function of $DT_{product}$, $PT_{system}$ and $E_{system}$ parameters, and the IoT sensor data.The proposed methodology is shown in figure 7.3. The various components of the proposed methodology are explained as follows:

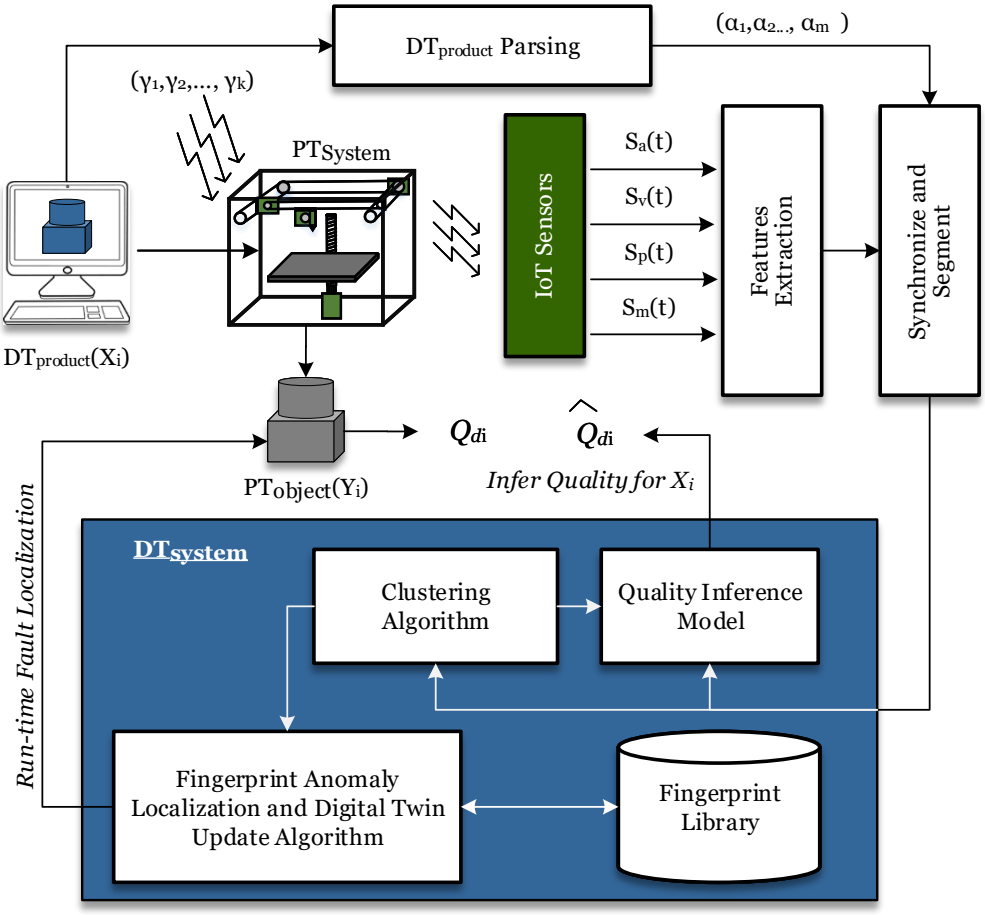



Figure 7.3: Digital twin modeling methodology

### 7.3.1 $DT_{product}$ parsing

For generating the fingerprint of the $DT_{product}$ from the IoT sensor data, first of all it is parsed to its corresponding parameters $(\alpha_1, \alpha_2, \ldots, \alpha_m)$. The parsed values will depend on the type of manufacturing system. In the experimental section, we will present the parsing for an additive manufacturing system that uses G/M-codes. These codes are the instruction that carries the process (machine specific parameters, such as temperature, acceleration values for motors, etc.) and product information (for example the geometry description). The parsing will break down the individual parameters from the product *digital twin*.

### 7.3.2 Feature extraction

For generating the fingerprint from the analog emissions, various time domain features such as *Energy*, *Energy Entropy*, *Peak to Peak features* (highest peaks, peak widths, peak prominence, etc.), *Root Mean Square values*, *Skewness*, *Standard Deviation*, *Zero Crossing Rate*, *Kurtosis* (114 features in total) and frequency domain features such as *Mean Frequency*, *Median Frequency*, *Signal to Noise Ratio*, *Spectral Entropy*, *Spectral Flux*, *Spectral Roll Off* from short term 50 millisecond time domain windows (also known as Short Term Fourier Transform) and Continuous Wavelet Transform (CWT) (140 in total), 20 Mel-frequency cepstral coefficients (MFCCs), etc., are analyzed from IoT sensor data. All the analog signals are first synchronized by performing up and down sampling and testing the various window size (5 ms to 100 ms) for highest model accuracy (50 ms in our case). These features have been selected by calculating the Gini importance or mean decrease impurity of well-known time and frequency domain features (¿¿1000 in total) used for the analysis of time-series data [175]. Principal Component Analysis (PCA) is then performed to further reduce the dimension of these features. Let $m$ be the total number of reduced feature set, then all the features are concatenated for $n$ total samples to create a feature matrix $O \in IR^{nxm}$.

### 7.3.3 Synchronize and segment

Before clustering is performed, the features are synchronized and segmented into subgroups based on the parsed $DT_{product}$ parameters $(\alpha_1, \alpha_2, \ldots, \alpha_m)$. For instance, the features are segmented based on conditions such as presence or absence of particular component's movement (for example, motors responsible for moving the 3D printer nozzle in $X, Y, Z$-Axes). By segmenting based on the parsed parameters, the features are reduced into smaller groups. This allows for further reducing the complexity in acquiring the fingerprints. Henceforth, *group* is used to denote the sub-division of the $DT_{product}$ parameters, which are different than

the clusters estimated in the subsequent sections.

### 7.3.4 Clustering algorithm

For generating the fingerprint of the parsed parameters of $DT_{product}$, a clustering algorithm is used to generate clusters that group similar features into a single cluster. For analyzing the clustering algorithm and the corresponding fitness of cluster number, the silhouette coefficient is calculated for each sample. It measures the similarity of the feature to its assigned cluster compared to other clusters, with a high value representing its close match to the assigned cluster. It is calculated as follows:

$$silhouette\ coefficient(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{7.5}$$

where $a(i)$ is the average intra-cluster distance, and $b(i)$ is the mean of the nearest cluster distance (lowest average distance of $i$ with all other points in another cluster where $i$ is not a member). The clustering is carried out for each group of the features for all the analog emissions. The cluster centers, cluster number and the corresponding average silhouette coefficient of all the analog emission is stored in a library, effectively representing the fingerprint for the given parsed $DT_{product}$ parameter.

The pseudo-code for generating the cluster and saving the fingerprint is presented in algorithm 11. Features with their corresponding group and channel name are passed as input and the fingerprint in the form of clusters and their corresponding silhouette scores are given as output. First, Line 1 and 2 initialize the cluster numbers and Silhouette Score Threshold for measuring the accuracy of the cluster estimation. Then Line 3 splits the features into test and train set. Normally 80% of the data is used for training and 20% is used for testing while performing k-fold cross-validation [176] to validate the accuracy. For each cluster number, Line 7 estimates the clusters for the training set. Then, Line 8 measures the ac-

**Algorithm 11:** Algorithm for the fingerprint library generation for digital twin

**Input:** Features: $O \in IR^{nxm}$, Groups: $G$, Channels: $Ch$
**Output:** Fingerprint: ($G$, $Ch$, Clusters:$C_k$, Silhouette Scores)

```
1  Initialize K= 1, 2, ..., m
2  Initialize Silhouette Score Threshold SC_Threshold
3  Split O ∈ IR^nxm into Test and Train set
4  foreach ch ∈ Ch do
5      foreach g ∈ G do
6          foreach i ∈ K do
7              Estimate i clusters using Train set of Features
8              Use SC_Threshold to measure accuracy for clustering the Test Set features
9          Select cluster number (k) with highest accuracy
10         Re-estimate k cluster with all the Features
11         Calculate and Store Silhouette Score_ch_g
12 return Fingerprint: (G,Ch, Clusters:C_k, Silhouette Score_ch_g)
```

curacy of the estimated cluster with a specified silhouette score threshold for the test set of features. Based on the obtained accuracy in Line 8, Line 9 to 11 select the cluster number, re-estimate the cluster and store the silhouette scores for all the groups and the channels (acoustic, magnetic, power, and vibration signals).

### 7.3.5 Anomaly localization algorithm

The proposed digital twin model of the system is utilized to detect and localize anomaly in the product. To do this, a fingerprint library is created using algorithm 11 is used for detecting and localizing the anomalous physical signals corresponding to the $DT_{product}$ while printing. The algorithm for detecting and localizing the deviation from the stored fingerprint is given in algorithm 12.

Algorithm 12 parses the features of the $DT_{product}$ either run time or after the product's *physical twin* has been created. Then, using the fingerprint library it estimates the new cluster labels for the parsed features in line 5. Using these labels and the features the new silhouette coefficient for the parsed features are calculated in line 6 using Equation

**Algorithm 12:** Algorithm for localizing deviation and checking for digital twin update

**Input:** Features: $O \in IR^{nxm}$, $DT_{product}$
**Input:** Fingerprint($G$, $Ch$, Clusters:$C_k$, Silhouette Scores $SC_{FP}$)
**Output:** Segment of $DT_{product} with deviation$

Parse $DT_{product}$ into corresponding parameters
Segment Feature into corresponding group
**foreach** $ch \in Ch$ **do**
    **foreach** $g \in G$ **do**
        Get cluster labels $CL_i$ for Features $O_i$ by assigning features to the nearest cluster in $C_k$
    Estimate current silhouette coefficient ($SC_{current}$) for estimated cluster labels
    **foreach** $O$ *and* $CL$ **do**
        **if** $SC_{current} ¡ SC_{FP} + SC_{Threshold}$ **then**
            Store $DT_{product}$ segment ($Seg$)
            $DeviationFlag_{ch_g}$+=1
    $\triangle Deviation_{ch_g}$=$DeviationFlag_{ch_g}$/Total $DT_{product}$ **if** $\triangle Deviation_{ch_g} ¿ feature_{Threshold}$ **then**
        $Deviation_g$+=1
$\triangle Deviation_g$=$Deviation_g$/Total Group
**if** $\triangle Deviation_g ¿ group_{Threshold}$ **then**
    $Deviation_{ch}$+=1
$\triangle Deviation_{ch}$=$Deviation_{ch}$/Total Channel
**if** $\triangle Deviation_{ch} ¿ channel_{Threshold}$ **then**
    Use algorithm 11 to update the library
**return** $Seg$

7.5. If the calculated silhouette coefficient is less than the stored silhouette coefficient ± threshold $SC_{Threshold}$ then the $DT_{product}$ segment corresponding to the feature is marked as deviating from the previous fingerprint and returned as containing a possible anomaly. Moreover, G/M-code adds layers to print the 3D object in sequential order. Hence, if a fault is detected at a certain time, it can be correlated to locate its position in the 3D object.

### 7.3.6 Digital twin update algorithm

For updating the *digital twin* model, the library of fingerprint for the $DT_{product}$ have to be updated. However, before updating the library, it should be checked if the anomaly in the

fingerprint is temporary or it is due to the degradation of the machine over time. In order to update the *digital twin*, line 10 in algorithm 12 keep tracks of all the $DT_{product}$ variables that deviated. Then line 11 checks if more than $feature_{Threshold}$ of the $DT_{product}$ parameters deviated from the previous fingerprint. Then line 14 checks if more than $group_{Threshold}$ of the groups deviated from the previous fingerprint. Finally, line 17 checks if more than $channel_{Threshold}$ of all the channels deviated. If these condition are met then in line 18 the library for the *digital twin* is updated. This threshold for checking the deviation from the fingerprint can be varied for different channels and groups based on the amount of information leaked by each of the side-channels.

### 7.3.7 Quality inference model

To infer the quality variation, we estimate a function $Q_d = \hat{f}(., \theta)$, where $\theta$ represents a function parameter that needs to be learned. Specifically, we treat $Q_d$ as a function of analog emissions, product design parameters, process parameters, and environmental parameters. The quality deviation occurs due to the fact that the environment ($\alpha$) affects the $PT_{system}$ process parameters ($\beta$). Due to this, when the $DT_{product}$ is sent to the manufacturing system, variations are introduced in the $PT_{product}$. However, when the environment affects ($\alpha$) the process parameters ($\beta$) it changes the physical structure of various components (for example creation of rust, mechanical eroding, etc.). These changes may cause the side-channel analog emissions from the $PT_{system}$ to vary. The relations between various environmental factors, process parameters, and design parameters may be modeled using first principle (using physics-based equations). However, estimating such functions will require rigorous multi-domain analysis of the complex mechanical system, and may not reflect variation introduced when the system is operating. Instead, we propose to use a data-driven modeling approach to estimate the function $Q_d = \hat{f}(., \theta)$. This function is estimated using a supervised learning algorithm. To do this, for various $\alpha$, $\beta$, and $\gamma$ values the corresponding emissions

needs to be collected. However, for experimental purpose, we assume that the environmental variation affects the $\beta$ values. Hence, we only vary $\alpha$ and $\beta$ values and collect the corresponding analog emissions from the side-channels. We extract various time and frequency domain features from these analog emissions and together with $\alpha$ and $\beta$, construct a feature matrix $O \in IR^{nxm}$. Where $m$ represent the total time and frequency domain features concatenated with $\alpha$ and $\beta$ parameters, and $n$ represents the total samples. Then, we label each of the rows of $O \in IR^{nxm}$ to its corresponding quality values and use supervised learning algorithm to estimate the function $Q_d = \hat{f}(O \in IR^{nxm}, \theta)$. More specifically, gradient boosting based regressor [177] is used to estimate the function $\hat{f}(.)$. It uses an ensemble of decision trees based regression models. This ensemble generates a new tree against the negative gradient of the loss function and combines weak learner to control over-fitting. Hence, they are robust to outliers and outperform many other learning algorithms as demonstrated in [32]. Since regression trees are used as weak learners, we need to estimate various hyper-parameters such as learning rate, number of weak estimators, maximum depth of the weak learners, etc., to improve the capability of the model to generalize. To do this, the collected feature matrix is divided into test and training set. Then, the testing and training accuracy is used to determine the hyper-parameters that best generalize the function. This estimation function is also updated when the *digital twin* update algorithm reaches a consensus that all the fingerprints are outdated.

## 7.4 Experimental Setup

### 7.4.1 IoT Sensors

For analyzing the analog emissions from the side-channels, four acoustic (AT2021 cardioid condenser and a contact microphone with sampling frequency set at 20 kHz, whereas high-end

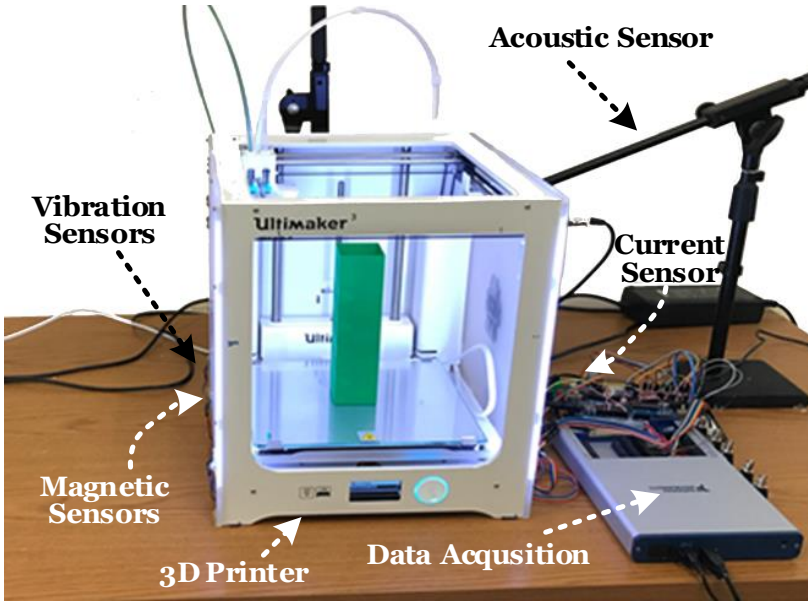



Figure 7.4: Experimental setup for modeling the digital twin

industrial microphones have higher sampling frequency greater than 40 kHz), one vibration (Adafruit triple-axis accelerometer with output date rate ranging only from 1.56 Hz to 800 Hz and measurement range of up to ±8g, whereas high-end accelerometers have ranges beyond 1 kHz with measurement range around ± 50g), one magnetic (Honeywell's magnetometer HMC5883L with output date rate ranging from 75 Hz to 160 Hz and measurement range between ±1 to ±8 gauss, whereas high-end magnetic field sensors have a date rate range of more than 1 kHz and measurement range between ±0.6 to ±100 gauss, and current (a low range Pico current clamp with measurement range of 10 mA to 20A DC or rms AC with AC sampling frequency up to 20 kHz with measurement accuracy of ±(6.0%±30 mA), whereas high-end sensors have a much smaller resolution of less than 5mA in measuring minute current fluctuations) sensors are placed non-intrusively without hampering the normal operation of the system. In our experiment for demonstrating the applicability of the proposed methodology, we have used the above-mentioned sensors which have similar sensor specifications available in IoT devices [178]. The Cartesian FDM based 3D printer selected for the experiment is an Ultimaker 3 [179]. The placement of these sensors is performed by position exploration in Cartesian coordinate. The vibration and magnetic sensors measure signals in the X, Y, and Z axis. Hence, in total there are four acoustic, three vibration, three

magnetic, and current sensors. We consider them as 11 separate channels. Analog emissions from the additive manufacturing system (or a 3D printer) are automatically collected using National Instruments Data Acquisition (NI DAQ) system whenever a print command is given to it. The analog emission acquisition was carried out in a lab environment with sound pressure level varying between 60-80 dB. The digital twin models are trained and estimated in a desktop computer with Intel i7-6900K CPU with 3.20 GHz clock frequency, 32 GB of DDR3 RAM, and 12 GB of NVIDIA Titan X GPU. Moreover, the digital twin models are stored and retrieved using pickle operation in Python.

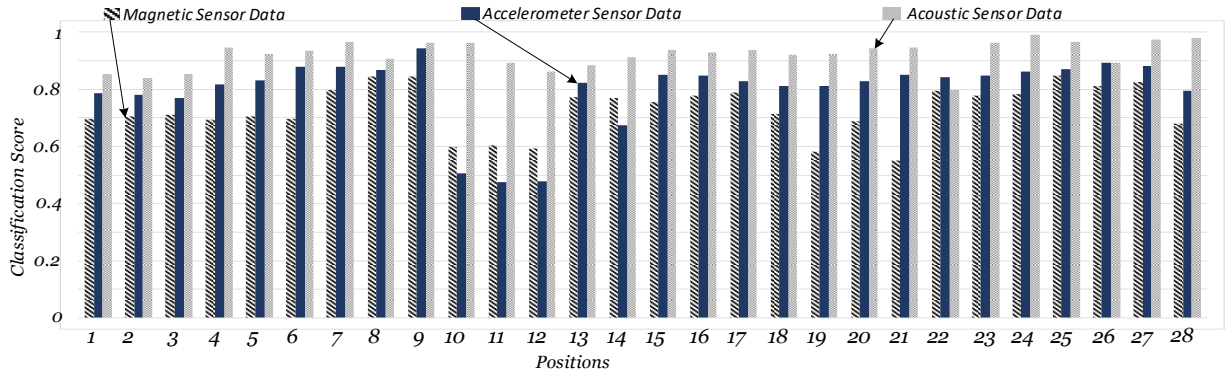



Figure 7.5: Classification accuracy score for sensors positions

### 7.4.2 Digital Twin parameters

The sample G/M-code ($DT_{product}$) consists of maximum six parameters, $G/M$ code specifying whether it is machine instruction or coordinate geometry information, travel feed rate $F$ of the nozzle head, the coordinates in $XYZ$-Axes each and amount of extrusion $E$. Various 3D test objects normally used for calibrating the 3D printer are downloaded from the open-source website [97] to extract sample $G/M - codes$. The parsing algorithm, in this case, separates the $DT_{product}$ based on presence or absence of 5 of these parameters, $G/M$, $X$, $Y$, $Z$, and $E$. Hence, $DT_{product} = \{\alpha_1, \alpha_2, \ldots, \alpha_{32}\}$ and there are 32 groups. When the manufacturing system is operating, the environmental parameters $E_{system} = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ affect the *physical twin* parameters $PT_{system} = \{\beta_1, \beta_2, \ldots, \beta_n\}$. This change in $\beta$ eventually

affects the $DT_{product}$ parameters, which in return affects the quality of the product. For example, environmental parameters such as humidity, temperature, etc., may affect the gearbox of the system, which in return may affect the flow-rate of the manufacturing system. For experimental purpose changing the environment parameters $E_{system} = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$ was not performed, instead we have assumed that this parameters eventually affect the $\beta$ parameter. Hence, analog emissions ($s_a(t)$, $s_v(t)$, $s_m(t)$, and $s_p(t)$) for various $\alpha$ parameters have been been collected for optimal $\beta$ parameters, and the environmental variability have been simulated by varying the $\beta$ parameters beyond their optimal values to check if the *digital twin* model is able to reflect those changes. After collecting the analog emissions, time and frequency domain features are extracted from each of them. Moreover, the $DT_{product} = \{\alpha_1, \alpha_2, \ldots, \alpha_{32}\}$ consists of timestamps to segment and synchronize the features. For the initial training phase, various G/M-code of the 3D-objects (cube, pyramid, cylinder, etc.) are given to 3D printer and their corresponding analog emissions are collected. From them, we proceed to generate the fingerprint library for maintaining the *digital twin* of the system. Furthermore, in all the training algorithms K-fold cross-validation has been performed to measure the performance of the models and prevent over-fitting.

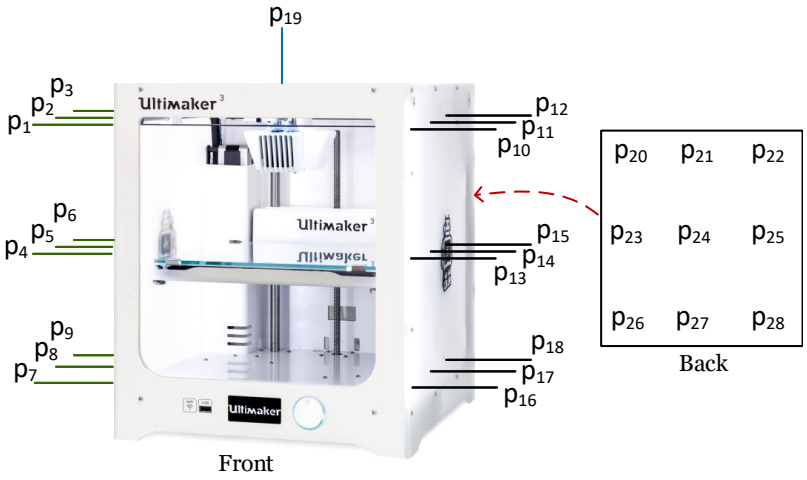



Figure 7.6: Experimental setup for sensor position exploration

### 7.4.3 Sensor position analysis

One of the challenges in IoT sensor-based information extraction is figuring out a non-intrusive position of the sensors. This task may also be machine specific. In our experiment, the 3D printers' external surface is considered for non-intrusively placing the sensors. Total of *28* uniform positions are selected. For each of the positions, vibration, acoustic, and magnetic sensors are placed and data is collected for various $DT_{product}$ parameters. Then a gradient boosted random forest is used to create a simple classifier to estimate the accuracy of the model based on various sensor location data. The accuracy of the classifier is given as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.6}$$

Where TP stands for total true positives, TN stands for total true negatives, FP stands for total false positives and FN stands for total false negatives. is taken as a metric for determining the placement of the sensors around the 3D printer. $DT_{product}$ parameters selected for estimating the classifier consists of simple G/M-code instructions (such as presence or absence of stepper motors movement in X, Y, and Z-axes).

Accuracy score of IoT sensor data is shown in figure 7.5, this score shows which of the positions of the sensors are capable of better classifying the stepper motor movements. It may be noticed that for different positions the classification accuracy is different. Moreover, these accuracy results also correlate the mutual information between the various sensor position and the side-channels themselves. Based on these values, a single position is selected for each of the sensors. However, since four acoustic sensors are used, positions with top four classification accuracy are selected for the sensor placement.

### 7.4.4 Performance of clustering algorithms

Various algorithms are explored for creating clusters to generate the fingerprints. Among them are *Mini batch K-means*, *Spectral Clustering*, *Ward*, *Agglomerative Clustering*, *Birch*, and *Gaussian Mixture method*. For each of the clustering algorithm, a varying number of clusters are initialized, and the corresponding silhouette coefficient is calculated for measuring the fitness of the features into these clusters.

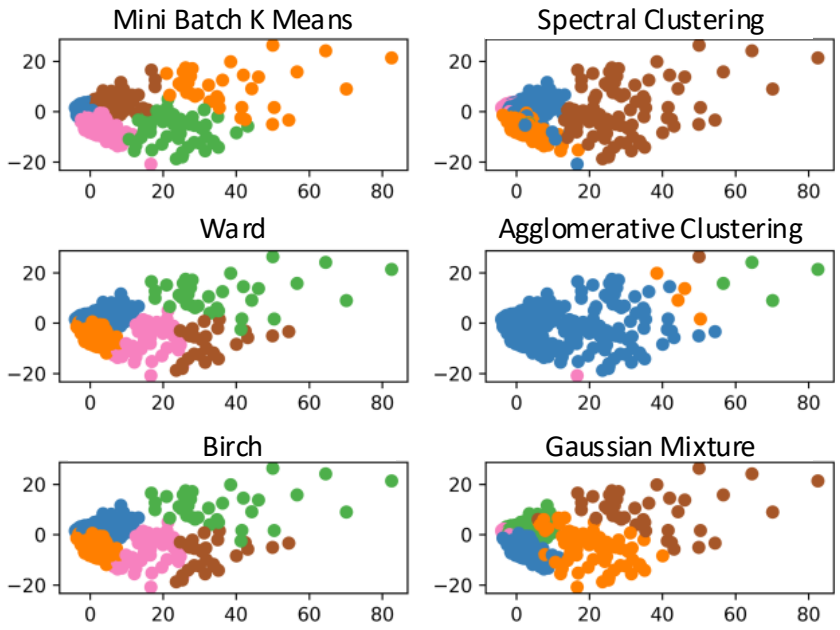



Figure 7.7: Scatter plots of the clusters plotted for acoustic side-channel

The average silhouette coefficients of the clustering algorithms for all groups and channels are shown in figure 7.8, and the corresponding scatter plots of acoustic side-channel for cluster number five is shown in figure 7.7. It may be noticed that although the *Agglomerative Clustering* has a higher silhouette coefficient value, from the scatter plot, the clusters are not well distributed in the scatter plot. However, the *Birch* clustering algorithm has relatively higher silhouette coefficient with a better spread of the cluster centers. Hence, *Birch* algorithm is selected for generating the clusters for fingerprinting the $DT_{product}$. Furthermore, the number of clusters is also estimated based on the accuracy of the *Birch* algorithm using

algorithm 11.

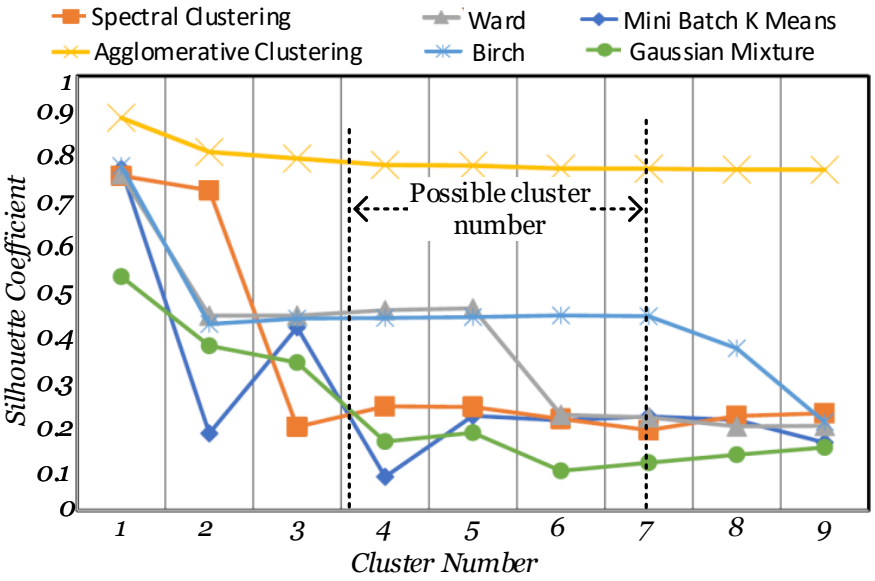



Figure 7.8: Silhouette coefficient of clustering algorithms

### 7.4.5 Anomaly localization accuracy

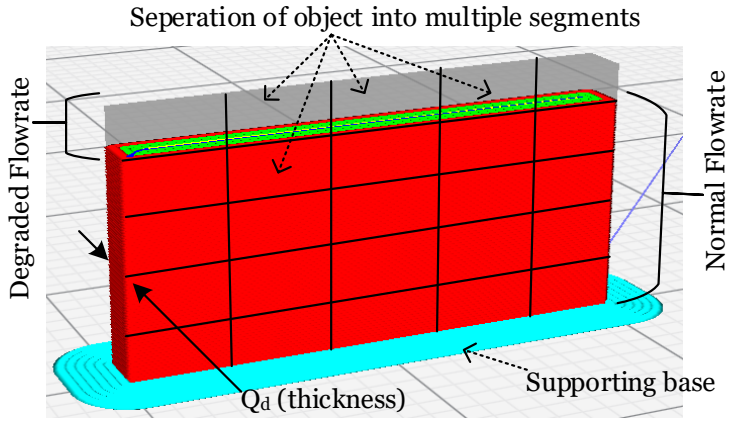



Figure 7.9: Test object for testing anomaly localization

For testing the accuracy of the *digital twin* for detecting the anomalous signals that can possibly cause deviation in the quality of the product, specialized test 3D object is designed (see figure 7.9). We have simulated variability of the environment by varying one of the $PT_{system} = \{\beta_1, \beta_2, \ldots, \beta_n\}$ parameters. In our experiment, we have selected flowrate as one

of the $\beta$ parameters. Flowrate should be maintained for uniform deposition of the filament while printing in fused deposition modeling based 3D printers. However, due to sudden slippage, faulty filament, etc., the flow of the filament may deviate from its nominal value.

The flow rate, a process specific parameter, is calculated as follows:

$$W * H = A = \frac{Q}{v_{feed}} \tag{7.7}$$

Where $W$ is the width and $H$ is the height of the line-segment being printed on the XY-plane, $Q$ is the constant volumetric flow rate of the material. $Q$ is estimated based on die swelling ration, pressure drop value and buckling pressure of the filament. And $v_{feed} = \omega_r * R_r$ is the feed velocity of the filament. Where $\omega_r$ is the angular velocity of the pinch rollers, and $R_r$ is the radius of the pinch rollers. Then, the pressure drop is calculated as follows:

$$P_{motor} = \frac{1}{2} \triangle P * Q \tag{7.8}$$

Where, $P_{motor}$ is the pressure applied by the stepper motors, $\triangle P$ is the pressure drop. Hence, the pressure applied by the motor needs to be maintained for the constant volumetric flow rate. This pressure needs to be less than buckling pressure which is calculated as follows:

$$P_{cr} = \frac{\pi^2 * E * d_f^2}{16 * L_f^2} \tag{7.9}$$

Where $E$ is the elastic modulus of the filament, $d_f$ is the diameter of the filament, and $L_f$ is the length of the filament from the roller to the entrance of the liquefier present in the nozzle. A sudden change in the pressure can cause the uniform flow of the filament to be disrupted. For validating the application of the digital twin in anomaly localization, the flow rate is varied outside the optimal range ($< 80\%$ and $> 120\%$) at a specific location (see figure 7.9) for multiple 3D objects. The anomalous flowrate variation introduced is between 40% and 180% with the step-size of $\pm 10\%$. Then the *digital twin* is tested to see if can accurately

classify the deviation in quality as an anomaly at the specific location. This is done by first segmenting the 3D object and assigning labels (1 for anomalous flow rate outside the optimal range, and 0 for normal flowrate) to these segments. Then comparing these labels with the results of the algorithm 12.

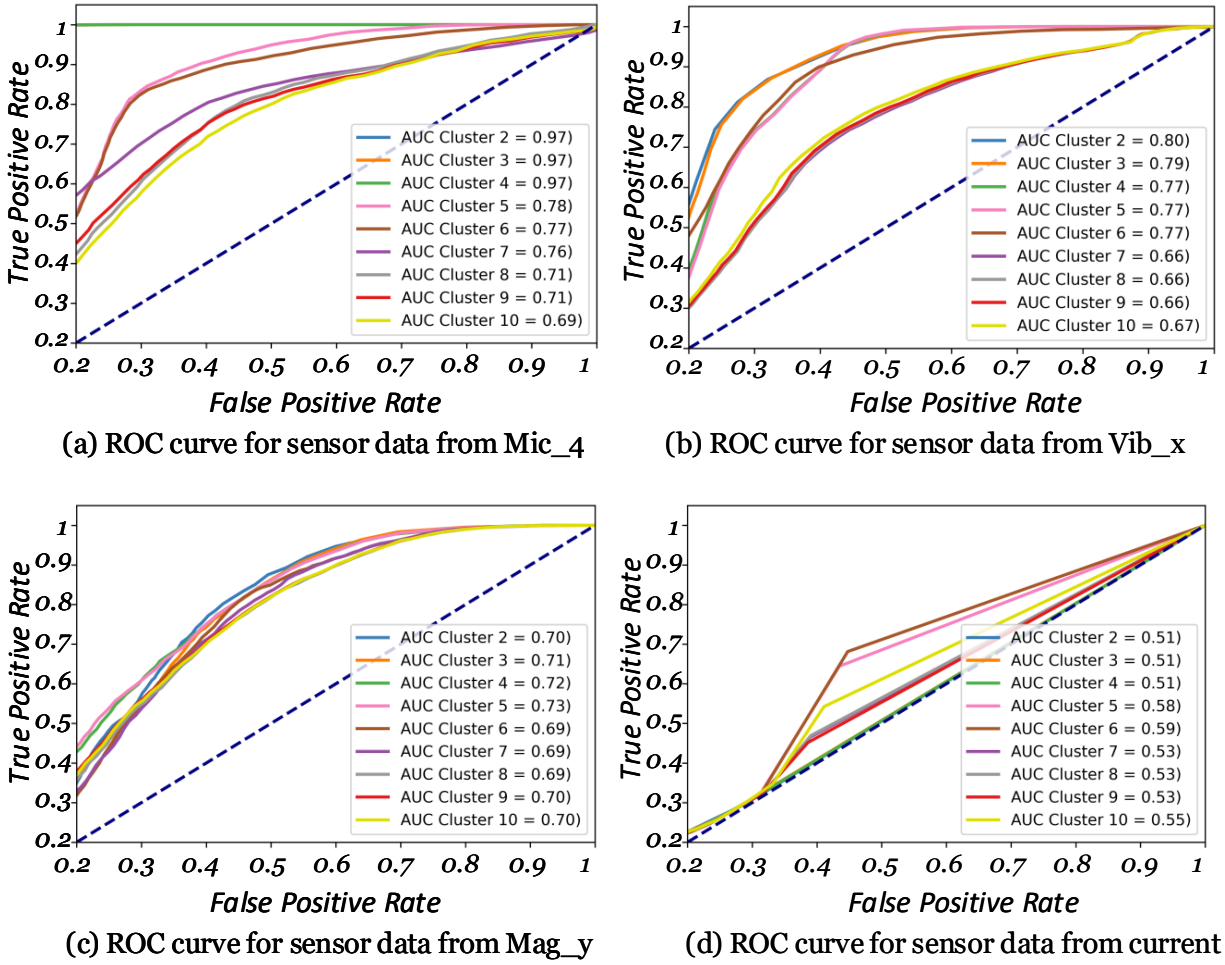



Figure 7.10: Average receiver operating characteristic curve for anomaly localization

The *digital twin* consists of fingerprints for the optimal flow rate in its library, and the corresponding clusters of the individual channels. When the object is printed, the corresponding features are passed to the *digital twin*, and the silhouetted coefficients corresponding the $DT_{product}$ is calculated. Based on algorithm 12, the analog emissions in each channel is labeled as either being within the deviation limit or exceeding the deviation limit of the silhouette coefficient. For selecting the optimal threshold for making this decision, initially the

threshold is varied and the corresponding accuracy of the detection mechanism is measured.

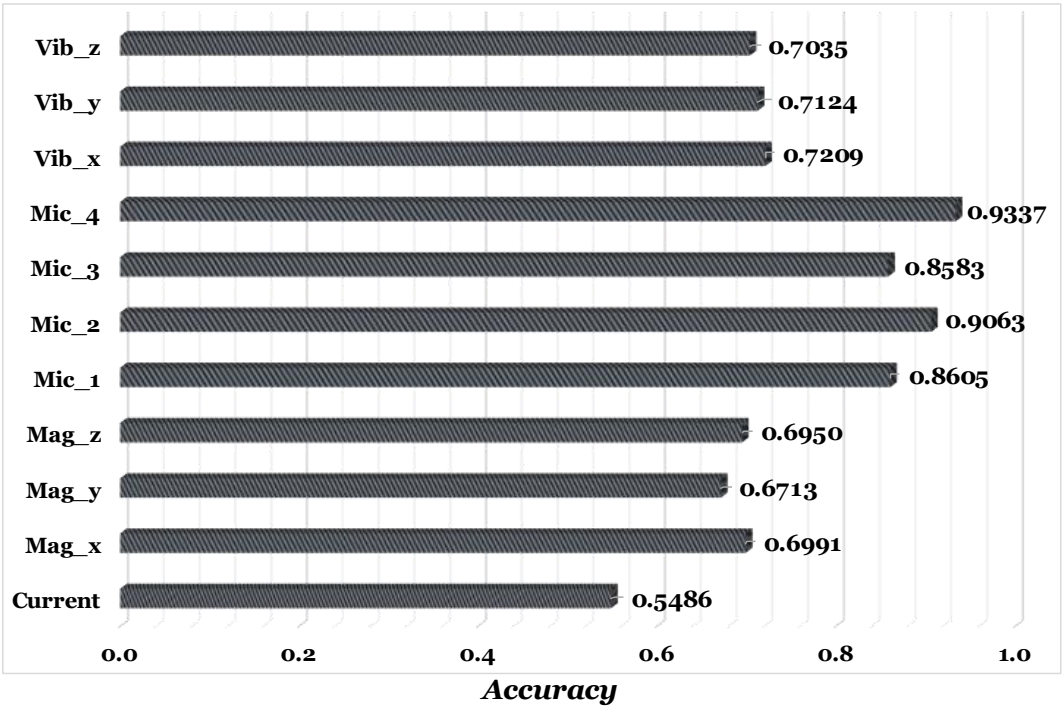



Figure 7.11: Accuracy of the digital twin's anomaly localization

Based on the highest accuracy acquired for each of the channels, the threshold is set and the corresponding classification accuracy for the segments that have been degraded is calculated. Corresponding to the varying threshold the Receiver Operating Characteristic (ROC) curve for some of the channels is presented in figure 7.10. The accuracy of each channel in detecting the anomalous flowrate is shown in figure 7.11. Since the features are time stamped, the corresponding section of the $DT_{product}$ maybe calculated after the *digital twin* has marked the features to be anomalous. From figure 7.11, it can be seen that analog emissions from microphone number four are more accurate in detecting the degradation of the flow rate. This is due to the fact that this emission is collected by the contact microphone attached near the extruder's stepper motor. Moreover, the average accuracy across all the channels in detecting the anomalous flowrate is **83.09%**.

### 7.4.6 System degradation prediction analysis

For detecting the degradation of the System, and hence the need for updating the *digital twin*, the flow rate for the entire $DT_{product}$ is varied beyond the optimal range. From equation 7.7 to 7.9, it is evident that various mechanical degradation (such as worn out rollers), stepper motor degradation over time, etc., may cause the flow rate to be reduced over time.

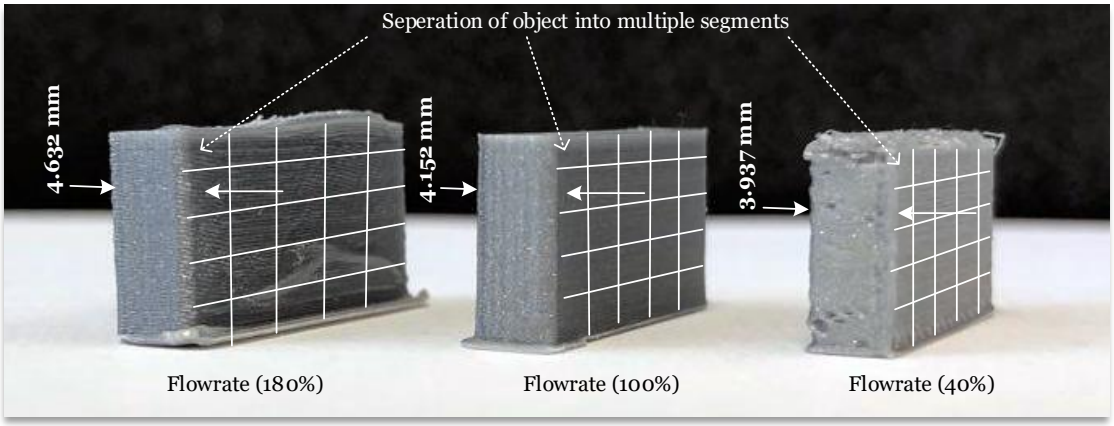



Figure 7.12: $PT_{product}$ created for testing quality inference and update capability

To check if the digital twin model gets updated to reflect the current status of the system, we perform two experiments. In the first experiment, the current *digital twin* with its fingerprint library is used to predict the class labels (*True* for update and *False* for do not update) for the degraded flowrate (60%). Then based on the result of algorithm 12, the updated (or the old) *digital twin* is used to predict the class labels again for the same degraded flowrate (60%) to see if the *digital twin* gets updated again. The result of degradation analysis is presented in Table 7.1.

Table 7.1 consists of *true negative rate*, *false positive rate* and update decision taken for each channel for the old cluster. When the system degrades, we expect the *digital twin* to find higher negative labels being generated as the silhouette score will be lower than the average silhouette score stored for all the channels and groups. It can be seen that out of eleven channels four of them had the decision of not updating the cluster, and seven of them opting

Table 7.1: Degradation test result for the digital twin

| | Old Clusters | | | New Clusters | | |
|---|---|---|---|---|---|---|
| **Channel** | **TNR** | **FPR** | **Update** | **TPR** | **FNR** | **Update** |
| **Mic_1** | 0.97423 | 0.0257 | True | 0.6040 | 0.3960 | False |
| **Mic_2** | 0.4962 | 0.5038 | False | 0.7460 | 0.2540 | False |
| **Mic_3** | 0.9705 | 0.0295 | True | 0.5731 | 0.4269 | False |
| **Mic_4** | 0.9867 | 0.0133 | True | 0.9798 | 0.0202 | False |
| **Current** | 0.5924 | 0.4076 | True | 0.5545 | 0.4455 | False |
| **Vib_x** | 0.9324 | 0.0676 | True | 0.8681 | 0.1319 | False |
| **Vib_y** | 0.9695 | 0.0305 | True | 0.7224 | 0.2776 | False |
| **Vib_z** | 0.4602 | 0.5398 | False | 0.4400 | 0.5600 | True |
| **Mag_x** | 0.4791 | 0.5210 | False | 0.6718 | 0.3282 | False |
| **Mag_y** | 0.4382 | 0.5618 | False | 0.4344 | 0.5656 | True |
| **Mag_z** | 0.6267 | 0.3733 | True | 0.3669 | 0.6331 | True |

for updating the clusters. Hence, the clusters are updated by algorithm 12. On the other hand, once the cluster has been updated, the analog emissions are labeled as true, hence we expect to see a higher true positive rate and lower false negative rate. In the table 7.1, it can be seen that only three of the channels gave the decision for updating the clusters again, however, eight of them opted for not updating the cluster. This shows that the *digital twin* is able to update itself during degradation that causes emissions in multiple side-channels to vary.

It may be noted that the side-channels gave different decisions for updating the digital twin. Out of them, the acoustic sensors and vibration side-channels were mostly able to predict the right decision, whereas the magnetic sensors were mostly wrong in this decision. This also correlates with the accuracy values presented in Figure 7.11. One anomaly to this is the current sensor data. However, it may be noticed that during both the decision model's true positive and true negative rates are very low compared to the acoustic and magnetic sensors. This means that the current side-channel data is not as reliable for the update decision as to the acoustic and the vibration side-channels.

### 7.4.7 Quality inference

For checking the accuracy of the *digital twin* in inferring the deviation of quality ($Q_d$), first of all, gradient boosting based ensemble of regressors is used to estimate the function $Q_d = \hat{f}(.)$ for the optimal flowrate range (80% to 120%). For each flow rate, five test objects (with the thickness of 4 mm for $DT_{product}$) are 3D printed, and for each test object, various segments (see figure 7.12) are created to measure the thickness using the micrometer. Then all the groups of features lying in these segments are assigned a single thickness value. Initially function $Q_d = \hat{f}(.)$ is estimated using optimal flowrate. Then it is used to infer the thickness of the 3D object for various feature samples with varying flow rates. The accuracy of the $DT_{systems}$ quality inference model is measured using mean absolute error value.

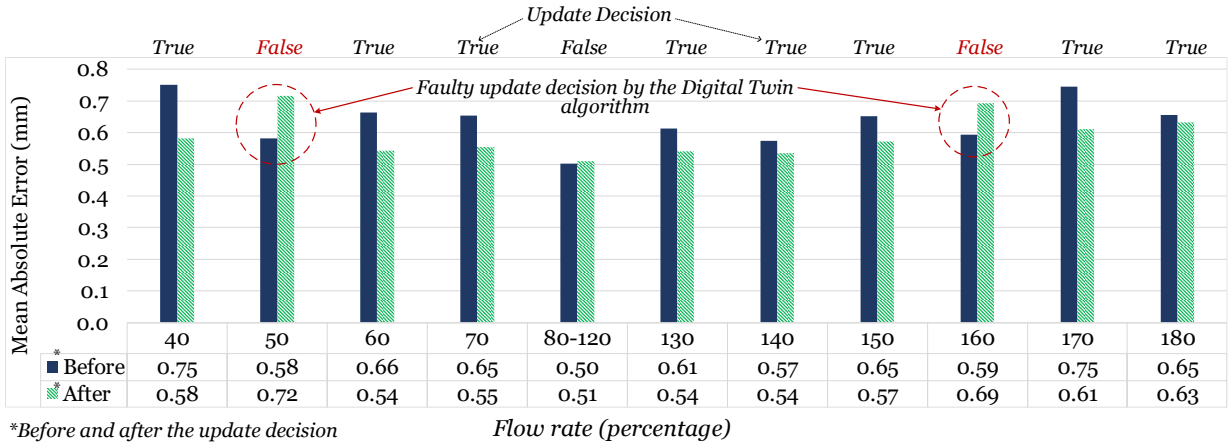



Figure 7.13: Accuracy of the quality inference model

The result of the quality inference is shown in figure 7.13. At first, the mean absolute error value of the inference model trained with optimal flowrate range is measured. It can be seen in the figure that for optimal flow rate ranges, the mean absolute error value is around 0.5 mm. Then, at each consecutive step the flow rate of the $Pt_{system}$ is varied with a step size of +10% in the positive direction ($> 120\%$) and at the same time +10% in the negative direction ($< 80\%$). It may be seen that in both directions when the system ages (degrades with an increase or decrease in the flow rate), the $DT_{system}$ has increased mean absolute error without the update. This is intuitive as the $DT_{system}$ has not been updated to the new fingerprints.

However, once it has been updated the mean absolute error is lower. It may also be noticed that when the system degraded with flowrates at 160% and 50%, the wrong decision was taken by the algorithm 12 in not updating the quality inference model. Due to this, a large increase in mean absolute error was observed for quality prediction other $DT_{product}$. However, this faulty decision was recovered in the consecutive stages. Moreover, the average mean absolute error in predicting the quality was 0.59 mm (calculated by averaging the mean absolute errors of the inference model after update decision).

### 7.4.8 Comparative analysis

Although we present a novel methodology of building a living *digital twin* by using IoT based sensors, there has been a considerable amount of work in quality prediction in additive manufacturing or manufacturing systems in general. In this section, we provide a qualitative comparative study of the various non-exhaustive list of methods compared to the proposed methodology. The result of the comparison is shown in Table 7.2. It may be observed that there are three general categories of research effort in maintaining quality.

The first is the first principle-based approach (simulation) [181, 167], where quality inference model is based on the process and design parameters. These models although are accurate, they do not account system degradation over time and requires a non-trivial formulation of physics-based equations. The second category involves in-situ process monitoring methodologies [180, 183]. These methods monitor the process variation using high-end acoustic and piezoelectric sensors. Compared to these high-end sensor-based methods, our method is able to keep the model updated even using low-end sensor data for fault localization and quality inference[1]. The third category involves process monitoring using low-end sensor placement [171, 172, 173]. They focus either on specific anomaly detection or quality variation detec-

[1]With high-end sensors as theirs, our methodology may achieve higher accuracy in anomaly detection along with the capability of keeping the digital twin most up-to-date at the cost of more computational and resource requirements, which may not be feasible for an IoT paradigm.

Table 7.2: Comparative analysis of the proposed methodology

| **Work/ System** | **Method** | **Metric** | **Sensors** | **Anomaly Detection Accuracy** | **Checks Model Update** | **Quality Inference Accuracy** |
|---|---|---|---|---|---|---|
| [171]/ FFF | Bayesian DP mixture model | Build Failure Detection | Accelerometer, Thermocouple, IR, Borescope | 85% (Average F-score) | × | - |
| [172] / FDM | Functional Qualitative Quantitative Model | Dimension, Surface | Accelerometer, IR, Thermocouple | - | × | ∼0.3 mm (median RMSPE) |
| [180] / FDM | Support Vector Machines | Abnormal Extrusion Detection | High-end Acoustic Sensor | 95% | × | - |
| [181] / FDM | First Principle Model of Filament | Dimension | - | - | × | ∼0.1 mm MAE |
| [173] / FFF | Online sparse estimation-based classification | Abnormal Extrusion Detection | Accelerometer, IR, Thermocouple | 90% (F-score) | × | - |
| [182] / FDM | Hidden semi-Markov model | Abnormal Extrusion Detection | High-end Acoustic Sensor | 91.9% (Accuracy rate) | × | - |
| [167] / FDM | Theoretical Model | Surface | - | - | × | 5.66% MAPE |
| [170] / SLM | Spectral Convolutional Neural Networks | Build Quality | High-end Acoustic Sensor | 79-84% | × | - |
| [183] / FDM | Heterodyne Technique - | Belt Fault Detection | High-end Acoustic & Piezoelectric Sensor | - | × | - |
| **QUILT** /FDM | Behavioral Modeling (Random Forest, Clustering) | Dimension | Low-end Acoustic, Accelerometer, Magnetic, Current | 83.09% (Classification Score) | ✓ | 0.59 mm MAE |

FDM: Fused Deposition Modeling SLM: Selective Lase Melting MAE: Mean Absolute Error
FFF: Fused Filament Fabrication RMSPE: Root Mean Square Percentage Error MAPE: Mean Absolute Percentage Error

tion. However, these methods do not consider checking the aliveness (up-to-date model) of the model and are mostly limited to anomaly detection. Each of these techniques has its own merit, hence, the proposed methodology is not intended to function independently but in conjunction with various approaches to fully realize the concept of *digital twin*.

## 7.5 Discussion

**Quality inference:** To validate the proposed methodology, the flow rate was used to detect anomalous system behavior and overall system degradation behavior. However, there can be multiple $PT_{system}$ parameters that might affect the quality. However, our methodology can be adjusted over time to consider variation in other $PT_{system}$ parameters over time. We have considered only dimension (thickness of a simple 3D object) as a quality metric. However, for building the full scale $DT_{system}$, multiple metrics are needed to be considered.

**More IoT sensors and placement**: In this work, sensors with a low sampling rate and resolution were used. The number of sensors was limited as well. To improve the accuracy of the *digital twin* techniques such as [157] needs to be incorporated for the development of IoT sensor arrays.

**Implementation using IoT device:** For building the $DT_{system}$ using IoT devices, further consideration is required for off-the-shelf and wireless IoT devices [157]. Hence, further analysis is required to understand the trade-off between power, time, and performance of the $DT_{system}$ in localizing and inferring the quality variation.

**More test cases:** One of the limitations of the experimental section was in using a limited number of test 3D objects for inferring the quality and localizing the faults. However, these 3D objects contain structures which provide large possible variation in G/M-code for building the digital twin models. Nonetheless, digital twin models will be more accurate if large test data are incorporated.

**Sensor fusion analysis:** In the motivation section, we provided mutual information analysis for individual side-channels. We acknowledge that a rigorous analysis of the calculation of mutual information by fusing these sensors may further justify the proposed methodology. The machine learning algorithm achieves this by carefully selecting the features extracted

Table 7.3: Other additive manufacturing technologies

| **Technology** | Source of analog emissions | | |
|---|---|---|---|
| | Acoustic | Vibration | Power |
| SLA | Build Platform, Stepper Motor | Sweeper | Motor Controller |
| SLS | Fabrication Piston | Rollers | Power supply |
| MJ | Build Tray, Jetting Head | Moving head, Blower, Position belt | Heater, Coil |
| SLM | Retractable Platform | Leveling Cylinder | Power Supply |
| EBM | Build Platform | Build Platform | High Voltage Cable |

SLA: Stereolithography
SLS: Selective Laser Sintering
MJ: Material Jetting
SLM: Selective Laser Melting
EBM: Electron Beam Melting

from the side-channels to build a model to get the highest possible accuracy. However, in our future work, we will dwell deeper in performing mutual information analysis to further clarify the contribution of individual side-channels in indirectly building the digital twin.

**Generalizability of the proposed method:** As a case study, we presented applicability of the proposed methodology in fused-deposition modeling based additive manufacturing. However, many other technologies also have analog emissions (see Table 7.3), which may be capable of aiding in the indirect method of building digital twin models. Hence, we hypothesize that the proposed methodology will be able to scale across multiple manufacturing systems.

## 7.6 Summary

In this chapter, we present a novel methodology to build a living *digital twin* of the fused deposition modeling technology based additive manufacturing system by utilizing various retrofitted low-end sensors available in IoT devices to indirectly monitor the system through various side-channels (such as acoustic, vibration, magnetic, and power). Based on these signals, a clustering algorithm is used to generate a fingerprint library, that effectively rep-

resents the physical status or the *physical twin* of the system. The *digital twin* is used for localizing the anomalous physical emissions that have the potential of resulting in quality variation. For localizing the error, the *digital twin* achieved an average accuracy of **83.09%**. Moreover, we also presented an algorithm for updating the *digital twin* and inferring the quality deviation. As a case study the *digital twin* modeling was performed on the additive manufacturing system. Compared to the state-of-the-art methods (which do not consider model aliveness), our methodology is able to update itself, infer quality deviation and localize anomalous faults in the additive manufacturing system.

# Chapter 8

# Part III: Graph Convolutional Neural Network

## 8.1 Introduction

So far in the previous chapters, we have utilized euclidean data for performing data-driven modeling. In part III of the thesis, we focus our attention on data-driven modeling algorithms for non-Euclidean data. We specifically focus on developing algorithms for handling non-Euclidean data present in cyber-physical systems.

A system design process consists of various steps such as problem definition, background research, requirement specification, brainstorming solutions, selecting the best solution, developing a prototype, testing, and finally redesigning [184]. During each of these steps, a wide variety of high volume and continuous data is generated. These design steps are repeated throughout various systems such as mechanical, electronic, software, etc. Due to the repetitive nature of these tasks, engineers can save a large amount of time if they can compare existing similar designs that closely match the desired functionality while designing

a new system. Rather than having to redesign, they can find functionally similar designs and modify such designs to fit their needs. Moreover, this can further lead to the creation of artificial intelligent assistants that assist human experts to design new systems faster.

The engineering design data varies from domain to domain. In electronic design, it consists of high-level design descriptions, register transfer level descriptions in Verilog or VHDL, schematics, etc. In mechanical design, it consists of data regarding structural designs, modeling, and analysis of components, etc. Moreover, there is a wealth of data generated throughout the supply chain of engineering including Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) tools. In order to perform meaningful learning from this data, we need to utilize non-euclidean or graph learning algorithms that are able to extract, categorize, and label these sparse data.

In this chapter, we propose to utilize a structural graph learning algorithm to abstract the detailed engineering data (such as configuration, code, hybrid equations, geometry, sensor data, etc.). This chapter expands the general view presented in [41]. To achieve this, we first represent the engineering data using a knowledge graph and then perform semi-supervised learning to be able to classify the sub-graphs based on their structural property and the corresponding attributes. This will allow engineers to compare and cluster functionally similar designs, configurations, codes, geometry, sensor data, etc. This in return will allow engineers to automate the process of quickly searching the required engineering designs available in their library that meets the functional requirements presented in the specification.

## 8.2 Related work

Design automation of engineering designs such as electronics, mechanics, etc., has seen an influx in the usage of machine learning and artificial intelligence algorithms [185, 186, 187,

188, 189]. These learning algorithms have helped design automation by making the design process easier, faster, and efficient. However, current approaches are mostly limited in the utilization of Euclidean domain data and algorithms.

Research on more general non-euclidean domain-based learning algorithms has recently gained momentum [190]. Moreover, a significant amount of work has been done in implementing the convolutional neural network on non-euclidean structure data and manifolds of 3D objects [191, 192, 193, 194, 195, 196, 197, 198]. These works can be divided into two general directions in which the learning algorithms have been implemented on non-euclidean data (such as a graph). The first direction is in the spectral domain, and the second direction is in the spatial or vertex domain. In spectral domain based analysis, just like how filter weight is learned in the traditional 2-dimensional convolutional neural network, filter kernels are learned. In order to do this, first, the graph is transformed into the Fourier domain by projecting the high-dimensional vertex domain graph to low-dimensional space using Eigen-basis of the graph Laplacian operation [199]. There are works where the various form of graph Laplacian operator and approximation methods for reducing the size of the graph kernel and the Fourier transformation of the graphs have been proposed[197, 198, 200, 201, 202]. In the second direction, first, the neighborhood information of a vertex is gathered using various techniques. This aggregated neighborhood information is then treated as features and different transformation on these features are proposed [203, 204, 205, 206, 39]. The major contribution of this work comes from the fact that the sampling and aggregation can focus on a node's neighborhood, thus not requiring the whole graph to be seen during the sampling and aggregation steps. The sampling and aggregation can be used either by using bread-first [203] or utilizing both bread-first and depth-first search [205]. These algorithms can effectively extract the node features based on their neighborhood which can be used to perform clustering and classification. Moreover, it has been shown to be effective in generating the graph embedding using the auto-encoders [206]. Furthermore, some work [202] have even proposed to combine the vertex domain and spectral domain approaches to utilize the

strengths of each domain.

Both the spectral domain and vertex domain based approaches have shown tremendous potential in generalizing the graph learning algorithms. However, each of these algorithms has a major weakness. The spectral domain based approach relies highly on the Laplacian matrix. It filters the graph by using the eigenvalues of the Laplacian operator. However, these Laplacian matrix is dependent on the graph and the filter weights trained in one graph cannot be applied to another one. On the other hand, the vertex domain approaches have shown high efficiency in node-level clustering and classification. Although, there are algorithms such as sub-graph2vec [**?** ], struct2vec [207], a more general CNN based approach for learning rich features from a sub-graph is lacking. In engineering design automation, we would require a more general graph learning algorithm that is able to learn sub-graph or whole-graph property for providing more intuitive functional classification, clustering, or even generation.

In addition, custom graph kernel modeling for mining the graph by measuring the structural similarity between the pairs of graphs have also been proposed [208]. The major limitation of this approach is that they do not consider the features of the individual nodes and only rely on the structural similarity of the graph. Hence, it may be useful in mining structures from design engineering data, however, it will not help if the similar structure have different node features (which normally is the case in the engineering data). To address the existing limitation in the graph learning algorithms, in this chapter we introduce the structural graph convolutional neural network that is graph invariant (unlike spectral domain approaches), can learn rich features from nodes, sub-graph or the whole graph, and is able to use both the structure and node features to learn rich features to classify and cluster different engineering domain graphs to aid in design automation.

## 8.3 Graph Learning using Convolutional Neural Network

Let us define some preliminary notations for explaining the graph structure. The graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices of the graph is denoted as $\mathcal{V}$ and the set of edges of the graph is denoted as $\mathcal{E}$. The graph edges can be weighted $w_{ij}, i, j \in \mathcal{V}$ and be directed. In this chapter, we will consider unweighted graphs with $w_{ij} = 1$. However, we may easily expand the graph learning algorithm for weighted graphs as well. When the engineering data is represented using such graph structure, each of the vertices will have some features (such as design version, mechanical properties, etc.). We represent such features for each vertices using the symbol $f_i$, where $i \in \mathcal{V}$. $f_i$ consists of a vector whose dimension depends on the amount of information present in the engineering data. The raw format of features may vary from being a text, image, continuous or discrete signals, etc. In such a situation, these features need to be converted to its corresponding vector representation using auto-encoders. The adjacency matrix of the graph $\mathcal{G}$ is denoted by $\bar{A}$. The sub-graph of $\mathcal{G}$ is denoted as $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where $\mathcal{V}_s \subseteq \mathcal{V}$ and $\mathcal{E}_s \subseteq \mathcal{E}$.

The proposed graph learning algorithm's architecture is shown in Figure 8.1. Before the algorithm can be used, the raw engineering design data needs to be converted into a knowledge-base graph [209]. The structure of the knowledge-based graph should be tailored to the engineering domain. The next step will involve converting the high-dimensional information present in each of the vertices and edges to a lower dimensional vector space embedding. These embedding will form the features $f_i$ for all the vertices. The major contributions of the proposed graph learning algorithm are (1) Neighbor Node Aggregation Layer, (2) Sub-graph Convolutional Kernel, and (3) Graph Pooling Algorithm. However, there are various components of the proposed algorithm which enable it to function. Each of these components is explained in the following subsections.

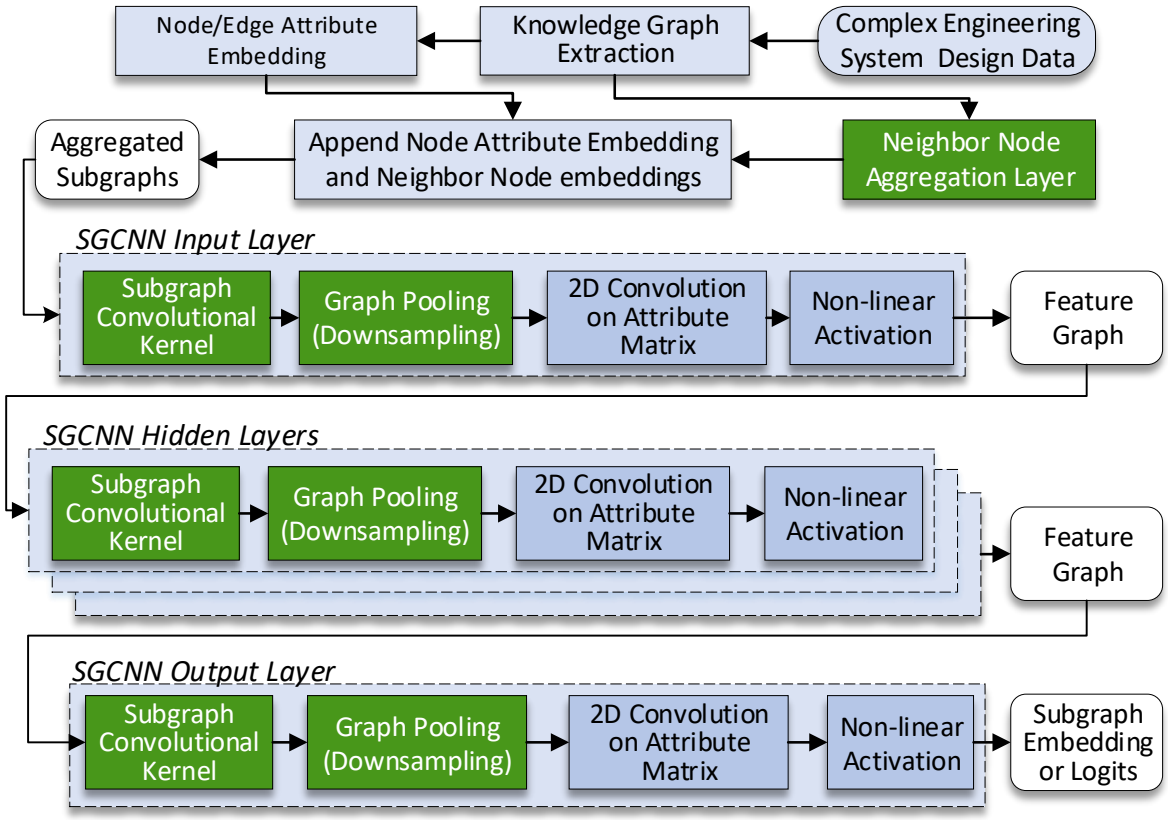



Figure 8.1: Proposed graph learning algorithm for engineering design automation

### 8.3.1 Knowledge Graph Extraction

The first task for utilizing the proposed graph learning algorithm is converting the complex engineering domain data into a meaningful data structure which can concisely and precisely represent the engineering domain data. For this purpose, we propose to utilize a knowledge graph. The knowledge graph stores information between the various uniquely identifiable entities and their corresponding relationship. These relationships are stored in the form of a triple (node-edge-node) relationship. This type of triples has previously been used to create knowledge graphs such as DBPedia [210]. The main advantage of using such knowledge graphs is that it can store rich engineering domain information and continuously evolve, grow, and be linked to other engineering domain data as well.

### 8.3.2 Attribute Embedding

After the knowledge graph has been created, we will have the structure of the graph ready to be utilized in the graph learning algorithm. However, in a knowledge graph, the node and edge may be in different data format (such as text, images, etc.). These high-dimensional attribute of the nodes and edges needs to be converted into a low dimensional feature embedding. To embed such attributes we utilize various state-of-the-art auto-encoders. For example, for encoding the text we will utilize word2vec [211]. For embedding images, we will utilize existing deep auto-encoder [212]. These vector embedding generated from the attribute of the nodes and edges form the feature which is utilized in the attribute matrix described in the Section 8.3.4.

### 8.3.3 Neighbor Nodes Aggregation

One of the fundamental tasks in performing graph learning in engineering domain data is being able to capture the features of the vertices with respect to its location in the graph. Each of the vertices not only has special topology but also share a set of attributes across the knowledge graph. Hence, it is necessary to capture each of the unique structural and feature based relation of vertices with respect to its neighbor. In order to do this, in the proposed graph learning algorithm, we utilize concepts similar to the vertex domain approach [203] (see Figure 8.2). Before the neighbor node aggregation is performed, a user-defined query or a *schema* is used to induce a sub-graph. This induction process is domain specific and can further be tailored to aid the graph learning algorithm by inducing sub-graphs that capture the meaning engineering design. For example, the schema can be used to induce graphs that certain keyword (such as engines, valves, etc.,) if the graph learning is applied to engine design data. Based on this schema various instances are generated by the sample generator. These induced samples are then passed to two blocks in parallel. One block

converts the node/edges attribute to its corresponding vector form, whereas the other block is the neighbor node aggregation layer. In the neighbor node aggregation layer, for the induced sub-graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, our algorithm performs both breadth-first and depth-first search to collect the neighbor nodes of the give sub-graph. We utilize the parameter $d$ as the depth to search in the graph, and $n$ as the number of paths to be searched. The neighbor nodes borrow the concept from node2vec [205], where a balance is maintained between the depth and the bread for context generation. This balance helps us to maintain a balance between the local and non-local neighbors to the sub-graph.

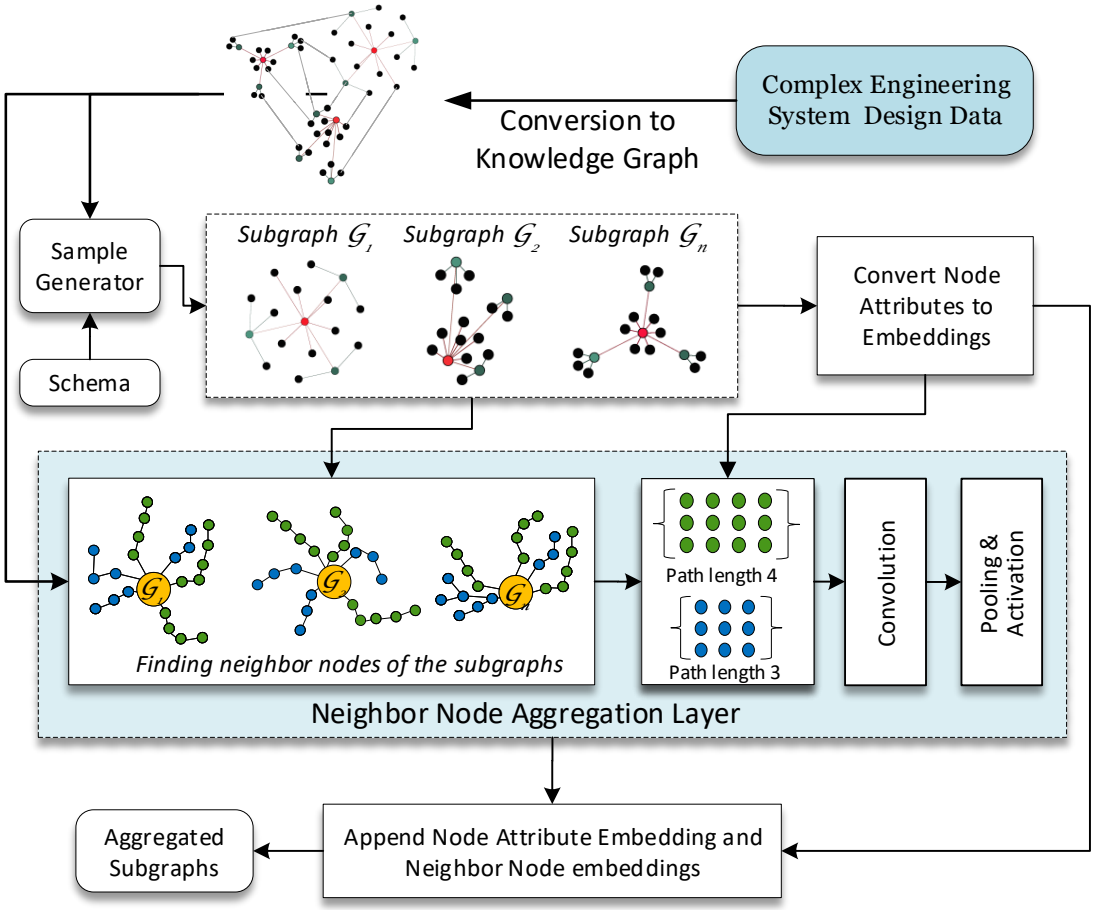



Figure 8.2: Proposed neighbour node aggregation layer architecture

In order to find the neighbors, for all $v_i \in \mathcal{V}_t$, we search the original knowledge graph $\mathcal{G}$ to find $n$ paths of length $d$. Let us denote each of these paths by $\mathcal{P}_i^d$, where $d$ denotes the path length, and $i$ denoted the $i^{th}$ path of length $d$. For the path to be considered for aggregation

the nodes lying in the path $v_i \notin \mathcal{V}_t$. We acquire vector of paths $\mathcal{P}^d = \{\mathcal{P}_1^d \mathcal{P}_2^d, ...\}$ for each of the sub-graph $\mathcal{G}_t$. However, finding and using all of such paths is non-trivial, hence we then sample $s$ paths of length $d$ for each of the sub-graph. This sampling is done randomly in the proposed algorithm. Hence, $s$ is another input parameter to the propose graph learning algorithm. Using these randomly sampled paths, we form a neighbor feature matrix $\bar{N}$. The bar is $s$ by $d$ matrix with each element having a feature vector having extracted earlier. The number of paths found in $\mathcal{P}^d$ may be smaller than $s$, $\mathcal{P}^d$ can be padded to make $\bar{N}$ with at least $s$ number of rows/paths. The algorithm for neighbor node aggregation is presented in Algorithm 13.

**Algorithm 13:** Neighbor node aggregation algorithm

**Input:** Induced Sub-graphs: $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t$
**Input:** A list of depth to search: $D = d_1, d_2, ..., d_n$
**Input:** A list of sample numbers per depth: $S = s_1, s_2, ..., s_n$
**Output:** A feature vector: $x_n$

```
foreach d_i ∈ D do
    foreach v_j ∈ V_t do
        Find all length d_i paths P_j^{d_i}
        Remove paths containing nodes in V_t from P_j^{d_i}
        Add P_j^{d_i} into P^{d_i}
    Construct N̄ by randomly selecting s_i paths from P^{d_i}
    Extract feature x_{di} from N̄
x_n = f^d_{pool}(x_{d1}, x_{d1}, ..., x_{dn})
return x_n
```

The input to the Algorithm 13 is the knowledge graph $\{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_t\}$ using the user-defined schema, list of depth to search $D = \{d_1, d_2, ..., d_n\}$, list of sample numbers per depth $S = \{s_1, s_2, ..., s_n\}$, and feature vector $x_n$ for the vertices. The output of the Algorithm 13 is the extracted feature to be appended to the vertices of the sub-graph $G_t$. One of the challenges in extracting the features from the neighborhood nodes as discussed in in [203] is the fact that the non-Euclidean data has no natural ordering. Which means that the feature extraction should be applied over an unordered set of paths to make sure that the arbitrary change in the order of rows of the matrix $\bar{N}$ still results in the same feature being extracted. In order to achieve this in Algorithm 13, we first apply a general 1-D convolution operation

with trainable 1 by $d$ weight matrix $\bar{W}$ on $\bar{N}$ and then utilize a symmetric pooling function to extract the neighbor nodes' feature vector $x_n$ as follows:

$$x_n = \sigma(f_{pool}(\bar{W} \circledast \bar{N}) + b) \tag{8.1}$$

This equation is used in Line 8 of the Algorithm 13. The $b$ in Equation 8.1 is a bias variable and $\sigma$ is an activation function (e.g. ReLU, LeakyRelu, Sigmoid, Tanh, etc.,), and $f_{pool}$ is a pooling function. As mentioned, the pooling function has to be invariant to permutation of rows in $\bar{N}$. To achieve this, pooling function such as a mean operator can be applied over all the rows in the matrix, or we may utilize a max pool operator that extracts max values out of all the rows. The use of a specific pooling function is treated as a hyper-parameter in the graph learning algorithm and later configured in the training and hyper-parameter tuning process. As shown in Figure 8.2, we extract various path length $\{d_1, d_2, d_3, ..., d_k\}$. These path lengths will integrate various topological and localized attributes of the engineering knowledge graph. Hence, Algorithm 13 will return feature vector $\{x_{d1}, x_{d2}, x_{d3}, ..., x_{dk}\}$ depending on total number of path lengths extracted from the graph. Each of these path lengths is aggregated to the sub-graph. If the extracted path length number is large, we may perform pooling to reduce the dimension of the extracted features as:

$$x_n = f^d_{pool}(\{x_{d1}, x_{d2}, x_{d3}, ..., x_{dk}\}) \tag{8.2}$$

The returned neighborhood feature vector is then concatenated to all the feature vectors of the vertices $v_i \in \mathcal{V}_t$ as $x_{agg_i} = \{x_i, x_n\}$. Since the neighborhood aggregation layer is part of the graph learning algorithm, all the parameters of the neighborhood aggregation layer (such as weights of the 1-D convolution neural network) are learned during training. This allows the algorithm to automatically focus on relevant neighborhood node features to aggregate during the training.

### 8.3.4 Structural Graph Convolutional Neural Network Layers

The structural graph convolutional neural network (SGCNN) layers are shown in Figure 8.3. The input of the SGCNN are the aggregated sub-graphs that were generated by the neighbor node aggregation layer described in Section 8.3.3 earlier. Each of these aggregated sub-graphs is passed in batches with individual vertices having a feature matrix. The SGCNN layers consist of input, hidden and output layers. These layers abstract the aggregated graph's feature in each layer just like the traditional 2D-convolutional neural networks. Each of the SGCNN consists of various components: (1) Sub-graph Convolution Kernel, (2) Graph Pooling, (3) 2D Convolution on Adjacency Matrix, (4) New Adjacency Matrix Calculation, and (5) non-linear activation. We will discuss each of these components in details in the following sections.

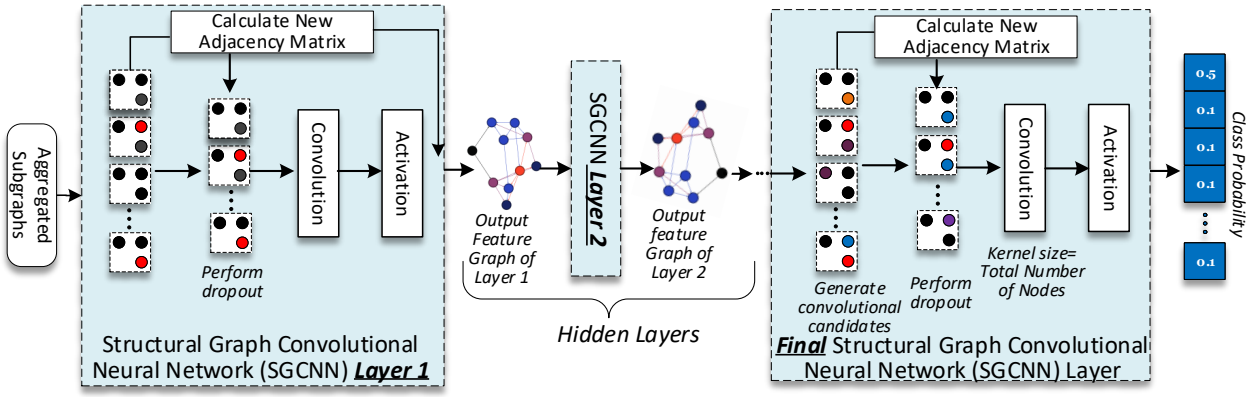



Figure 8.3: Proposed structural graph convolutional neural network architecture

#### Sub-graph Convolution Kernel

The main block of the SGCNN is the sub-graph convolutional kernel. The task of the sub-graph convolutional kernel is to abstract meaningful feature vectors from the aggregated graph. The kernel receives the aggregated graph $\mathcal{G}_t$ with the corresponding feature matrix $\bar{X}$, which consists of the aggregated features $x_{agg_i} = \{x_i, x_n\}$ for $v_i \in \mathcal{V}_t$. It also receives the adjacency matrix $\bar{A}$ of the aggregated graph $\mathcal{G}_t$. With the adjacency matrix and the feature

matrix, the first step the sub-graph convolutional kernel block will do is create the adjacency matrix $\bar{A}r$ by taking the Hadamard Product between $\bar{X}$ and $\bar{A} + \bar{I}$ as follows:

$$\bar{A}r = \bar{X} \circ (\bar{A} + \bar{I}) \tag{8.3}$$

We have added the identity matrix $\bar{I}$ to make sure that the vertices do not lose their feature information. This self-loop to each of the vertices in the sub-graph makes sure that each vertex retain their own information while calculating the Hadamard Product. An example of an attribute matrix is shown in Figure 8.4. The attribute matrix $\bar{A}r$ is used to define

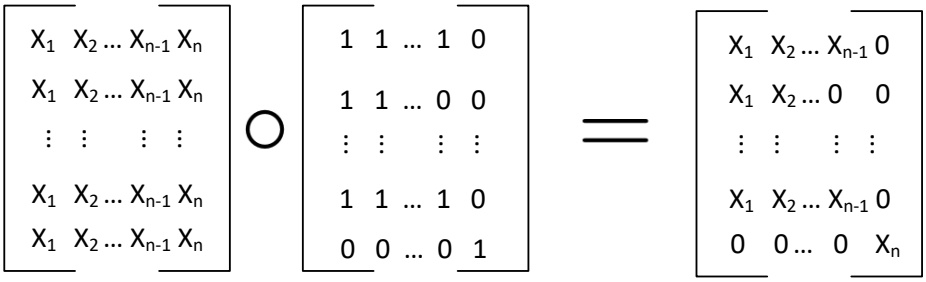



Figure 8.4: Example of an attribute matrix calculated using Hadamard product

a graph-convolutional kernel. The kernel consists of a $k$ by $k$ weight matrix $\bar{W}^k$. The size of the $k$ can be varied just like the 2D-convolutional kernels. The convolutional is applied between the $\bar{W}^k$ and $\bar{A}r$. In traditional 2D-convolution, a kernel is slid from left to right and top-down in an order to perform the 2D convolution. However, since there is no notion of 2D-grid structure in the graph, we cannot perform convolution in a similar manner. Hence, in our graph-convolution, we propose a new method to perform a convolution operation for the graph data structure.

The algorithm for performing the graph-convolution is presented in Algorithm 14. The first step in performing the graph convolution involves generating candidate graph kernels with size $k$ by $k$, where we select the $k$ vertices at a time. This candidate graph is then convoluted with the weight matrix $\bar{W}^k$. In order to generate the candidate graph kernels, we use the fact that removing the $i^{th}$ row and $i^{th}$ column in $\bar{A}r$ is equivalent to removing the vertex $i$

from $\mathcal{G}_t$. Hence, assuming that the total vertices in the induced aggregated sub-graph $\mathcal{G}_t$, let it be denoted by $n$, is greater than the size of the kernel $k$, we will remove $n-k$ number of vertex from $\mathcal{G}_t$. The new sub-graph will be left with a new $k$ by $k$ attribute matrix $\bar{A}r^k$. The drawback of generating the attribute matrix like this is that there are $O\binom{n}{k}$ possible ways to generate the attribute matrix $\bar{A}r^k$. This might be okay for an induced graph with a lower number of nodes, however for engineering data the induced graph size can have a large number of nodes. And normally the size of the $k << n$, making the complexity of generating the $\bar{A}r^k$ very high. Hence, to tackle this impractical $\bar{A}r^k$ generation step, we propose to relax it by only picking $s$ number out of $O\binom{n}{k}$ as a convolution candidate. Hence, the total number of the possible $\bar{A}r^k$ is thus reduced to $O(s)$. By doing this, we make it feasible to generate the $\bar{A}r^k$ from the $\mathcal{G}_t$ as a potential candidate of graph kernels to be convoluted with the $k$ by $k$ weight filter matrix $\bar{W}^k$. The procedure of down-sampling of $O\binom{n}{k}$ possible $\bar{A}r^k$ values to just s is explained in Section 19).

**Algorithm 14:** Graph convolution calculation algorithm

**Input:** An input graph: $\mathcal{G}_t$ with $n$ vertices
**Input:** A convolution kernel: $\bar{W}^k$
**Input:** Sample size: $s$
**Output:** An output feature graph: $G'$

Generate adjacency matrix $\bar{A}$ from $G$
Using the same vertices order to generate list of features $\mathcal{X}$
Create feature matrix $\bar{X}$ with $n$ rows, and each row being $\mathcal{X}$
$\bar{A}r = \bar{X} \circ (\bar{A} + \bar{I})$
$m = \binom{n}{k}$
$CombList$ = Enumerating choice of $n-k$ elements from $1, 2, ..., n$
**foreach** $comb \in CombList$ **do**
    $\bar{A}r_{comb}$ = remove rows and columns list in $cand$ from $\bar{A}r$
    Add $\bar{A}r_{comb}$ into $CandList$
**if** $m > s$ **then**
    Down-sample $CandList$ to $s$ elements
**else if** $m < s$ **then**
    Pad $CandList$ to $s$ elements
**foreach** $cand \in CandList$ **do**
    $x^k = \bar{W}^k \circledast cand + b$
    Add new vertex $v_k$ into $G'$
    Add feature vector $x^k$ on $v_k$
    Connect $v_k$ based on $cand$'s connection in $G$
**return** $G'$

Algorithm 14 presents the graph convolution steps in detail. The input to the graph is the induced aggregated graph $\mathcal{G}_t$. The output of the algorithm is the new graph where each node represents the merged vertices present in $\bar{A}r^k$. In Line 1, we first generate the adjacency matrix from the graph. In Line 4, the attribute matrix is generated by performing the Hadamard product between the feature matrix and the adjacency matrix. In Line 7, possible combination of $\bar{A}r^k$ is listed and down-sampled in Line 10. In Line 14, a 2D convolution is performed between the $\bar{A}r^k$ and the filter weight matrix $W^k$. Finally, the new graph is returned in Line 19. Each of the SGCNN layers will generate a new graph which progressively abstracts and fuses the topological and attributes of the previous graph.

**Graph Pooling Algorithm**

When we generate $\bar{A}r^k$, there are $O\binom{n}{k}$ possible in the beginning. After down-sampling it to $s$ number of $\bar{A}r^k$, the next stage will have $O\binom{s}{k}$ possible combination of $\bar{A}r^k$. One of the desired property of convolutional neural networks is to be able to perform the convolution over a deep number of layers. Hence, to be able to perform the graph convolution in deeper layers, we need to perform down-sampling or pooling at each layer to manage the size of the possible $\bar{A}r^k$ matrices generated from the graph. Without down-sampling, it will be unfeasible to perform a large number of convolutional operation between $\bar{A}r^k$ and $W^k$. Hence, we propose to perform the pooling operation before the convolution operation in each of the SGCNN layers. The proposed down-sampling/pooling operation utilizes the topology of the graph to remove samples from combinations of $\bar{A}r^k$. For each of the possible $\bar{A}r^k$, we calculate the corresponding total degrees. The intuition behind the proposed down-sampling algorithm is that out of $O\binom{n}{k}$, due to the sparsity of the aggregated sub-graph, combination of various nodes will not have any edges or lower number of edges among them. Hence, combining them together to perform convolution will be less fruit-full than selecting the combination of $\bar{A}r^k$ that have higher connectivity among the vertices. The proposed down-sampling algorithm

is presented in Algorithm 15.

**Algorithm 15:** Graph pooling algorithm

**Input:** An input graph: $\mathcal{G}$
**Input:** The list of the candidate nodes combinations: $Comb$
**Input:** Sample size: $s$
**Input:** Dropout Rate: $d$
**Output:** The list of down-sampled candidate nodes: $Comb'$

```
1  Randomly sample Comb' from Comb using d
2  Generate adjacency matrix Ā from G for only Comb'
3  foreach c ∈ Comb' do
4      d_c = 0
5      foreach n ∈ c do
6          Calculate Degree of n as d_n
7          d_c = d_c + d_n
8      foreach c' ∈ Comb' do
9          if c' is connected with c in Ā then
10             d_c = d_c + 1
11 Keep the s number of nodes with the highest degrees and store in Comb''
12 return Comb'';
```

The input to the Algorithm 15 is the graph $\mathcal{G}$, the possible list of candidate combination $Comb$, the pooling sample size $s$, and the dropout rate $d$. The output of the algorithm is the list of combination $Comb''$ which is down-sampled. To achieve this, first in Line 1, it randomly samples $Comb'$ by using the dropout rate $d$. This is necessary because it is computationally unfeasible to calculate the adjacency matrix in the next step for all the possible candidates in $Comb$. In Line 2, we generate the adjacency matrix $\bar{A}$ for the new candidate combination of nodes. This adjacency matrix carries the graph structural data and passes it through the deeper layers. This step is important, as the different layers will be able to abstract the graph data in a hierarchical manner. Lines 3-10 compute the total degrees of each candidate nodes combination $Comb'$, which are combinations of the nodes in $G$ that are generated by the convolution kernel and these combinations will serve as the new nodes of the output feature graph after the graph convolution kernel. Specifically, lines 3-7 compute the total degrees inside the combination, and lines 8-10 compute the degrees in between different combinations. Finally, we keep the $s$ number of nodes combinations which have the highest degree and remove the rest. We significantly reduce the size of the graph

convolution kernel by dropping the combinations in the calculated degrees using the max pooling. Nevertheless, we ensure that the convolution is performed on the graph structures with higher connectivity.

### 2D Convolutions on Attribute Matrix

After we have down-sample $s$ possible candidate $\bar{A}r^k$ matrices, we apply simple 2D convolution operation to extract the feature vector for the give combination of the $\bar{A}r^k$. The convolution operation can be written as follows:

$$x^k = \phi(\bar{W}^k \circledast \bar{A}r^k + b) \tag{8.4}$$

Where $\phi(.)$ is a non-linear activation function. We will have $s$ extracted feature vectors as: $x_1^k$, $x_2^k$, ..., $x_s^k$. We consider the extracted feature vectors $x_1^k$, $x_2^k$, ..., $x_s^k$ as a new feature graph $\mathcal{G}'$ with $s$ number of vertices, and $x_i^k$ as the feature vector for node $i$. An example of this process is shown in Figure 8.5. Given the input graph with 4 vertices, a 3-by-3 convolution kernel is selected. As a result, 4 convolution candidates are generated, down-sampled, and adjacency recalculated resulting in a new graph with 3 vertices.

### New Adjacency Matrix Calculation

For the deep SGCNN architecture to work, we have to keep track of the adjacency of the new $\mathcal{G}'$ generate at each of the layers. This $\mathcal{G}'$ is used as input to another sub-graph Convolution Layer to form a deep SGCNN model. The constructed adjacency matrix will allow the next SGCNN layer to recalculate the new attribute matrix and the potential candidates of $\bar{A}r^k$ to be selected for convolution and down-sampling.To calculate the new adjacency for the new graph, we check the edges between inter and intra nodes of the graph convolution kernels.

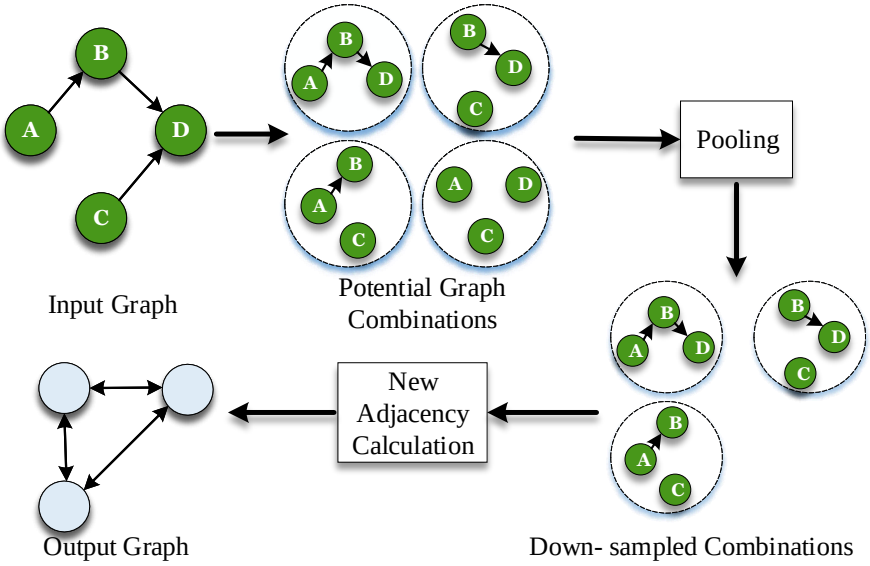



Figure 8.5: Convolution kernel example

The edge between these intra and inter nodes allow the graph structure to be propagated through the SGCNN layers, making sure the topological information is being utilized to learn the filter weight matrix $W^k$ at each layer.

### 8.3.5 Classification for engineering design abstraction

Given a large graph and labeled sub-graphs representing engineering design, the SGCNN can be used to classify them. While classifying these sub-graphs, various features regarding the design are learned by the graph convolutional kernels. Once the design and the labels (which can be the function served by the design) are trained, the design is abstracted based on their functions. This classification is done by using a softmax function and cross-entropy of the logits. In addition, the feature vectors (sub-graph embedding) generated by the final SGCNN can also be used by clustering algorithms to identify nearest neighbors sub-graphs that have an equivalent function in the graph based on their node attributes and structure. In engineering, there are several use-cases for sub-graph embedding including the identification of functionally equivalent structures that engineers are unaware of, and to identify structures that mislabeled.

### 8.3.6 Graph Learning Algorithm Hyper-parameters

In deep learning convolutional neural networks, hyper-parameters play a crucial role in improving the accuracy and convergence of the algorithm. These hyper-parameters are difficult to derive during the training as it requires large resources. Hence, to ease the resources they are instead selected prior to training. Grid search or other efficient approaches are utilized to fine-tune the hyper-parameter values. For Euclidean domain, extensive study [213, 214] have been carried out to aid in the hyper-parameter selection. Some of the hyper-parameters that even common to non-Euclidean deep graph convolutional neural networks are *activation function, hidden layers, number of iteration, learning rate, and batch size*. However, as the propose deep graph learning algorithm for the engineering design automation is relatively new, there are few other hyper-parameters that need to be highlighted. These hyper-parameters are as follows:

**Path length in node aggregation layer:**

As mentioned in Section 8.3.3, the neighbor node aggregation layer utilizes a specific path number of various length size. The path length determines how much of the knowledge graph should be considered in embedding the feature for the give induced sub-graph. If the length is small then we will focus on the local structure and attributes of the sub-graph, while selecting longer path length will allow the sub-graph to be embedded with global features. Hence, this value needs to be optimized and fine-tuned according to the specific engineering domain data.

**Graph convolution kernel size:**

The graph convolutional kernel size determines how many vertices to be considered for convolution with the filter weight matrix at each layer. The smaller kernel size means that each of the layers will abstract sub-graph feature by considering its immediate neighbors. However, if the kernel size is small while the number of vertices in the induced sub-graph is large than deeper SGCNN layers may need to be implemented. However, if we select larger kernel the SGCNN layer may be shallow. The size of the kernel may depend on the induced graph's size and structure and will require tuning before the training is performed.

**Dropout of candidate kernels:**

As presented in Section 19, we have used dropout to make the deep graph learning feasible. We have combined the random and degree-based dropout. If permitted by the resource, taking a large number of candidate attribute matrix may be helpful to better abstract the induced sub-graph. Due to the sparse nature of the engineering design data taking large combinations of attribute matrix for convolution may also be a waste of resources. Hence, a number of candidate kernels to drop out before the convolution is performed needs to be tuned as a hyperparameter.

## 8.4 GrabCAD Dataset

To demonstrate the applicability of the proposed graph learning algorithm in engineering design automation, we have selected the 3D engineering CAD models as training as a testing dataset for engineering design functionality classification. We have extracted the dataset from GrabCAD [1], which is an online repository of 3D CAD models shared and maintained by

[1] `https://grabcad.com/`

an online community of designers, engineers, and manufacturers. It consists of over 4 million members with over 2 million engineering design models. From this vast online dataset, we have extracted meta-information from six functional categories of 3D CAD models. These functional category are *Car*, *Engine*, *Robotic arm*, *Airplane*, *Gear*, and *Wheel*.

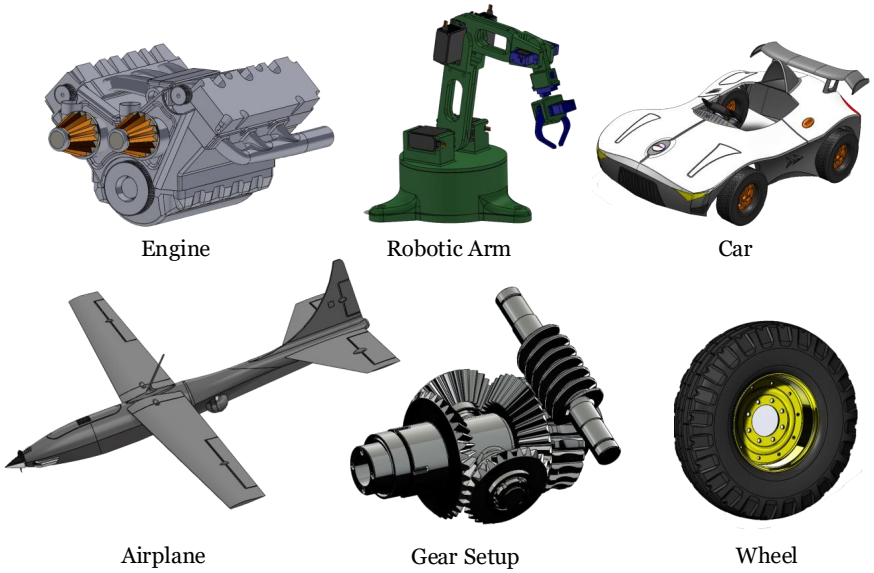



Figure 8.6: Sample of the 3D CAD models extracted from GrabCAD repository

Since these are engineering designs are from the mechanical domain, we have tailored a custom schema consisting of the properties such as 3D model's name, author of the design, description of the model, parts names, tags, likes, time-stamps, and comments on the engineering design. With this schema we have induced sub-graphs for each of the categories with 2,271 samples for *Car*, 1,597 samples for *Engine*, 2,013 samples for *Robotic arm*, 2,114 samples for *Airplane*, 1,732 samples for *Gear*, and 2,404 samples for *Wheel*. The induced sub-graph consists of 17 nodes consisting of both social network data (such a user-to-user interaction through comments and likes) and engineering data (such as model-to-tags relationship, model-to-model relationships, etc.). By inducing a sub-graph in GrabCAD dataset, we aim to demonstrate that similar designs with functional description represented by a knowledge graph can be efficiently classified using the proposed supervised graph learning algorithm.

Table 8.1: Accuracy for various hyper-parameters

| | | | | | |
|---|---|---|---|---|---|
| **Learning Rate** | 1.00E-04 | 1.00E-03 | 1.00E-02 | 1.00E-01 | *Hidden Layer Feature=100, Epoch =100, Output Layer Kernel Size=25, Batch size =64* |
| **Accuracy (%)** | 45.70 | 66.051 | 79.26 | 80.15 | |
| **Batch size** | 32 | 64 | 128 | 256 | *Hidden Layer Feature=100, Epoch=100, Learning rate=1e-2, Output Layer Kernel Size=25* |
| **Accuracy (%)** | 79.22 | 78.65 | 76.7 | 72.01 | |
| **Hidden Layer Feature Size** | 20 | 50 | 100 | 150 | *Output Layer Kernel Size=25, Epoch=100, Learning rate=1e-2, Batch size =64* |
| **Accuracy (%)** | 82.28 | 83.10 | 85.54 | 87.89 | |
| **Epochs** | 100 | 200 | 300 | 500 | *Hidden Layer Feature=100, batch=64, Learning rate=1e-2, Output Layer Kernel Size=25* |
| **Accuracy (%)** | 78.87 | 79.68 | 80.43 | 81.12 | |
| **Output Layer Kernel** | 25 | 30 | 40 | 45 | *Hidden Layer Feature=100, Epoch=100, Learning rate=1e-2, Batch size=64* |
| **Accuracy (%)** | 85.61 | 85.22 | 86.43 | 87.92 | |

Three Layers (Aggregate, SGCNN Input, SGCNN Output), Random Dropout

## 8.5 Results

In our experiment, the total number of the induced graph from all the engineering design is 14,131. From this sample, we have used 11,304 as training samples and 2,827 samples for testing if the similar functional designs get classified accurately. In order to make sure that proper hyper-parameters are selected and tuned, we have used a grid search approach over the possible hyper-parameter values. The result of hyper-parameter selection is shown in Table 8.1. In the table, we have tested the accuracy of various hyper-parameters. The first hyper-parameter is the learning rate of the optimization algorithm. We have used ADAM optimizer [215] to adjust the filter weights. From the table, it can be seen that the learning rate of 0.01 is able to achieve higher accuracy. The second hyper-parameter in the table is the batch size. The batch size determines how many of the induced sub-graph are passed together once for calculating the loss and updating the gradient.

It can be seen the batch size of 32 is able to obtain the highest accuracy of 79.22% classification accuracy. The next hyper-parameter is the hidden layer feature size. Each of the SGCNN layers is able to determine the size to give as an output. It can be noticed that the higher feature size of 150 is able to achieve better accuracy. With a larger feature size, we also increase the number of parameters in the filter weight matrix. The nest hyper-parameter is the epoch (the total number of times the training goes over the whole dataset). It can be seen that a higher epoch number of 500 is able to achieve higher classification accuracy. The last hyper-parameter is shown in Table 8.1 is the output layer kernel size. The final output layer of the SGCNN acts like a dense layer. Which means that it will try to generate probability values for each of the categories. The final layer of the kernel depends on the dropout carried out earlier. From the table, it can be seen that lower dropout or larger output kernel produces higher accuracy. Beside these hyper-parameters, we have also tested other hyper-parameters which are presented in the following sections.

### 8.5.1 Activation functions

Activation functions are used before the output in each of the SGCNN layers. Most of the operation before the activation function is mostly linear. However, the activation function increases the capacity of the SGCNN layer by making it non-linear. We have explored various activation functions which are well studied in Euclidean domain. These activation function are *sigmoid*, *softplus*, *tanh*, *rectifier linear unit*, and *leaky rectifier linear unit*. The training loss and engineering design classification accuracy is during testing is shown in Figure 8.7. The configuration for acquiring the results consists of layers size of 2 consisting of aggregate and embedding layer only. It can be seen that out of all activation functions, leaky rectifier linear unit ($f(x) = \alpha x$ for $x < 0$ and $f(x) = x$ for $x >= 0$) activation function (with $\alpha = 0.2$) is able to achieve lower loss and higher classification accuracy compared to other activation functions. Although the accuracy is higher, it introduces some noise in both the

loss and the accuracy values. One of the parameters that can be used in *LeakyRelu* is the alpha value. It determines the slope to be used to cut off the values when $x < 0$. Further analysis is required to tune the value of the $\alpha$.

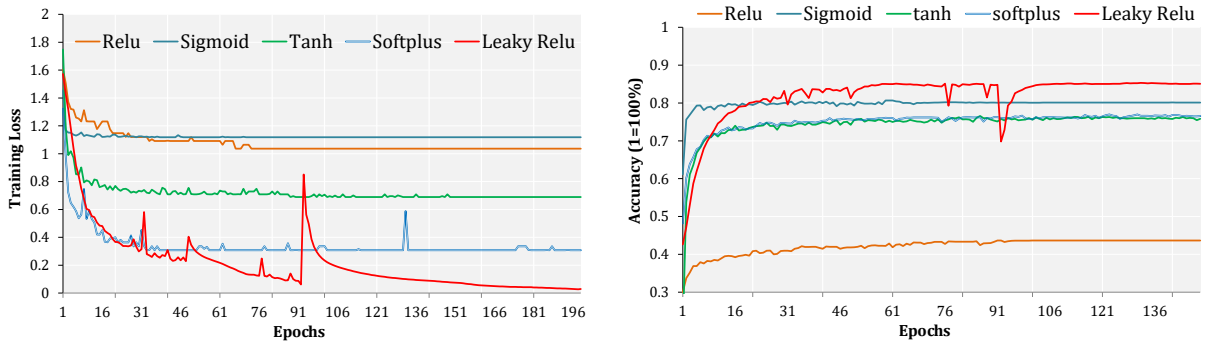



Figure 8.7: Training loss and accuracy for different activation functions

The training loss and engineering design classification accuracy is during testing is shown in Figure 8.7. The configuration for acquiring the results consists of layers size of 2 consisting of aggregate and embedding layer only. It can be seen that out of all activation functions, leaky rectifier linear unit ($f(x) = \alpha x$ for $x < 0$ and $f(x) = x$ for $x >= 0$) activation function (with $\alpha = 0.2$) is able to achieve lower loss and higher classification accuracy compared to other activation functions. Although the accuracy is higher, it introduces some noise in both the loss and the accuracy values. One of the parameters that can be used in *LeakyRelu* is the alpha value. It determines the slope to be used to cut off the values when $x < 0$. Further analysis is required to tune the value of the $\alpha$.

### 8.5.2 Kernel Size

One of the important hyper-parameters for the SGCNN layers is the size of the kernel used to perform the convolution with the attribute matrix $\bar{A}r^k$. The value of $k$ determines the total number of vertices that are considered at a time to perform the convolution with the filter weights $W^k$. In our experiment, the total of nodes available in the induced sub-graph is 17, hence, we have selected kernel size as 2, 4, 6, 8, 10, 12, and 14. It can be seen

from Figure 8.8 that the kernel size has a drastic effect on the classification accuracy of the engineering design data. The configuration of SGCNN consists of random dropout, last layer kernel size of 5, three total layers, relu as hidden layer activation, and leaky relu as the final layer activation function. For three layers of SGCNN, larger kernel size is able to achieve higher testing accuracy and lower training loss. However, having a larger kernel size in every hidden layer is not feasible, as it increases the complexity of the training algorithm. In Section 8.5.4, it can be seen that with deeper layers, a smaller kernel size is able to obtain higher testing classification accuracy as well.

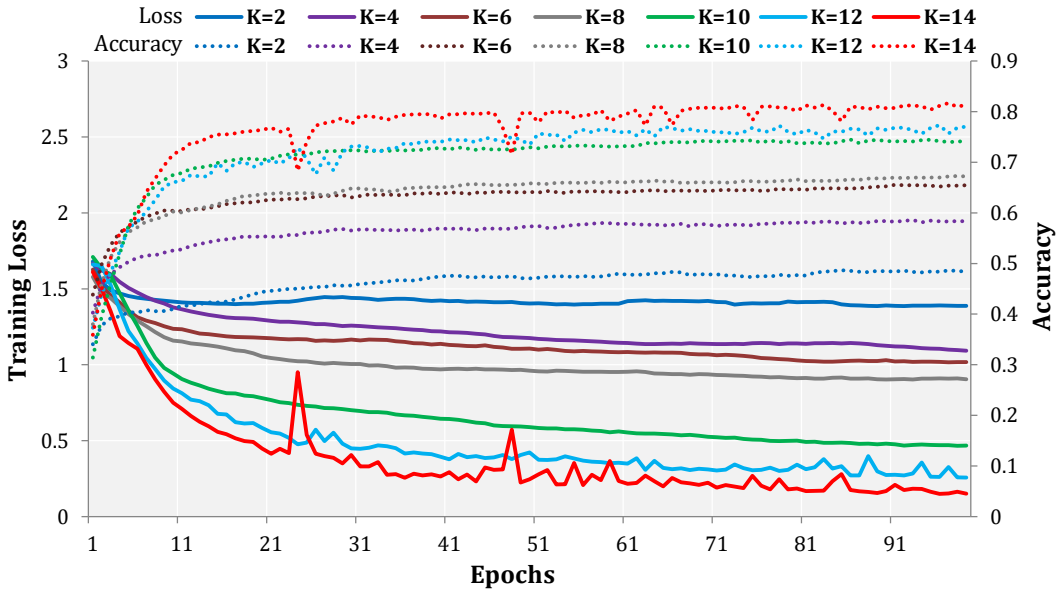



Figure 8.8: Training loss and accuracy for different size of kernels

### 8.5.3 Dropout

As mentioned earlier, without dropout, it becomes unfeasible to create deep SGCNN layers. To improve the scalability of the SGCNN layers, in the proposed graph learning algorithm we utilized a combination of random and adjacency based dropout. In the adjacency based dropout, we utilize the degree of the vertices to select the candidate combination of $\bar{A}r^k$ for the convolution. The dropout rate determines the size of the final kernel size. If the dropout

rate is higher the size of the kernel at the output SGCNN layer will be smaller and vice versa. The result of the various size of the final layer's kernel size as a result of changing the dropout is shown in Figure 8.9. The SGCNN configuration consists of hidden layer kernel size of fourteen, total of three layers, relu as hidden layer activation, and leaky relu as the final layer activation function. It maybe notices that if the dropout is less (resulting in larger kernel size in the output layer) the testing accuracy is higher and training loss is lower. The result is shown for just three layers of the SGCNN. The total number of possible combination of $\bar{A}r^k$ with the kernel size of 14 for the first layer is $\binom{17}{14} = 680$. We can notice that even when the total candidates have been drastically dropped to just 5, 10, 15, and 20, the graph learning algorithm is able to perform quite well in classifying the engineering designs. This may be due to the fact that the induced graphs are sparse in nature and that the initial kernel size of 14 is able to capture all of the node's features during convolution.

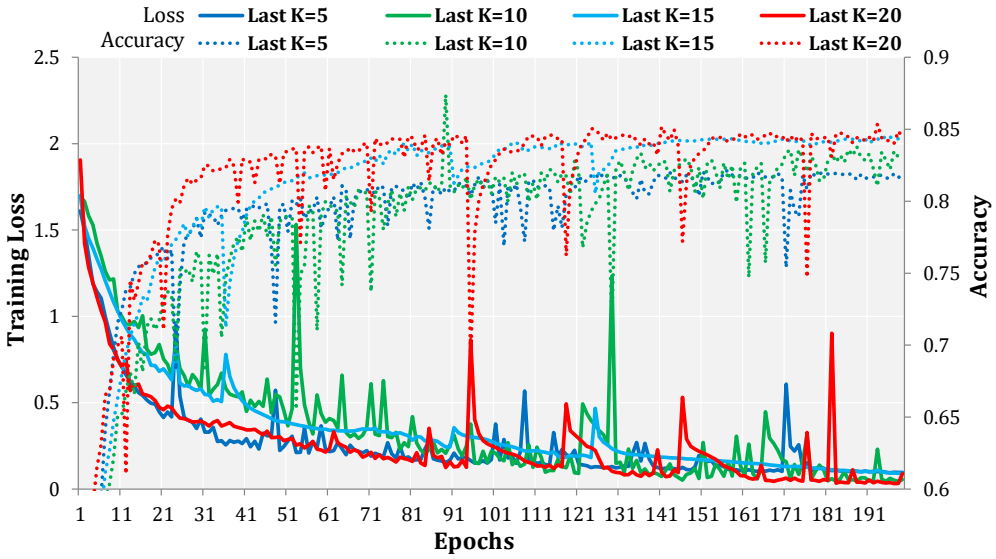



Figure 8.9: Training loss and accuracy for various random dropouts

In addition to random dropout, we have implemented the adjacency based dropout as well. The down-sampling algorithm first uses random dropout to initially reduce the possible candidates of $\bar{A}r^k$, and out of the remaining selects the ones with higher connectivity. The result for the down-sampling combined with the random dropout is shown in Figure 8.10.

It maybe notices that adjacency based dropout is able to achieve the highest accuracy of around 90%.

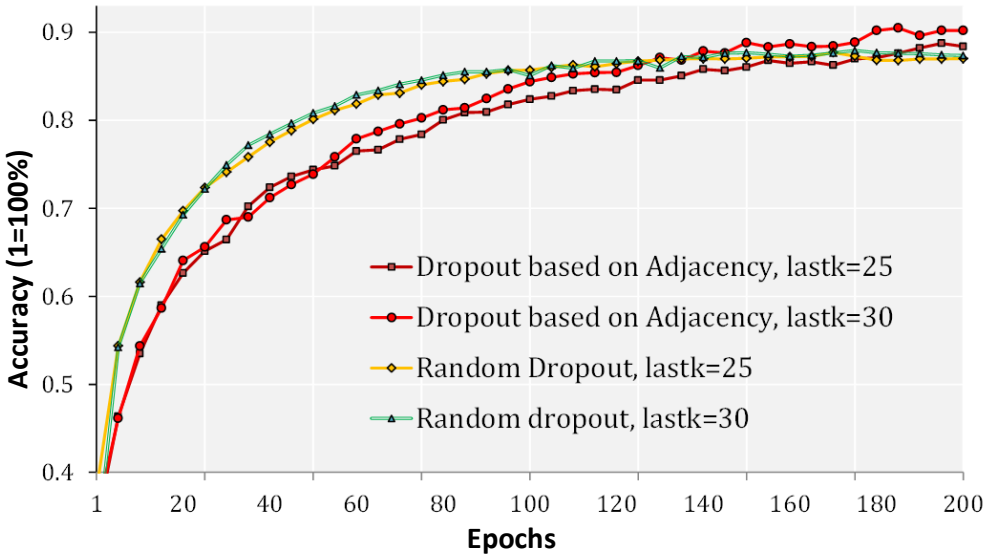



Figure 8.10: Comparison between random dropout and adjacency based dropout

### 8.5.4 Layers

The most advantageous property of the proposed graph learning algorithm is able to have deeper layers that are able to abstract the features of the induced graph in each iteration. To demonstrate this capability, we have selected a kernel size of 2 for the filter weights, and measure the accuracy of the graph learning algorithm for layer size of 3, 4, 5, and 6. Furthermore, the hidden layer kernel size is set to two, leaky relu is chosen as the activation function, last layer kernel size is set to thirty, and random dropout is selected. It can be noticed from Figure 8.11 that layer size of 4 and 5 is able to achieve higher testing classification accuracy compared to shallow 3 layers and deeper 6 layer size. The highest accuracy achieved was ≈91% with four layers. The deep SGCNN layers are able to abstract the structural and attribute properties of the induced sub-graph in a hierarchical manner by using a smaller kernel size of 2. This means that in each layer smaller node size is fused and

the new combined features are learned. This property can be helpful in engineering design data, where there is some form of hierarchy in terms of design.

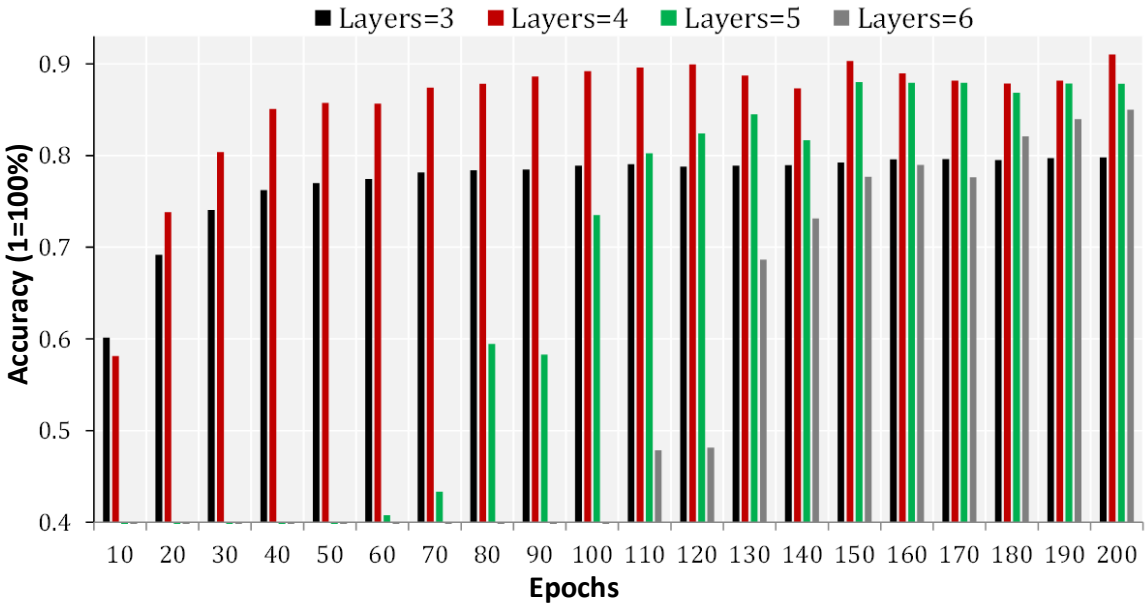



Figure 8.11: Performance comparison between various layer sizes

## 8.6 Discussion

The SGCNN's capability to learn sub-graph structure embedded with attributes was demonstrated in Section 9.6. It achieves very positive results on functional lifting using the GrabCAD dataset. We have presented the GrabCAD dataset as an engineering data set to show the applicability of design automation. However, in our future work, we will demonstrate the use of the SGCNN to electronic design dataset as well. Although SGCNN was created to address the functional lifting problem in engineering, we believe this is broadly applicable to other domains. In our future work, we will compare the performance of SGCNN against the latest work on graph convolutional networks targeting sub-graph-level embedding [216, 217]. Currently, we have shown that SGCNN is able to perform supervised learning in abstracting the design features and classifying them based on their functions. For robust design

automation, designers would be interested not only in classifying the designs but being able to reproduce or generate designs that slightly vary in functionality. In order to do this, the same structure of the SGCNN may be used for generative learning algorithms such as Variational Auto-Encoder (VAE) [218] or Generative Adversarial Networks (GAN) [121].

## 8.7 Summary

In this chapter, we present a novel structural graph convolutional neural network which can be used to abstract the non-euclidean graph or sub-graph dataset. These graphs can be used to represent the engineering design data and allow designers to effectively perform design automation by effectively clustering engineering designs with similar functionality. This allows designers to search for designs that are similar to the required functionality and aid in design automation.

# Chapter 9

# Part III: Dynamic Graph Embedding

## 9.1 Introduction

In chapter 8, we presented a structural graph convolutional neural network which is capable of performing supervising learning to estimate a function between non-Euclidean data and categorical data. In this chapter, we focus on non-Euclidean data which are evolving over time. In the cyber-physical system, most of the non-Euclidean data (such as engineering data, energy, and signal flow graph, call graph of the firmware, etc.) are always evolving. Hence, it is necessary to utilize algorithms that are capable of handling such temporally evolving non-Euclidean data. In this chapter, we present a novel dynamic graph embedding algorithm to handle this issue. In the rest of the chapter, we consider temporally evolving graphs as the non-Euclidean data and present an algorithm capable of capturing the pattern of time-varying links.

Understanding and analyzing graphs is an essential topic that has been widely studied over the past decades. Many real-world problems can be formulated as link predictions in graphs [219, 220, 221, 222]. For example, link prediction in an author collaboration

network [219] can be used to predict potential future author collaboration. Similarly, new connections between proteins can be discovered using protein interaction networks [223], and new friendships can be predicted using social networks [224]. Recent work on obtaining such predictions use graph representation learning. These methods represent each node in the network with a fixed dimensional embedding and map link prediction in the network space to the nearest neighbor search in the embedding space [225]. It has been shown that such techniques can outperform traditional link prediction methods on graphs [226, 227].

Existing works on graph representation learning primarily focus on static graphs of two types: (i) aggregated, consisting of all edges until time $T$; and (ii) snapshot, which comprise of edges at the current time step $t$. These models learn latent representations of the static graph and use them to predict missing links [228, 229, 230, 231, 226, 227, 232]. However, real networks often have complex dynamics which govern their evolution.

In this work, we aim to capture the underlying network dynamics of evolution. Given temporal snapshots of graphs, our goal is to learn a representation of nodes at each time step while capturing the dynamics such that we can predict their future connections. Learning such representations is a challenging task. Firstly, the temporal patterns may exist over varying period lengths. For example, in Figure **??**, user A may hold to each friend for a varying $k$ length. Secondly, different vertices may have different patterns. In Figure **??**, user A may break ties with friends whereas other users continue with their ties. Capturing such variations is extremely challenging. Existing research builds upon simplified assumptions to overcome these challenges. Methods including DynamicTriad [233], DynGEM [234] and TIMERS [235] assume that the patterns are of short duration (length 2) and only consider the previous time step graph to predict new links. Furthermore, DynGEM and TIMERS make the assumption that the changes are smooth and use regularization to disallow rapid changes.

### 9.1.1 Research Challenges

In this chapter, we present a model which overcomes the above challenges. *dyngraph2vec* uses multiple non-linear layers to learn structural patterns in each network. Furthermore, it uses recurrent layers to learn the temporal transitions in the network. The look back parameter in the recurrent layers controls the length of temporal patterns learned. We focus our experiments on the task of link prediction. We compare dyngraph2vec with the state-of-the-art algorithms for dynamic graph embedding and show its performance on several real-world networks including collaboration networks and social networks. Our experiments show that using a deep model with recurrent layers can capture temporal dynamics of the networks and significantly outperform the state-of-the-art methods on link prediction. We emphasize that the work in this chapter is targeted towards link prediction and not node classification.

### 9.1.2 Our Contribution

Overall, the contributions made by the work presented in this chapter are as follows:

1. We propose *dyngraph2vec*, a dynamic graph embedding model which captures temporal dynamics.

2. We demonstrate that capturing network dynamics can significantly improve the performance on link prediction.

3. We present variations of our model to show the key advantages and differences.

4. We publish a library, DynamicGEM [1], implementing the variations of our model and state-of-the-art dynamic embedding approaches.

[1] https://github.com/palash1992/DynamicGEM

## 9.2 Related Work

Graph representation learning techniques can be broadly divided into two categories: (i) static graph embedding, which represents each node in the graph with a single vector; and (ii) dynamic graph embedding, which considers multiple snapshots of a graph and obtains a time series of vectors for each node. Most analysis has been done on static graph embedding. Recently, however, some works have been devoted to studying dynamic graph embedding.

### 9.2.1 Static Graph Embedding

Methods to represent nodes of a graph typically aim to preserve certain properties of the original graph in the embedding space. Based on this observation, methods can be divided into (i) distance preserving, and (ii) structure preserving. Distance preserving methods devise objective functions such that the distance between nodes in the original graph and the embedding space have similar rankings. For example, Laplacian Eigenmaps [236] minimizes the sum of the distance between the embeddings of neighboring nodes under the constraints of translational invariance, thus keeping the nodes close in the embedding space. Similarly, Graph Factorization [228] approximates the edge weight with the dot product of the nodes' embeddings, thus preserving distance in the inner product space. Recent methods have gone further to preserve higher order distances. Higher Order Proximity Embedding (HOPE) [227] uses multiple higher-order functions to compute a similarity matrix from a graph's adjacency matrix and uses Singular Value Decomposition (SVD) to learn the representation. GraRep [230] considers the node transition matrix and its higher powers to construct a similarity matrix.

On the other hand, structure-preserving methods aim to preserve the roles of individual nodes in the graph. *node2vec* [226] uses a combination of breadth-first search and depth-first search

to find nodes similar to a node in terms of distance and role. Recently, deep learning methods to learn network representations have been proposed. These methods inherently preserve the higher order graph properties including distance and structure. SDNE [237], DNGR [238] and VGAE [239] use deep autoencoders for this purpose. Some other recent approaches use graph convolutional networks to learn inherent graph structure [198, 240, 196].

### 9.2.2 Dynamic Graph Embedding

Embedding dynamic graphs is an emerging topic still under investigation. Some methods have been proposed to extend static graph embedding approaches by adding regularization [241, 235]. DynGEM [242] uses the learned embedding from previous time step graphs to initialize the current time step embedding. Although it does not explicitly use regularization, such initialization implicitly keeps the new embedding close to the previous. DynamicTriad [233] relaxes the temporal smoothness assumption but only considers patterns spanning two-time steps. TIMERS [235] incrementally updates the embedding using incremental Singular Value Decomposition (SVD) and reruns SVD when the error increases above a threshold.

DYLINK2VEC [243] learns embedding of links (node pairs instead of nodes) and uses temporal functions to learn patterns over time.

Link embedding renders the method non-scalable for graphs with high density. Our model uses recurrent layers to learn temporal patterns over long sequences of graphs and multiple fully connected layer to capture intricate patterns at each time step.

### 9.2.3 Dynamic Link Prediction

Several methods have been proposed on dynamic link prediction without emphasis on graph embedding. Many of these methods use probabilistic non-parametric approaches [244, 245]. NonParam [244] uses kernel functions to model the dynamics of individual node features influenced by the neighbor features. Another class of algorithms uses matrix and tensor factorization to model link dynamics [246, 247]. Further, many dynamic link prediction models have been proposed for specific applications including recommendation systems [248] and attributed graphs [249]. These methods often have assumptions about the inherent structure of the network and require node attributes. Our model, however, extends the traditional graph embedding framework to capture network dynamics.

## 9.3 Motivating Example

We consider a toy example to motivate the idea of capturing network dynamics. Consider an evolution of graph $G$, $\mathcal{G} = \{G_1, .., G_T\}$, where $G_t$ represents the state of graph at time $t$. The initial graph $G_1$ is generated using the Stochastic Block Model [250] with 2 communities (represented by colors indigo and yellow in Figure 9.1), each with 500 nodes. To properly demonstrate the changes in the community, we have only shown a total of 50 nodes (25 from each of the community) and shown only two migrating nodes in Figure 9.1. The in-block and cross-block probabilities are set to 0.1 and 0.01 respectively. The evolution pattern can be defined as a three-step process. In the first step (shown in Figure 9.1(a)), we randomly and uniformly select 10 nodes (colored red in Figure 9.1) from the yellow community. In step two (shown in Figure 9.1(b)), we randomly add 30 edges between each of the selected nodes in step one and random nodes in the Indigo community. This is similar to having more than cross-block probability but less than in-block probability. In step three (shown in Figure 9.1(c)), the community membership of the nodes selected in step 2 is changed from

yellow to indigo. Similarly, the edges (colored red in Figure 9.1) are either removed or added to reflect the cross-block and in-block connection probabilities. Then, for the next time step (shown in Figure 9.1(d)), the same three steps are repeated to evolve the graph. Informally, this can be interpreted as a two-step movement of users from one community to another by initially increasing friends in the other community and subsequently moving to it.

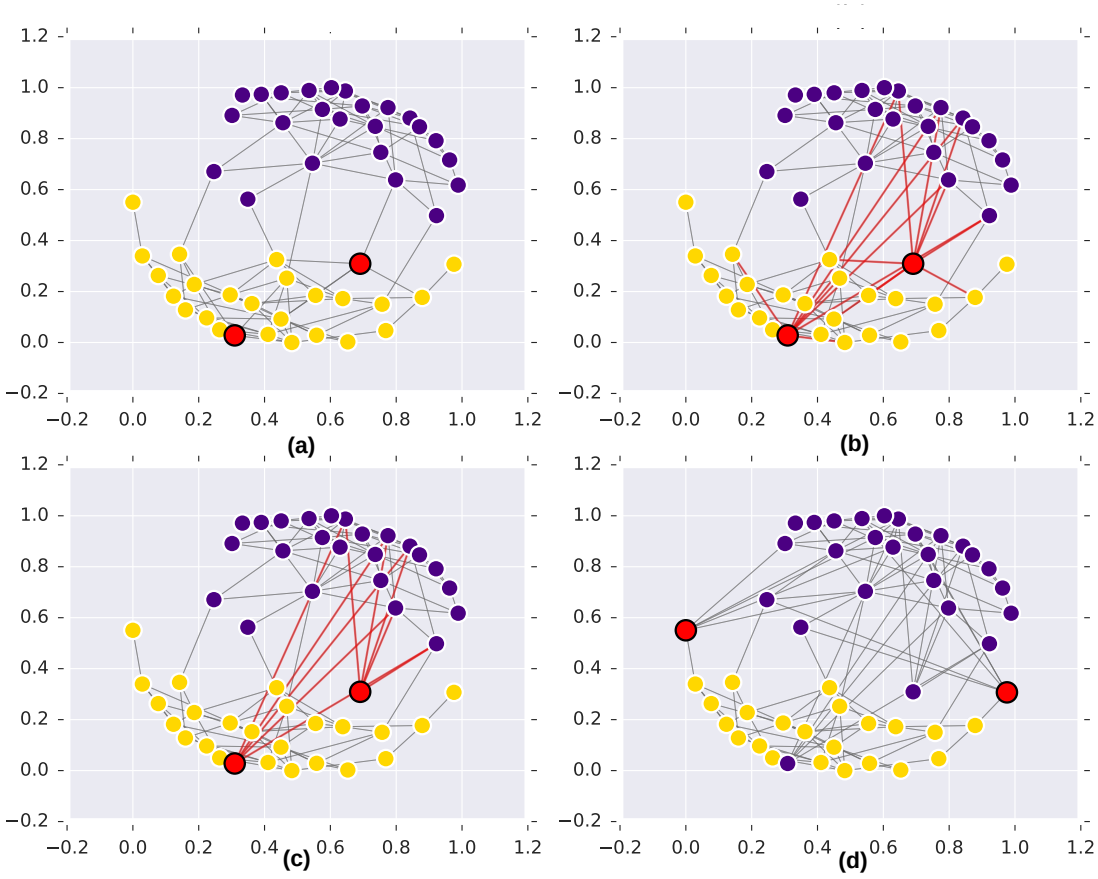



Figure 9.1: Motivating example of network evolution - community shift

Our task is to learn the embeddings predictive of the change in the community of the 10 nodes. Figure 9.2 shows the results of the state-of-the-art dynamic graph embedding techniques (*DynGEM*, *optimalSVD*, and *DynamicTriad*) and the three variations of our model: *dyngraph2vecAE*, *dyngraph2vecRNN* and *dyngraph2vecAERNN* (see Methodology Section for the description of the methods). Figure 9.2 shows the embeddings of nodes after the first step of evolution. The nodes selected for community shift are colored in red. We

show the results for 4 runs of the model to ensure robustness. Figure 9.2(a) shows that DynGEM brings the red nodes closer to the edge of the yellow community but does not move any of the nodes to the other community.

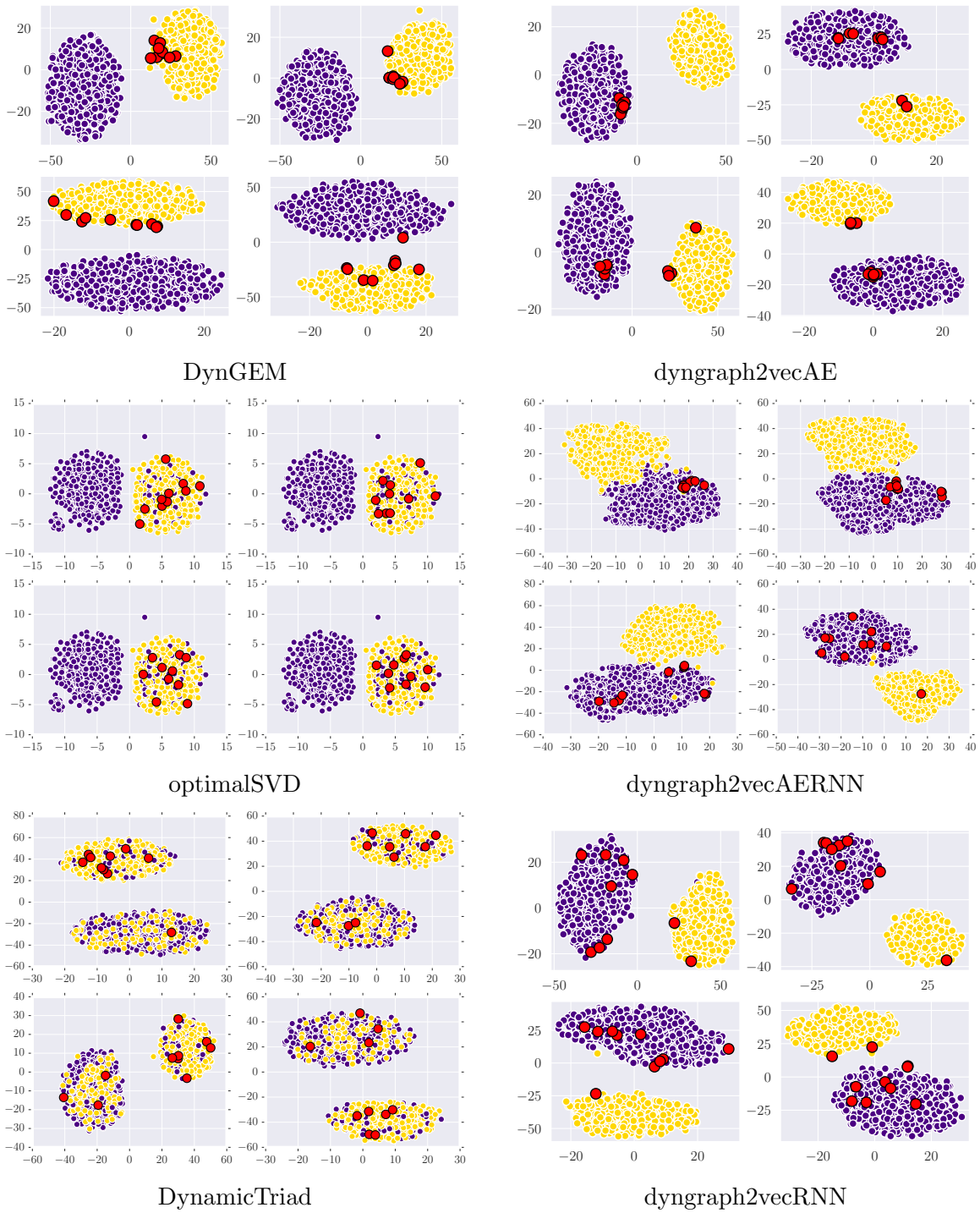



Figure 9.2: Motivating example of network evolution - community shift

Similarly, DynamicTriad results in Figure 9.2(c) show that it only shifts 1 to 4 nodes to its actual community in the next step. The optimalSVD method in Figure 9.2(b) is not able to shift any nodes. However, our *dyngraph2vecAE* and *dyngraph2vecRNN*, and *dyngraph2vecAERNN* (shown in Figure 9.2(d-f)) successfully capture the dynamics and move the embedding of most of the 10 selected nodes to the indigo community, keeping the rest of the nodes intact. This shows that capturing dynamics is critical in understanding the evolution of networks.

## 9.4 Methodology

In this section, we define the problem statement. We then explain multiple variations of deep learning models capable of capturing temporal patterns in dynamic graphs. Finally, we design the loss functions and optimization approach.

### 9.4.1 Problem Statement

Consider a weighted graph $G(V, E)$, with $V$ and $E$ as the set of vertices and edges respectively. We denote the adjacency matrix of $G$ by $A$, i.e. for an edge $(i, j) \in E$, $A_{ij}$ denotes its weight, else $A_{ij} = 0$. An evolution of graph $G$ is denoted as $\mathcal{G} = \{G_1, .., G_T\}$, where $G_t$ represents the state of graph at time $t$.

We define our problem as follows: *Given an evolution of graph $G$, $\mathcal{G}$, we aim to represent each node $v$ in a series of low-dimensional vector space $y_{v_1}, \dots y_{v_t}$, where $y_{v_t}$ is the embedding of node $v$ at time $t$, by learning mappings $f_t : \{V_1, \dots, V_t, E_1, \dots E_t\} \rightarrow \mathbb{R}^d$ and $y_{v_i} = f_i(v_1, \dots, v_i, E_1, \dots E_i)$ such that $y_{v_i}$ can capture temporal patterns required to predict $y_{v_{i+1}}$.* In other words, the embedding function at each time step uses information from graph evolution to capture network dynamics and can thus predict links with higher precision.

### 9.4.2 dyngraph2vec algorithm

**Algorithm 16:** dyngraph2vec

**Input:** Graphs $\mathcal{G} = \{G_1, .., G_T\}$, Dimension $d$, Look back $lb$
**Output:** Embedding Vector $Y$

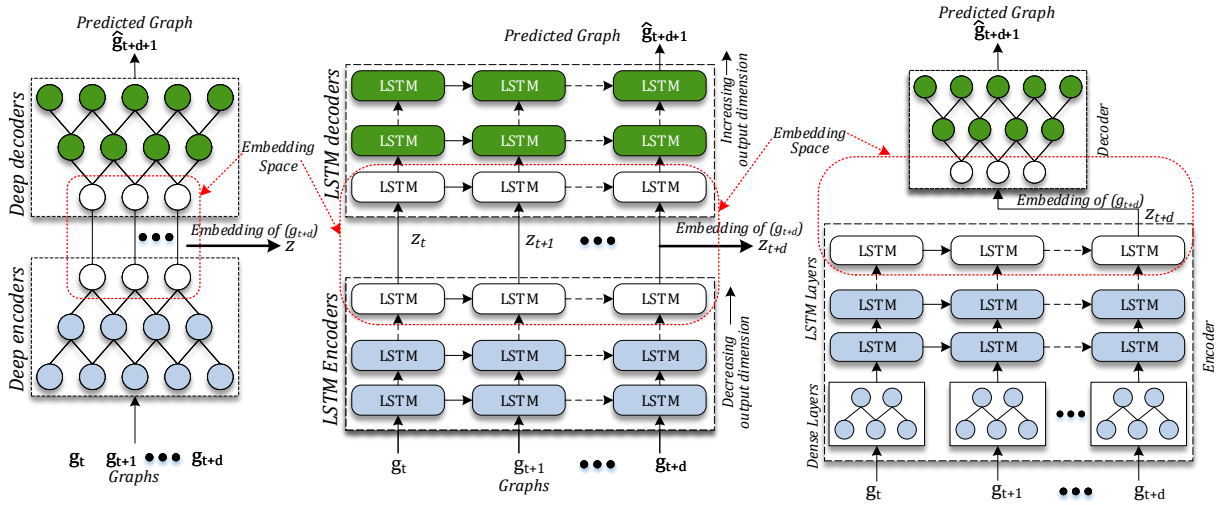
Generate adjacency matrices $\mathcal{A}$ from $\mathcal{G}$;
$\vartheta \leftarrow$ RandomInit();
Set $\mathcal{F} = \{(A_{t_u})\}$ for each $u \in V$, for each $t \in \{1..t\}$;
**for** $iter = 1 \dots I$ **do**
    $M \leftarrow$ getArchitectureInput($\mathcal{F}$, $lb$);
    Choose $L$ based on the architecture used;
    $grad \leftarrow \partial L / \partial \vartheta$;
    $\vartheta \leftarrow$ UpdateGradient($\vartheta$, $grad$);
**end**
$Y \leftarrow$ EncoderForwardPass($G$, $\vartheta$);
return $Y$

Our *dyngraph2vec* is a deep learning model that takes as input a set of previous graphs and generates as output the graph at the next time step, thus capturing highly non-linear interactions between vertices at each time step and across multiple time steps. Since the embedding values capture the temporal evolution of the links, it allows us to predict the next time step graph link. The model learns the network embedding at time step $t$ by optimizing the following loss function:

$$\begin{aligned} L_{t+l} &= \|(\hat{A}_{t+l+1} - A_{t+l+1}) \odot \mathcal{B}\|_F^2, \\ &= \|(f(A_t, \dots, A_{t+l}) - A_{t+l+1}) \odot \mathcal{B}\|_F^2. \end{aligned} \tag{9.1}$$

Here we penalize the incorrect reconstruction of edges at time $t+l+1$ by using the embedding at time step $t + l$. Minimizing this loss function enforces the parameters to be tuned such that it can capture evolving patterns relations between nodes to predict the edges at a future time step. The embedding at time step $t + d$ is a function of the graphs at time steps $t, t + 1, \dots, t + l$ where $l$ is the temporal look back. We use a weighting matrix $\mathcal{B}$ to weight the reconstruction of observed edges higher than unobserved links as traditionally used in the literature [237]. Here, $\mathcal{B}_{ij} = \beta$ for $(i, j) \in E_{t+l+1}$, else 1, where $\beta$ is a hyperparameter

controlling the weight of penalizing observed edges. Note that $\odot$ represents elementwise product.




(a) Dynamic Graph to Vector Auto Encoder (**dyngraph2vecAE**)

(b) Dynamic Graph to Vector Recurrent Neural Network (**dyngraph2vecRNN**)

(c) Dynamic Graph to Vector Autoenncoder Recurrent Neural Network (**dyngraph2vecAERNN**)

Figure 9.3: dyngraph2vec architecture variations for dynamic graph embedding

We propose three variations of our model based on the architecture of deep learning models as shown in Figure 9.3: (i) *dyngraph2vecAE*, (ii) *dyngraph2vecRNN*, and (iii) *dyngraph2vecAERNN*. Our three methods differ in the formulation of the function $f(.)$.

A simple way to extend the autoencoders traditionally used to embed static graphs [237] to temporal graphs is to add the information about previous $l$ graphs as input to the autoencoder. This model (*dyngraph2vecAE*) thus uses multiple fully connected layers to model the interconnection of nodes within and across time. Concretely, for a node $u$ with neighborhood vector set $u_{1..t} = [a_{u_t}, \ldots, a_{u_{t+l}}]$, the hidden representation of the first layer is learned as:

$$y_{u_t}^{(1)} = f_a(W_{AE}^{(1)} u_{1..t} + b^{(1)}), \tag{9.2}$$

where $f_a$ is the activation function, $W_{AE}^{(1)} \in \mathbb{R}^{d^{(1)} \times nl}$ and $d^{(1)}$, $n$ and $l$ are the dimensions of representation learned by the first layer, number of nodes in the graph, and look back,

respectively. The representation of the $k^{th}$ layer is defined as:

$$y_{u_t}^{(k)} = f_a(W_{AE}^{(k)} y_{u_t}^{(k-1)} + b^{(k)}). \tag{9.3}$$

Note that *dyngraph2vecAE* has $O(nld^{(1)})$ parameters. As most real-world graphs are sparse, learning the parameters can be challenging.

To reduce the number of model parameters and achieve a more efficient temporal learning, we propose *dyngraph2vecRNN* and *dyngraph2vecAERNN*. In *dyngraph2vecRNN* we use sparsely connected Long Short Term Memory (LSTM) networks to learn the embedding. LSTM is a type of Recurrent Neural Network (RNN) capable of handling long-term dependency problems. In dynamic graphs, there can be long-term dependencies which may not be captured by fully connected auto-encoders. The hidden state representation of a single LSTM network is defined as:

$$y_{u_t}^{(1)} = o_{u_t}^{(1)} * \tanh(C_{u_t}^{(1)}) \tag{9.4a}$$

$$o_{u_t}^{(1)} = \sigma_{u_t}(W_{RNN}^{(1)}[y_{u_{t-1}}^{(1)}, u_{1..t}] + b_o^{(1)}) \tag{9.4b}$$

$$C_{u_t}^{(1)} = f_{u_t}^{(1)} * C_{u_{t-1}}^{(1)} + i_{u_t}^{(1)} * \tilde{C}_{u_t}^{(1)} \tag{9.4c}$$

$$\tilde{C}_{u_t}^{(1)} = \tanh(W_C^{(1)}.[y_{u_{t-1}}^{(1)}, u_{1..t} + b_c^{(1)}]) \tag{9.4d}$$

$$i_{u_t}^{(1)} = \sigma(W_i^{(1)}.[y_{u_{t-1}}^{(1)}, u_{1..t}] + b_i^{(1)}) \tag{9.4e}$$

$$f_{u_t}^{(1)} = \sigma(W_f^{(1)}.[y_{u_{t-1}}^{(1)}, u_{1..t} + b_f^{(1)}]) \tag{9.4f}$$

where $C_{u_t}$ represents the cell states of LSTM, $f_{u_t}$ is the value to trigger the forget gate, $o_{u_t}$ is the value to trigger the output gate, $i_{u_t}$ represents the value to trigger the update gate of the LSTM, $\tilde{C}_{u_t}$ represents the new estimated candidate state, and $b$ represents the biases. There can be $l$ LSTM networks connected in the first layer, where the cell states and hidden representation are passed in a chain from $t-l$ to $t$ LSTM networks. The representation of

the $k^{th}$ layer is then given as follows:

$$y_{u_t}^{(k)} = o_{u_t}^{(k)} * \tanh(C_{u_t}^{(k)}) \tag{9.5a}$$

$$o_{u_t}^{(k)} = \sigma_{u_t}(W_{RNN}^{(k)}[y_{u_{t-1}}^{(k)}, y_{u_t}^{(k-1)}] + b_o^{(k)}) \tag{9.5b}$$

The problem with passing the sparse neighbourhood vector $u_{1..t} = [a_{u_t}, \ldots, a_{u_{t+l}}]$ of node $u$ to the LSTM network is that the LSTM model parameters (such as the number of memory cells, number of input units, output units, etc.) needed to learn a low dimension representation become large. Rather, the LSTM network may be able to better learn the temporal representation if the sparse neighbourhood vector is reduced to a low dimension representation. To achieve this, we propose a variation of *dyngraph2vec* model called *dyngraph2vecAERNN*. In *dyngraph2vecAERNN* instead of passing the sparse neighbourhood vector, we use a fully connected encoder to initially acquire low dimensional hidden representation given as follows:

$$y_{u_t}^{(p)} = f_a(W_{AERNN}^{(p)} y_{u_t}^{(p-1)} + b^{(p)}). \tag{9.6}$$

where $p$ represents the output layer of the fully connected encoder. This representation is then passed to the LSTM networks.

$$y_{u_t}^{(p+1)} = o_{u_t}^{(p+1)} * \tanh(C_{u_t}^{(p+1)}) \tag{9.7a}$$

$$o_{u_t}^{(p+1)} = \sigma_{u_t}(W_{AERNN}^{(p+1)}[y_{u_{t-1}}^{(p+1)}, y_{u_t}^{(p)}] + b_o^{(p+1)}) \tag{9.7b}$$

Then the hidden representation generated by the LSTM network is passed to a fully connected decoder.

### 9.4.3 Optimization

We optimize the loss function defined above to get the optimal model parameters. By applying the gradient with respect to the decoder weights on equation 9.1, we get:

$$\frac{\partial L_t}{\partial W_*^{(K)}} = [2(\hat{A}_{t+1} - A_{t+1}) \odot \mathcal{B}][\frac{\partial f_a(Y^{(K-1)}W_*^{(K)} + b^{(K)})}{\partial W_*^{(K)}}],$$

where $W_*^{(K)}$ is the weight matrix of the penultimate layer for all the three models. For each individual model, we back propagate the gradients based on the neural units to get the derivatives for all previous layers. For the LSTM based *dyngraph2vec* models, back propagation through time is performed to update the weights of the LSTM networks.

After obtaining the derivatives, we optimize the model using stochastic gradient descent (SGD) [251] with Adaptive Moment Estimation (Adam)[215]. The algorithm is specified in Algorithm 16.

## 9.5 Experiments

In this section, we describe the data sets used and establish the baselines for comparison. Furthermore, we define the evaluation metrics for our experiments and parameter settings. All the experiments were performed on a 64 bit Ubuntu 16.04.1 LTS system with Intel (R) Core (TM) i9-7900X CPU with 19 processors, 10 CPU cores, 3.30 GHz CPU clock frequency, 64 GB RAM, and two Nvidia Titan X, each with 12 GB memory.

Table 9.1: Dataset statistics

| Name | SBM | Hep-th | AS |
|---|---|---|---|
| Nodes $n$ | 1000 | 150-14446 | 7716 |
| Edges $m$ | 56016 | 268-48274 | 487-26467 |
| Time steps $T$ | 10 | 136 | 733 |

### 9.5.1 Datasets

We conduct experiments on two real-world datasets and a synthetic dataset to evaluate our proposed algorithm. We assume that the proposed models are aware of all the nodes, and that no new nodes are introduced in subsequent time steps. Rather, the links between the existing nodes change with a certain temporal pattern. The datasets are summarized in Table 9.1.

**Stochastic Block Model (SBM) - community diminishing**: In order to test the performance of various static and dynamic graph embedding algorithms, we generated synthetic SBM data with two communities and a total of 1000 nodes. The cross-block connectivity probability is 0.01 and in-block connectivity probability is set to 0.1. One of the communities is continuously diminished by migrating the 10-20 nodes to the other community. A total of 10 dynamic graphs are generated for the evaluation. Since SBM is a synthetic dataset, there is no notion of time steps.

**Hep-th** [219]: The first real-world data set used to test the dynamic graph embedding algorithms is the collaboration graph of authors in High Energy Physics Theory conference. The original data set contains abstracts of papers in High Energy Physics Theory conference in the period from January 1993 to April 2003. Hence, the resolution of the time step is one month. For our evaluation, we consider the last 50 snapshots of this dataset. From this dataset 2000 nodes are sampled for training and testing the proposed models.

**Autonomous Systems (AS)** [252]: The second real-world dataset utilized is a communication network of who-talks-to-whom from the BGP (Border Gateway Protocol) logs. The dataset contains 733 instances spanning from November 8, 1997, to January 2, 2000. Hence, the resolution of the time step for AS dataset is one month as well. For our evaluation, we consider a subset of this dataset which contains the last 50 snapshots. From this dataset 2000 nodes are sampled for training and testing the proposed models.

### 9.5.2 Baselines

We compare our model with the following state-of-the-art static and dynamic graph embedding methods:

- *Optimal Singular Value Decomposition* (**OptimalSVD**) [253]: It uses the singular value decomposition of the adjacency matrix or its variation (i.e., the transition matrix) to represent the individual nodes in the graph. The low rank SVD decomposition with largest $d$ singular values are then used for graph structure matching, clustering, etc.

- *Incremental Singular Value Decomposition* (**IncSVD**) [254]: It utilizes a perturbation matrix which captures the changing dynamics of the graphs and performs additive modification on the SVD.

- *Rerun Singular Value Decomposition* (**RerunSVD** or TIMERS) [235]: It utilizes the incremental SVD to get the dynamic graph embedding, however, it also uses a tolerance threshold to restart the optimal SVD calculation when the incremental graph embedding starts to deviate.

- *Dynamic Embedding using Dynamic Triad Closure Process* (**dynamicTriad**) [233]: It utilizes the triadic closure process to generate a graph embedding that preserves structural and evolution patterns of the graph.

- *Deep Embedding Method for Dynamic Graphs* (**dynGEM**) [234]: It utilizes deep auto-encoders to incrementally generate embedding of a dynamic graph at snapshot $t$ by using only the snapshot at time $t-1$.

### 9.5.3 Evaluation Metrics

In our experiments, we evaluate our model on link prediction at time step $t+1$ by using all graphs until the time step $t$ . We use Mean Average Precision (MAP) as our metrics. $precision@k$ is the fraction of correct predictions in the top $k$ predictions. It is defined as $P@k = \frac{|E_{pred}(k) \cap E_{gt}|}{k}$, where $E_{pred}$ and $E_{gt}$ are the predicted and ground truth edges respectively. MAP averages the precision over all nodes. It can be written as $\frac{\sum_i AP(i)}{|V|}$ where $AP(i) = \frac{\sum_k precision@k(i) \cdot \mathbb{I}\{E_{pred_i}(k) \in E_{gt_i}\}}{|\{k : E_{pred_i}(k) \in E_{gt_i}\}|}$ and $precision@k(i) = \frac{|E_{pred_i}(1{:}k) \cap E_{gt_i}|}{k}$. $P@k$ values are used to test the top predictions made by the model. MAP values are more robust and average the predictions for all nodes. High MAP values imply that the model can make good predictions for most nodes.

## 9.6 Results and Analysis

In this section, we present the performance result of various models for link prediction on different datasets. We train the model on graphs from time step $t$ to $t+l$ where $l$ is the lookback of the model, and predict the links of the graph at time step $t+l+1$. The lookback $l$ is a model hyperparameter. For an evolving graph with $T$ steps, we perform the above prediction from $T/2$ to $T$ and report the average MAP of link prediction. Furthermore, we also present the performance of models when an increasing length of the graph sequence are provided in the training data. Unless explicitly mentioned, for the models consisting of a recurrent neural network, a lookback value of 3 is used for the training and testing purpose.

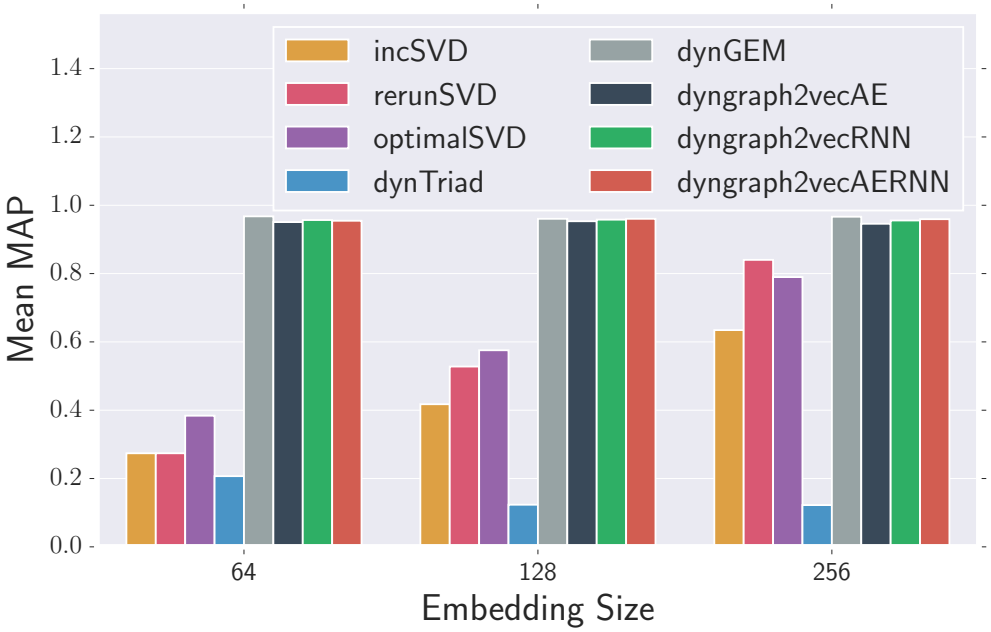



Figure 9.4: MAP values for the SBM dataset

### 9.6.1 SBM Dataset

The MAP values for various algorithms with SBM dataset with a diminishing community is shown in Figure 9.4. The MAP values shown are for link prediction with embedding sizes *64*, *128* and *256*. This figure shows that our methods *dyngraph2vecAE*, *dyngraph2vecRNN* and *dyngraph2vecAERNN* all have higher MAP values compared to the rest of the base-lines except for *dynGEM*. The *dynGEM* algorithm is able to have higher MAP values than all the algorithms. This is due to the fact that *dynGEM* also generates the embedding of the graph at snapshot $t+1$ using the graph at snapshot $t$. Since in our SBM dataset the node-migration criteria are introduced only a one-time step earlier, the *dynGEM* node embedding technique is able to capture these dynamics. However, the proposed *dyngraph2vec* methods also achieve average MAP values within $\pm 1.5\%$ of the MAP values achieved by *dynGEM*. Notice that the MAP values of SVD based methods increase as the embedding size increases. However, this is not the case for *dynTriad*.

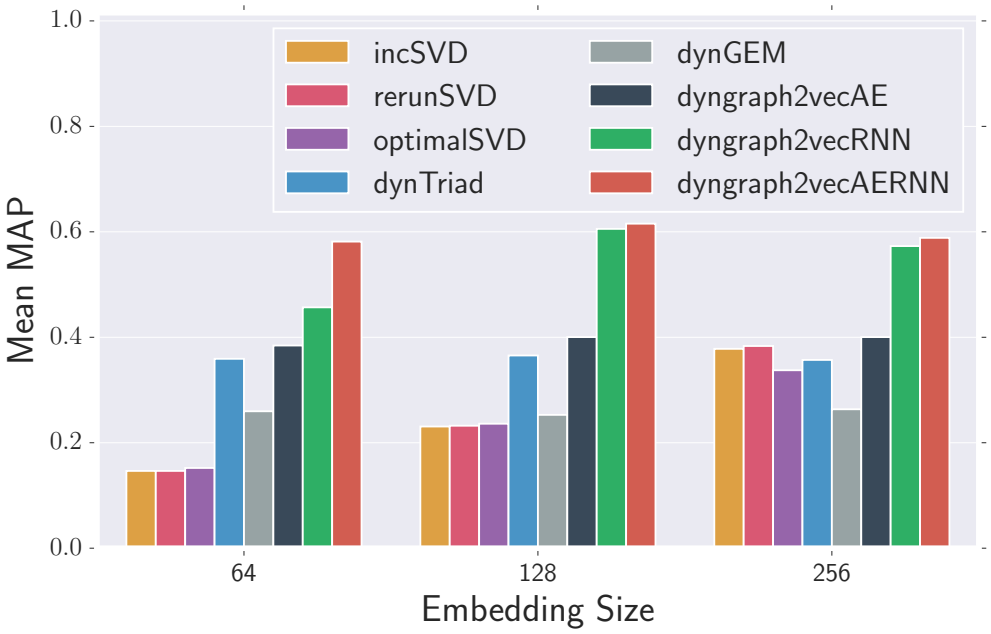



Figure 9.5: MAP values for the Hep-th dataset

### 9.6.2 Hep-th Dataset

The link prediction results for the Hep-th dataset is shown in Figure 9.5. The proposed *dyngraph2vec* algorithms outperform all the other state-of-the-art static and dynamic algorithms. Among the proposed algorithms, *dyngraph2vecAERNN* has the highest MAP values, followed by *dyngraph2vecRNN* and *dyngraph2vecAE*, respectively. The *dynamicTriad* is able to perform better than the SVD based algorithms. Notice that *dynGEM* is not able to have higher MAP values than the *dyngraph2vec* algorithms in the Hep-th dataset.Since dyngraph2vec utilizes not only $t-1$ but $t-l-1$ time-steps to predict the link for the time-step $t$, it has higher performance compared to other state-of-the-art algorithms.

### 9.6.3 AS Dataset

The MAP value for link prediction with various algorithms for the AS dataset is shown in Figure 9.6. *dyngraph2vecAERNN* outperforms all the state-of-the-art algorithms. The algorithm with the second highest MAP score is *dyngraph2vecRNN*. However, *dyngraph2vecAE*

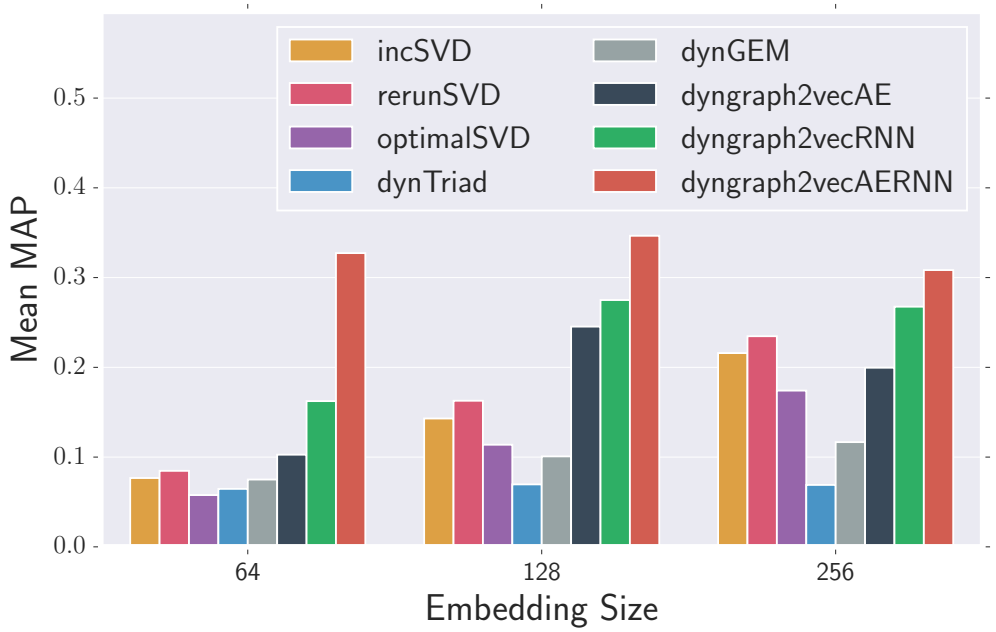



Figure 9.6: MAP values for the AS dataset

has a higher MAP only with a lower embedding of size 64. SVD methods are able to improve their MAP values by increasing the embedding size. However, they are not able to outperform the *dyngraph2vec* algorithms.

### 9.6.4 MAP exploration

The summary of MAP values for different embedding sizes (64, 128 and 256) for different datasets is presented in Table 9.2. The top three highest MAP values are highlighted in bold. For the synthetic SBM dataset, the top three algorithms with highest MAP values are *dynGEM*, *dyngraph2VecAERNN*, and *dyngraph2vecRNN*, respectively. Since the changing pattern for the SBM dataset is introduced only at timestep $t-1$, *dynGEM* is able to better predict the links. The model architecture of *dynGEM* and *dyngraph2vecAE* are only different on what data are fed to train the model. In *dyngraph2vecAE*, we essentially feed more data depending on the size of the lookback. The lookback size increases the model complexity. Since the SBM dataset doesn't have temporal patterns evolving for more than one-time steps, the dyngraph2vec models are only able to achieve comparable but not better result

compared to *dynGEM*.

Table 9.2: Average MAP values over different embedding sizes

| | Average MAP | | |
|---|---|---|---|
| Method | SBM | Hep-th | AS |
| IncrementalSVD | 0.4421 | 0.2518 | 0.1452 |
| rerunSVD | 0.5474 | 0.2541 | 0.1607 |
| optimalSVD | 0.5831 | 0.2419 | 0.1152 |
| dynamicTriad | 0.1509 | 0.3606 | 0.0677 |
| dynGEM | **0.9648** | 0.2587 | 0.0975 |
| **dyngraph2vec**AE (lb=3) | 0.9500 | 0.3951 | 0.1825 |
| **dyngraph2vec**AE (lb=5) | - | **0.512** | **0.2800** |
| **dyngraph2vec**RNN (lb=3) | **0.9567** | 0.5451 | 0.2350 |
| **dyngraph2vec**RNN | - | **0.7290** (lb=8) | **0.313** (lb=10) |
| **dyngraph2vec**AERNN (lb=3) | **0.9581** | 0.5952 | 0.3274 |
| **dyngraph2vec**AERNN | - | **0.739 (lb=8)** | **0.3801 (lb=10)** |

lb = Lookback value

For the Hep-th dataset, the top three algorithm with highest MAP values are *dyngraph2VecAERNN*, *dyngraph2VecRNN*, and *dyngraph2VecAE*, respectively. In fact, compared to the state-of-the-art algorithm *dynamicTriad*, the proposed models *dyngraph2VecAERNN*(with lookback=8), *dyngraph2VecRNN* (with lookback=8)), and *dyngraph2VecAE*(with lookback=5) obtain ≈105%, ≈102%, and ≈42% higher average MAP values, respectively.

For the AS dataset, the top three algorithm with highest MAP values are *dyngraph2VecAERNN*, *dyngraph2VecRNN*, and *dyngraph2VecAE*, respectively. Compared to the state-of-the-art *rerunSVD* algorithm, the proposed models *dyngraph2VecAERNN*(with lookback=10), *dyngraph2VecRNN* (with lookback=10), and *dyngraph2VecAE* (with lookback=5) obtain ≈137%, ≈95%, and ≈74% higher average MAP values, respectively.

These results show that the dyngraph2vec variants are able to capture the graph dynamics much better than most of the state-of-the-art algorithms in general.

### 9.6.5 Hyper-parameter Sensitivity: Lookback

One of the important parameters for time-series analysis is how much in the past the method looks to predict the future. To analyze the effect of look back on the MAP score we have trained the *dyngraph2Vec* algorithms with various look back values. The embedding dimension is fixed to 128. The look back size is varied from 1 to 10. We then tested the change in MAP values with the real word datasets AS and Hep-th.


Figure 9.7: Mean MAP values for various lookback numbers for Hep-th dataset

Performance of *dyngraph2Vec* algorithms with various lookback values for the Hep-th dataset is presented in Figure 9.7. It can be noticed that increasing lookback values consistently increase the average MAP values. Moreover, it is interesting to notice that *dyngraph2VecAE* although has increased in performance until lookback size of 8, its performance is decreased for lookback value of 10. Since it does not have memory units to store the temporal patterns like the recurrent variations, it relies solely on the fully connected dense layers to encode to the pattern. This seems rather ineffective compared to the *dyngraph2VecRNN* and *dyngraph2vecAERNN* for the Hep-th dataset. The highest MAP values achieved if by *dyngraph2vecAERNN* is 0.739 for the lookback size of 8.

Similarly, the performance of the proposed models while changing the lookback size for AS

dataset is presented in Figure 9.8. The average MAP values also increase with the increasing lookback size in the AS dataset. The highest MAP value of 0.3801 is again achieved by *dyngraph2vecAERNN* with the lookback size of 10. The *dyngraph2vecAE* model, initially, has comparable and sometimes even higher MAP value with respect to *dyngraph2vecRNN*. However, it can be noticed that for the lookback size of 10, the *dyngraph2vecRNN* outperforms *dyngraph2vecAE* model consisting of just the fully connected neural networks. In fact, the MAP value does not increase after the lookback size of 5 for *dyngraph2vecAE*.

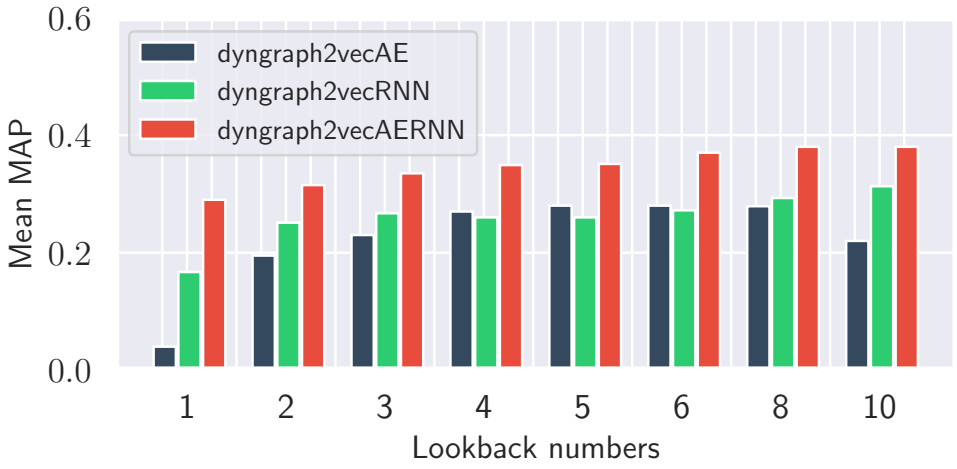



Figure 9.8: Mean MAP values for various lookback numbers for AS dataset

### 9.6.6 Length of training sequence versus MAP value

In this section, we present the impact of length of graph sequence supplied to the models during training on its performance. In order to conduct this experiment, the graph sequence provided as training data is increased one step at a time. Hence, we use graph sequence of length 1 to $t \in [T, T+1, T+2, T+3, \ldots, T+n]$ to predict the links for graph at time step $t \in [T+1, T+2, \ldots, T+n+1]$, where $T \geq lookback$. The experiment is performed on Hep-th and AS dataset with a fixed lookback size of 8. The total sequence of data is 50 and it is split between 25 for training and 25 for testing. Hence, in the experiment, the training data sequence increases from a total of 25 sequences to 49 graph sequence. The results in

Figure 9.9 and 9.10 shows the average MAP values for predicting the links starting the graph sequence at $26^{th}$ to all the way to $50^{th}$ time-step. Where each time step represents a month.

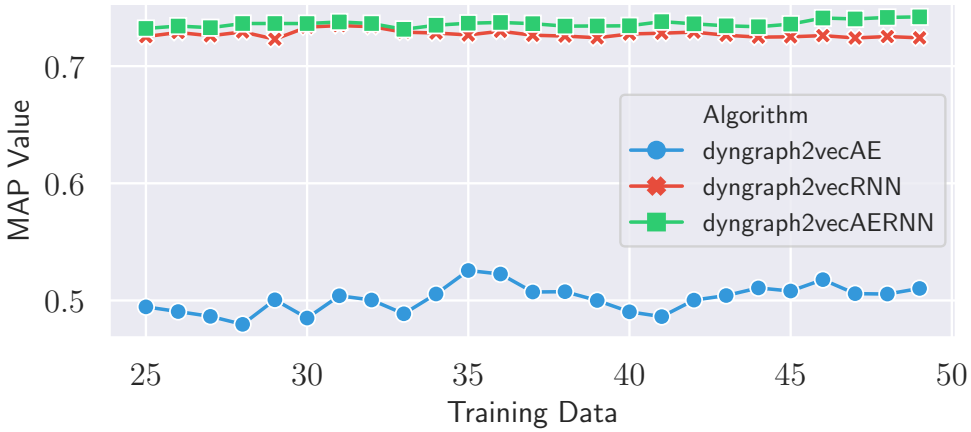



Figure 9.9: MAP value with increasing amount of temporal graphs added in the training data for Hep-th dataset (lookback = 8).

The result of increasing the amount of graph sequence in training data for Hep-th dataset is shown in Figure 9.9. It can be noticed that for both the *RNN* and *AERNN* the increasing amount of graph sequence in the data does not drastically increase the MAP value. For, *dyngraph2vecAE* there is a slight increase in the MAP value towards the end.

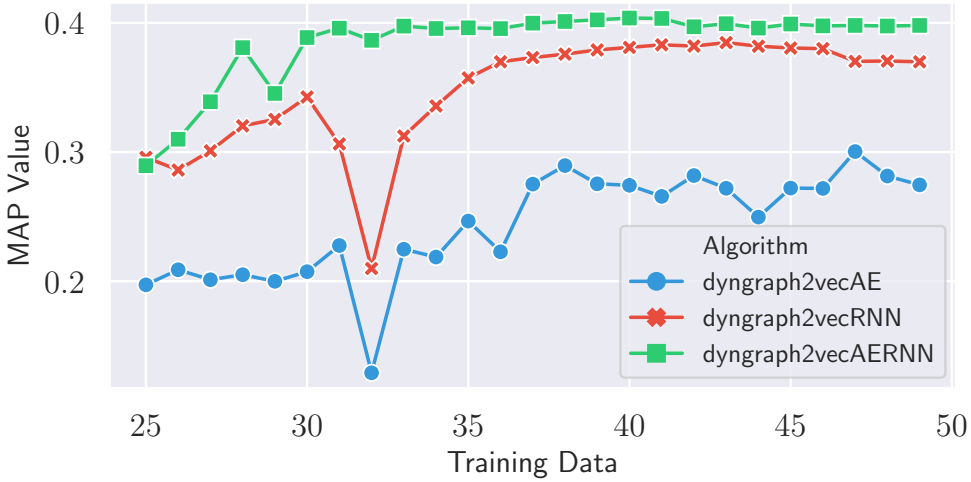



Figure 9.10: MAP value with increasing amount of temporal graphs added in the training data for AS dataset (lookback = 8)

On the other hand, increasing the amount of graph sequence for the AS dataset during

training gives a gradual improvement in link prediction performance in the testing phase. However, they start converging eventually after seeing 80% (total of 40 graph sequence) of the sequence data.

## 9.7 Discussion

**Model Variation:** It can be observed that among different proposed models, the recurrent variation was capable of achieving higher average MAP values. These architectures are efficient in learning short and long term temporal patterns and provide an edge in learning the temporal evolution of the graphs compared to the fully connected neural networks without recurrent units.

**Dataset:** We observe that depending on the dataset, the same model architecture provides different performance. Due to the nature of data, it may have different temporal patterns, periodic, semi-periodic, stationary, etc. Hence, to capture all these patterns, we found out that the models have to be tuned specifically to the dataset.

**Sampling:** One of the weakness of the proposed algorithms is that the model size (in terms of the number of weights to be trained) increases based on the size of the nodes considered during the training phase. To overcome this, the nodes have been sampled. Currently, we utilize uniform sampling of the nodes to mitigate this issue. However, we believe that a better sampling scheme that is aware of the graph properties may further improve its performance.

**Large Lookbacks:** While it is desirable to test large lookback values for learning the temporal evolution with the current hardware resources, we constantly ran into resource exhausted error with lookbacks greater than 10.

## 9.8 Summary

In this chapter, we introduced dyngraph2vec, a dynamic data-driven model for capturing temporal patterns in dynamic networks. It learns the evolution patterns of individual nodes and provides an embedding capable of predicting future links with higher precision. We propose three variations of our model based on the architecture with varying capabilities. The experiments show that our model can capture temporal patterns on synthetic and real datasets and outperform state-of-the-art methods in link prediction. There are several directions for future work: (1) interpretability by extending the model to provide more insight into network dynamics and better understand temporal dynamics; (2) automatic hyperparameter optimization for higher accuracy; and (3) graph convolutions to learn from node attributes and reduce the number of parameters. The unsupervised data-driven approach can be applied to abstract temporally evolving non-Euclidean data present in a cyber-physical system.

# Chapter 10

# Conclusion

Data-driven modeling is a powerful tool to aid designers, system architects, design automation tools, etc., to name few, for quickly estimating the desired relationship between the input and the output data. In a cyber-physical system, it can play a major role due to large volume, variety, and velocity of data being produced every day by these systems. As presented in this thesis, it can be used to discover new attack models to breach the confidentiality of the cyber-physical systems. It can be used to provide defensive mechanisms such as leakage aware computer-aided manufacturing tools or kinetic cyber attack detection tools. Moreover, it can also be used to model the relationship between the various information and energy flows of the cyber-physical systems. Furthermore, when novel algorithms that are able to process non-Euclidean data are developed, data-driven modeling can help us in estimating the relationship between complex graph data and categorical or nominal data. The aim of this thesis has been to explore the data-driven methodologies in conjunction with the side-channel analysis to aid in cyber-physical system modeling. However, it is not intended to replace the first principal-based rigorous modeling approaches which are proven and tested. The humble aim has been to provide supporting modeling methodologies.

# Bibliography


[1] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Design Automation Conference*, pages 731–736. IEEE, 2010.

[2] Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*. IEEE, 2008.

[3] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. A survey of cyber-physical systems. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[4] Imen Graja, Slim Kallel, Nawal Guermouche, Saoussen Cheikhrouhou, and Ahmed Hadj Kacem. A comprehensive survey on modeling of cyber-physical systems. *Concurrency and Computation: Practice and Experience*, page e4850.

[5] Remi R Lam, Lior Horesh, Haim Avron, and Karen E Willcox. Should you derive, or let the data drive? an optimization framework for hybrid first-principles data-driven modeling. *arXiv preprint arXiv:1711.04374*, 2017.

[6] Ricardo Bedin Franca, Jean-Paul Bodeveix, Mamoun Filali, Jean-Francois Rolland, David Chemouil, and Dave Thomas. The aadl behaviour annex–experiments and roadmap. In *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, pages 377–382. IEEE, 2007.

[7] Conrad Bock. Sysml and uml 2 support for activity modeling. *Systems Engineering*, 9(2):160–186, 2006.

[8] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. 1998.

[9] Janos Sztipanovits, Ted Bapty, Sandeep Neema, Larry Howard, and Ethan Jackson. Openmeta: A model-and component-based design tool chain for cyber-physical systems. In *From Programs to Systems. The Systems perspective in Computing*, pages 235–248. Springer, 2014.

[10] Nicholas Kottenstette, Gabor Karsai, and Janos Sztipanovits. The esmol language and tools for high-confidence distributed control systems design. part 1: Design language, modeling framework, and analysis. *ISIS*, 10:109.

[11] Joseph T Buck, Soonhoi Ha, Edward A Lee, and David G Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogeneous systems. 1994.

[12] Abhishek B Sharma, Franjo Ivančić, Alexandru Niculescu-Mizil, Haifeng Chen, and Guofei Jiang. Modeling and analytics for cyber-physical systems in the age of big data. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):74–77, 2014.

[13] Edward Lee. The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869, 2015.

[14] D Solomatine, Linda M See, and RJ Abrahart. Data-driven modelling: concepts, approaches and experiences. In *Practical hydroinformatics*, pages 17–30. Springer, 2009.

[15] François-Xavier Standaert et al. A unified framework for the analysis of side-channel key recovery attacks. In *Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009.

[16] Elisa Negri, Luca Fumagalli, et al. A Review of the Roles of Digital Twin in CPS-based Production Systems. *Procedia Manufacturing*, 2017.

[17] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems*. Springer, 2017.

[18] Ralf Schmitt, Stephen Rose, et al. Advance: Digital Enterprise - on the way to industrie 4.0. `https://goo.gl/LtL5oy`, 2015.

[19] General Electric Company. GE Digital Twin: Analytic engine for the digital power plant. `https://goo.gl/bhQzRF`, 2016.

[20] Edward Glaessgen and David Stargel. The digital twin paradigm for future NASA and US Air Force vehicles. In *Structures, Structural Dynamics and Materials Conference*, 2012.

[21] Edward M Kraft. The Air Force Digital thread/digital twin-life cycle integration and use of computational and experimental knowledge. In *54th AIAA Aerospace Sciences Meeting*, 2016.

[22] Sujit Rokka chhetri, A Canedo, and M Al Faruque. Confidentiality breach through acoustic side-channel in cyber-physical additive manufacturing systems. 2016.

[23] Al Faruque, Mohammad Abdullah, et al. Acoustic side-channel attacks on additive manufacturing systems. In *International Conference on Cyber-Physical Systems*. IEEE, 2016.

[24] Sujit Rokka Chhetri and Mohammad Abdullah Al Faruque. Side-channels of cyber-physical systems: Case study in additive manufacturing. *IEEE Design & Test*, 2017.

[25] Sina Faezi, Sujit Rokka Chhetri, et al. Oligo-snoop: A non-invasive side channel attack against dna synthesis machines. *Network and Distributed System Security Symposium (NDSS)*, 2019.

[26] **Sujit Rokka Chhetri**, Sina Faezi, Nafiul Rashid, and Mohammad Abdullah Al Faruque. Manufacturing supply chain and product lifecycle security in the era of industry 4.0 (**Accepted for publication**). In *Transactions on Dependable and Secure Computing*. IEEE, 2017.

[27] **Sujit Rokka Chhetri**, Sina Faezi, Nafiul Rashid, and Mohammad Abdullah Al Faruque. Manufacturing supply chain and product lifecycle security in the era of industry 4.0. ***Journal of Hardware and Systems Security***, 2(1):51–68, 2018.

[28] Sujit Rokka Chhetri, Jiang Wan, and Mohammad Abdullah Al Faruque. Cross-domain security of cyber-physical systems. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017.

[29] Mohammad Abdullah Al Faruque, **Sujit Rokka Chhetri**, A Canedo, and J Wan. Forensics of thermal side-channel in additive manufacturing systems. In *CECS Technical Report# 16-01*. University of California Irvine, 2016.

[30] Sujit Rokka Chhetri, Nafiul Rashid, Sina Faezi, and Mohammad Abdullah Al Faruque. Security trends and advances in manufacturing systems in the era of industry 4.0. 2017.

[31] **Sujit Rokka Chhetri**, Sina Faezi, and Mohammad Abdullah Al Faruque. Information leakage aware computer aided cyber-physical manufacturing (**Under Review**). In *Transactions on Information Forensics and Security*. IEEE, 2017.

[32] Sujit Rokka Chhetri, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Kcad: kinetic cyber-attack detection method for cyber-physical additive manufacturing systems. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 2016.

[33] **Sujit Rokka Chhetri**, Anthony Lopez, Jiang Wan, and Mohammad Abdullah Al Faruque. Gan-sec: **Generative Adversarial Network** modeling for the security analysis of cyber-physical production systems. In *accepted to be published in the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019.

[34] Sujit Rokka Chhetri, Sina Faezi, and Mohammad Abdullah Al Faruque. Fix the leak! an information leakage aware secured cyber-physical manufacturing system. In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017.

[35] **Sujit Rokka Chhetri**, Sina Faezi, and Mohammad Abdullah Al Faruque. Information leakage-aware computer-aided cyber-physical manufacturing. *IEEE Transactions on Information Forensics and Security*, 2018.

[36] Sujit Rokka Chhetri, S Faezi, and MA Al Faruque. Digital twin of manufacturing systems. *CECS Technical Report CECS TR-17-07*, 2017.

[37] **Sujit Rokka Chhetri**, Faezi Sina, Mohammad Abdullah Al Faruque, and Arquimedes Canedo. QUILT: Quality Inference from Living **Digital Twins** in IoT-Enabled Manufacturing Systems. *Accepted at International Conference on Internet-of-Things Design and Implementation Proceedings*, 2019.

[38] Sujit Rokka Chhetri, Jiang Wan, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Design automation using structural graph convolutional neural networks. In *Design Automation of Cyber-Physical Systems*, pages 237–259. Springer, 2019.

[39] *Palash Goyal, ***Sujit Rokka Chhetri**, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph **representation learning**. *arXiv preprint arXiv:1809.02657*, 2019. *Equal contribution.

[40] *Palash Goyal, ***Sujit Rokka Chhetri**, Ninareh Mehrabi, Emilio Ferrara, and Arquimedes Canedo. Dynamicgem: A library for **Dynamic Graph Embedding** methods. *arXiv preprint arXiv:1811.10734*, 2019. *Equal contribution.

[41] Jiang Wan, Blake S and **Sujit Rokka Chhetri** Pollard, et al. Future automation engineering using structural **Graph Convolutional Neural Networks**. *Computer-Aided Design (ICCAD), 2018 IEEE/ACM International Conference on*, 2018.

[42] Barbara Leukers et al. Hydroxyapatite scaffolds for bone tissue engineering made by 3D printing. *Journal of Materials Science: Materials in Medicine*, 16(12):1121–1124, 2005.

[43] ISS Platform and Feedstock Recycling. NASA advanced manufacturing technology. 2014.

[44] Billy Short. Quality metal additive manufacturing (QUALITY MADE) enabling capability, 2015. www.navy.mil.

[45] NDIA. Cyber security for advanced manufacturing. Technical report, National Defense Industrial Association, 2014.

[46] Cohn Reznick. Manufacturing: A persistent and prime cyber attack target. 2015. www.cohnreznick.com.

[47] Mark Yampolskiy, Todd R Andel, et al. Intellectual property protection in additive layer manufacturing: Requirements for secure outsourcing. In *Proceedings of the 4th Program Protection and Reverse Engineering Workshop*. ACM, 2014.

[48] Ian Gibson, David W Rosen, et al. *Additive manufacturing technologies*. Springer, 2010.

[49] Warwick Ashford. 21 percent of manufacturers hit by intellectual property theft. August 2014.

[50] David S WaII and Majid Yar. Intellectual property crime and the internet: cyber-piracy and stealing information intangibles. *Handbook of internet crime*, page 255, 2010.

[51] Tania Branigan. Google to end censorship in china over cyber attacks. *The Guardian*, pages 01–12, 2010.

[52] SketchUp Make, 2015. www.sketchup.com.

[53] Adobe Photoshop CC, 2015. www.adobe.com.

[54] Timothy R Holbrook and Lucas Osborn. Digital patent infringement in an era of 3D printing. *UC Davis Law Review, Forthcoming*, 2014.

[55] LD Sturm et al. Cyber-physical vulnerabilities in additive manufacturing systems. *Context*, 7:8, 2014.

[56] Michael Backes et al. Acoustic side-channel attacks on printers. In *USENIX Security Symposium*, pages 307–322, 2010.

[57] Ehsan Toreini, Brian Randell, and Feng Hao. An acoustic side channel attack on enigma. 2015.

[58] Abe Davis et al. The visual microphone: Passive recovery of sound from video. *ACM Trans. Graph*, 33(4):79, 2014.

[59] Hannah Vincent et al. Trojan detection and side-channel analyses for cyber-security in cyber-physical manufacturing systems. 2015.

[60] A Hughes and PJ Lawrenson. Electromagnetic damping in stepping motors. In *Proceedings of the Institution of Electrical Engineers*, volume 122, pages 819–824. IET, 1975.

[61] Takashi Kenjō and Akira Sugawara. *Stepping motors and their microprocessor controls*. Oxford University Press, USA, 1994.

[62] SJ Yang. *Low-noise electrical motors*, volume 13. Oxford University Press, USA, 1981.

[63] Bedřich Heller and Václav Hamata. *Harmonic field effects in induction machines*. Elsevier Science & Technology, 1977.

[64] Jacek F Gieras et al. *Noise of polyphase electric motors*. CRC press, 2005.

[65] László Timár-Peregrin Timár-P and PL Tímár. *Noise and vibration of electrical machines*, volume 34. North Holland, 1989.

[66] ECT So, RGD Williams, and SJ Yang. A simple model to calculate the stator radial vibration of a hybrid stepping motor. In *Industry Applications Society Annual Meeting, 1993., Conference Record of the 1993 IEEE*, pages 122–129. IEEE, 1993.

[67] Alan V Oppenheim et al. *Discrete-time signal processing*, volume 2. Prentice-hall Englewood Cliffs, 1989.

[68] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, et al. Acoustic side-channel attacks on additive manufacturing systems. In *International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016.

[69] Sergios Theodoridis et al. *Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*. Academic Press, 2010.

[70] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[71] Printrbot 3D Printers, 2015. www.printrbot.com.

[72] Zoom H6 Handy Recorder, 2015. www.zoom-na.com.

[73] MATLAB. *(R2015b)*. The MathWorks Inc., Natick, Massachusetts, 2015.

[74] Python 2.7.10, 2015. www.python.org.

[75] Dan Barry et al. Real-time sound source separation: Azimuth discrimination and resynthesis. In *Audio Engineering Society Convention 117*. Audio Engineering Society, 2004.

[76] Michael S Pedersen et al. A survey of convolutive blind source separation methods. *Multichannel Speech Processing Handbook*, pages 1065–1084, 2007.

[77] Avesta Hojjati, Anku Adhikari, et al. Leave your phone at the door: Side channels that reveal factory floor secrets. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[78] Sujit Rokka Chhetri et al. Confidentiality breach through acoustic side-channel in cyber-physical additive manufacturing systems. *ACM Transaction on Cyber-Physical Systems*, 2017.

[79] Jer straub. Identifying positioning-based attacks against 3d printed objects and the 3d printing process. In *SPIE Defense+ Security*. International Society for Optics and Photonics, 2017.

[80] Sofia Belikovetsky, Yosef Solewicz, et al. Detecting cyber-physical attacks in additive manufacturing using digital audio signing. *arXiv preprint arXiv:1705.06454*, 2017.

[81] Nektarios Georgios Tsoutsos, Homer Gamil, and Michail Maniatakos. Secure 3d printing: Reconstructing and validating solid geometries using toolpath reverse engineering. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM, 2017.

[82] David McMillen. Security trends in the manufacturing industry targeting control systems and crown jewels. IBM,https://securityintelligence.com/media/security-trends-in-the-manufacturing-industry/, 2016.

[83] Fei Chen, Gary Mac, and Nikhil Gupta. Security features embedded in computer aided design (cad) solid models for additive manufacturing. *Materials & Design*, 2017.

[84] Nikhil Gupta, Fei Chen, et al. Obfuscade: Obfuscating additive manufacturing cad models against counterfeiting. In *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

[85] Ron Jamieson and Herbert Hacker. Direct slicing of cad models for rapid prototyping. *Rapid Prototyping Journal*, 1995.

[86] GQ Jin, WD Li, et al. Adaptive tool-path generation of rapid prototyping for complex product models. *Journal of manufacturing systems*, 2011.

[87] Wei Yang, Yongbin Zhou, et al. Multi-channel fusion attacks. *IEEE Transactions on Information Forensics and Security*, 2017.

[88] Thomas Mativo, Colleen Fritz, and Ismail Fidan. Cyber acoustic analysis of additively manufactured objects. *The International Journal of Advanced Manufacturing Technology*, 2018.

[89] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: How, when and why?. In *CHES*, volume 5747, pages 429–443. Springer, 2009.

[90] Thomas R Kramer, Frederick M Proctor, et al. *The nist rs274ngc interpreter-version 3*, volume 5416. NISTIR, 2000.

[91] Ultimaker. Cura Software. https://ultimaker.com/en/products/cura-software, 2017.

[92] B Drumm. Printrbot: Your first 3d printer. *Retrieved November*, 26:2013, 2011.

[93] AT2021 Cardioid Condenser Microphone. Audio-Technica, 2016.

[94] Adafruit triple-axis accelerometer. www.adafruit.com, 2017.

[95] Pico TA018 AC/DC Current Probe. www.picotech.com, 2017.

[96] Magnetometer digital triple axis - hmc5883l. www.honeywell.com, 2017.

[97] Thingiverse. https://www.thingiverse.com/., 2017.

[98] Scott D Applegate. The dawn of kinetic cyber. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–15. IEEE, 2013.

[99] Nicolas Falliere et al. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5, 2011.

[100] Jill Slay and Michael Miller. *Lessons learned from the maroochy water breach*. Springer, 2007.

[101] Robert M Lee, Michael J Assante, and Tim Conway. German steel mill cyber attack. *Industrial Control Systems*, 30, 2014.

[102] Karl Koscher et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[103] L Sturm, C Williams, J Camelio, et al. Cyber-physical vunerabilities in additive manufacturing systems. *Context*, (2014), 2014.

[104] Mohammad Al Faruque, Francesco Regazzoni, and Miroslav Pajic. Design methodologies for securing cyber-physical systems. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*. IEEE Press, 2015.

[105] Jiang Wan et al. Exploiting wireless channel randomness to generate keys for automotive cyber-physical system security. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[106] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 2014.

[107] Hannah Vincent, Lee Wells, et al. Trojan detection and side-channel analyses for cybersecurity in cyber-physical manufacturing systems. *Procedia Manufacturing*, 2015.

[108] Mark Yampolskiy, Lena Schutzle, et al. Security challenges of additive manufacturing with metals and alloys. In *International Conference on Critical Infrastructure Protection*. Springer, 2015.

[109] Jeff C Jensen et al. A model-based design methodology for cyber-physical systems. In *Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2011.

[110] Ranganath Kothamasu et al. System health monitoring and prognostics–a review of current paradigms and practices. In *Handbook of Maintenance Management and Engineering*, pages 337–362. Springer, 2009.

[111] Mikell P Groover. *Automation, production systems, and computer-integrated manufacturing*. Prentice Hall Press, 2007.

[112] Mohammad Al Faruque et al. Acoustic side-channel attacks on additive manufacturing systems. In *ACM*, 2016.

[113] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[114] Alvaro Cardenas et al. Challenges for securing cyber physical systems. In *Workshop on future directions in cyber-physical systems security*, 2009.

[115] Brian Evans. *Practical 3D printers: The science and art of 3D printing*. Apress, 2012.

[116] J Hendershot. Causes and sources of audible noise in electrical motors. In *Incremental Motion Control Systems and Devices Symposium*, 1993.

[117] k-quad 5.1 250mm quadcopter frame. thingiverse, http://www.thingiverse.com/thing:397036, 2014.

[118] László Monostori. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 2014.

[119] Washington Post, `http://www.washingtonpost.com/wp-dyn/content/article/2008/06/05/AR2008060501958.html`, 2008.

[120] Ravi Akella, Han Tang, et al. Analysis of information flow security in cyber–physical systems. *International Journal of Critical Infrastructure Protection*, 3(3):157–173, 2010.

[121] Ian Goodfellow, Jean Pouget-Abadie, et al. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

[122] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[123] Steven Eric Zeltmann, Nikhil Gupta, et al. Manufacturing and security challenges in 3d printing. *Jom*, 2016.

[124] Jiang Wan, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Cyber–physical codesign at the functional level for multidomain automotive systems. *IEEE Systems Journal*, 2015.

[125] Jay Lee, Behrad Bagheri, and Chao Jin. Introduction to cyber manufacturing. *Manufacturing Letters*, 8:11–15, 2016.

[126] Jay Lee et al. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 2015.

[127] Ying Peng and Ming Dong. A prognosis method using age-dependent hidden semi-markov model for equipment health prediction. *Mechanical Systems and Signal Processing*, 25(1):237–252, 2011.

[128] Ranganath Kothamasu, Samuel H Huang, and William H VerDuin. System health monitoring and prognosticsa review of current paradigms and practices. *The International Journal of Advanced Manufacturing Technology*, 28(9-10):1012–1024, 2006.

[129] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[130] Smart Factory Task Group. Smart factory applications in discrete manufacturing. `https://goo.gl/tT9Sf4`, 2017.

[131] Roland Rosen, Georg von Wichert, et al. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 2015.

[132] Jay Lee, Edzel Lapira, Behrad Bagheri, and Hung-an Kao. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, 1(1):38–41, 2013.

[133] Siemens. On Track for the future - the Siemens Digital Twin show. https://www.youtube.com/watch?v=GzPWBHT1Hl4, 2015.

[134] IBM. Introduction to digital twin: Simple, but detailed. https://www.youtube.com/watch?v=RaOejcczPas, 2017.

[135] Michael Grieves. Digital twin: Manufacturing excellence through virtual factory replication. *White paper*, 2014.

[136] Mike Bacidore. Digital twin to enable asset optimization. https://www.smartindustry.com/articles/2015/digital-twin-to-enable-asset-optimization/, 2015.

[137] Ryan Kucera, Mike Aanenson, and Mark Benson. The augmented digital twin. https://goo.gl/RHSwZQ, 2017.

[138] Collin Parris, Brandon Laflen, et al. The future for industrial services: The digital twin. https://www.infosys.com/insights/services-being-digital/Documents/future-industrial-digital.pdf, 2017.

[139] Bernard Marr. What is Digital Twin Technology-And Why is it so important. https://goo.gl/v3cGwF, 2017.

[140] NSF I/UCRC Center for Intelligent Maintenance Systems (IMS). Digital twin for machine monitoring - cyber-physical interface for manufacturing, ims center. https://www.youtube.com/watch?v=6sh4U44AndQ, 2014.

[141] Eric J Tuegel et al. Reengineering aircraft structural life prediction using a digital twin. *Journal of Aerospace Engineering*, 2011.

[142] IBM. Creating a building's 'digital twin'. https://www.ibm.com/internet-of-things/iot-zones/iot-buildings/sensors-in-intelligent-buildings/, 2017.

[143] ANSYS. Excellence in engineering solutions, advantave: Spotlight on the digital twin. http://www.ansys.com/-/media/Ansys/corporate/resourcelibrary/article/ansys-advantage-digital-twin-aa-v11-i1.pdf, 2017.

[144] Mike Bacidore. The connected plant enables the digital twin. https://www.controlglobal.com/industrynews/2017/hug-7/, 2017.

[145] TWI. Twi embarks on lifecycle engineering asset management through digital twin technology. https://goo.gl/WxLPcy, 2017.

[146] T DebRoy, W Zhang, et al. Building digital twins of 3D printing machines. *Scripta Materialia*, 2017.

[147] GL Knapp, T Mukherjee, et al. Building blocks for a digital twin of additive manufacturing. *Acta Materialia*, 2017.

[148] Frederica Darema. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. *Computational Science-ICCS 2004*, pages 662–669, 2004.

[149] Richard S Hunter. Photoelectric color difference meter. *Josa*, 48(12):985–995, 1958.

[150] Zhi-Qiang Liu. Scale space approach to directional analysis of images. *Applied optics*, 30(11):1369–1373, 1991.

[151] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[152] Alin Dobra and Johannes Gehrke. Secret: a scalable linear regression tree algorithm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 481–487. ACM, 2002.

[153] Matti Maltamo and Annika Kangas. Methods based on k-nearest neighbor regression in the prediction of basal area diameter distribution. *Canadian Journal of Forest Research*, 28(8):1107–1115, 1998.

[154] Michael Collins, Robert E Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1):253–285, 2002.

[155] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[156] T Mukherjee and Tarasankar DebRoy. A digital twin for rapid qualification of 3d printed metallic components. *Applied Materials Today*, 2019.

[157] Felix Baumann, Manuel Schön, Julian Eichhoff, and Dieter Roller. Concept development of a sensor array for 3d printer. *Procedia CIRP*, 2016.

[158] Yuan He, Junchen Guo, et al. From surveillance to digital twin: Challenges and recent advances of signal processing for industrial internet of things. *Signal Processing Magazine*, 2018.

[159] Ahmed Banafa. Iot standardization and implementation challenges. *IEEE. org Newsletter*, 2016.

[160] Alexey S Petrenko, Sergei A Petrenko, et al. The iiot/iot device control model based on narrow-band iot (nb-iot). In *Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 2018.

[161] Chen Song, Feng Lin, et al. My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[162] Stefan Boschert and Roland Rosen. Digital twin-the simulation aspect. In *Mechatronic Futures*. Springer, 2016.

[163] Kazi Masudul Alam and Abdulmotaleb El Saddik. C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems. *Access*, 2017.

[164] Greyce N Schroeder et al. Digital twin data modeling with automationml and a communication methodology for data exchange. *IFAC-PapersOnLine*, 2016.

[165] Albert Cerrone et al. On the effects of modeling as-manufactured geometry: toward digital twin. *Journal of Aerospace Engineering*, 2014.

[166] Kévin Garanger, Eric Feron, et al. Foundations of Intelligent Additive Manufacturing. *arXiv preprint:1705.00960*, 2017.

[167] L Di Angelo, P Di Stefano, and A Marzola. Surface quality prediction in fdm additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 2017.

[168] Alberto Boschetto, Luana Bottini, and Francesco Veniali. Integration of fdm surface quality modeling with process design. *Additive Manufacturing*, 2016.

[169] SA Shevchik, C Kenel, C Leinenbach, and K Wasmer. Acoustic emission for in situ quality monitoring in additive manufacturing using spectral convolutional neural networks. *Additive Manufacturing*, 2018.

[170] K Wasmer, C Kenel, et al. In situ and real-time monitoring of powder-bed am by combining acoustic emission and artificial intelligence. In *International Conference on Additive Manufacturing in Products and Applications*. Springer, 2017.

[171] Prahalad K Rao, Jia Peter Liu, et al. Online real-time quality monitoring in additive manufacturing processes using heterogeneous sensors. *Journal of Manufacturing Science and Engineering*, 2015.

[172] Hongyue Sun, Prahalad K Rao, et al. Functional quantitative and qualitative models for quality modeling in a fused deposition modeling process. *Transactions on Automation Science and Engineering*, 2018.

[173] Kaveh Bastani, Prahalad K Rao, et al. An online sparse estimation-based classification approach for real-time monitoring in advanced manufacturing processes from heterogeneous sensor data. *IIE Transactions*, 2016.

[174] Li Zhu, Charlotta Johnsson, et al. Key performance indicators for manufacturing operations management in the process industry. In *Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2017.

[175] Maximilian Christ et al. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint:1610.07717*, 2016.

[176] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*. Springer, 2009.

[177] Jane Elith, John R Leathwick, and Trevor Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 2008.

[178] Postscapes. Iot Sensors and Actuators. `https://www.postscapes.com/trackers/video/the-internet-of-things-and-sensors-and-actuators/`, 2018.

[179] Ultimaker. Ultimaker 3. `https://ultimaker.com/en/products/ultimaker-3`, 2018.

[180] Haixi Wu, Yan Wang, and Zhonghua Yu. In situ monitoring of fdm machine condition via acoustic emission. *The International Journal of Advanced Manufacturing Technology*, 2016.

[181] Alberto Boschetto and Luana Bottini. Accuracy prediction in fused deposition modeling. *The international journal of advanced manufacturing technology*, 2014.

[182] Haixi Wu, Zhonghua Yu, and Yan Wang. Real-time fdm machine condition monitoring and diagnosis based on acoustic emission and hidden semi-markov model. *Journal of Advanced Manufacturing Technology*, 2017.

[183] Jae Yoon, David He, et al. A phm approach to additive manufacturing equipment health monitoring, fault diagnosis, and quality control. In *Prognostics and Health Management Society Conference*. Citeseer, 2014.

[184] Yousef Haik, Sangarappillai Sivaloganathan, and Tamer M Shahin. *Engineering design process*. Nelson Education, 2018.

[185] Bowen Li and Paul D Franzon. Machine learning in physical design. In *Electrical Performance Of Electronic Packaging And Systems (EPEPS), 2016 IEEE 25th Conference on*, pages 147–150. IEEE, 2016.

[186] Manish Pandey. Machine learning and systems for building the next generation of eda tools. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, pages 411–415. IEEE Press, 2018.

[187] Weiyi Qi. *IC Design Analysis, Optimization and Reuse via Machine Learning*. North Carolina State University, 2017.

[188] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George W Fitzmaurice. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *UIST*, pages 401–414, 2017.

[189] Xiang'Anthony' Chen, Ye Tao, Guanyun Wang, Runchang Kang, Tovi Grossman, Stelian Coros, and Scott E Hudson. Forte: User-driven generative design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 496. ACM, 2018.

[190] Geometric Deep Learning. `http://geometricdeeplearning.com/`, 2018.

[191] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[192] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[193] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[194] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[195] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[196] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[197] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[198] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[199] Fan RK Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[200] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *arXiv preprint arXiv:1705.07664*, 2017.

[201] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017.

[202] Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. *arXiv preprint arXiv:1802.08888*, 2018.

[203] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[204] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[205] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[206] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[207] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[208] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

[209] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 543–552. ACM, 2014.

[210] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[211] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[212] Dario Garcia-Gasulla, Eduard Ayguadé, Jesús Labarta, Javier Béjar, Ulises Cortés, Toyotaro Suzumura, and R Chen. A visual embedding for the unsupervised extraction of abstract semantics. *Cognitive Systems Research*, 42:73–81, 2017.

[213] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[214] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

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

[216] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*, 2016.

[217] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 170–182. Springer, 2018.

[218] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016.

[219] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 kdd cup. *ACM SIGKDD Explorations*, 5(2), 2003.

[220] Linton C Freeman. Visualizing social networks. *Journal of social structure*, 1(1):4, 2000.

[221] Athanasios Theocharidis, Stjin Van Dongen, Anton Enright, and Tom Freeman. Network visualization and analysis of gene expression data using biolayout express3d. *Nature protocols*, 4:1535–1550, 2009.

[222] Palash Goyal, Anna Sapienza, and Emilio Ferrara. Recommending teammates with deep neural networks. In *Proceedings of the 29th on Hypertext and Social Media*, pages 57–61. ACM, 2018.

[223] Georgios A Pavlopoulos, Anna-Lynn Wegener, and Reinhard Schneider. A survey of visualization tools for biological network analysis. *Biodata mining*, 1(1):12, 2008.

[224] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[225] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 2018.

[226] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[227] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proc. of ACM SIGKDD*, pages 1105–1114, 2016.

[228] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.

[229] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings 20th international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[230] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *KDD15*, pages 891–900, 2015.

[231] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings 24th International Conference on World Wide Web*, pages 1067–1077, 2015.

[232] Palash Goyal, Homa Hosseinmardi, Emilio Ferrara, and Aram Galstyan. Embedding networks with edge attributes. In *Proceedings of the 29th on Hypertext and Social Media*, pages 38–42. ACM, 2018.

[233] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang. Dynamic Network Embedding by Modelling Triadic Closure Process. In *AAAI*, 2018.

[234] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[235] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. Timers: Error-bounded svd restart on dynamic networks. *arXiv preprint arXiv:1711.09541*, 2017.

[236] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[237] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234. ACM, 2016.

[238] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1145–1152. AAAI Press, 2016.

[239] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[240] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[241] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2765–2777, 2016.

[242] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. In *IJCAI International Workshop on Representation Learning for Graphs*, 2017.

[243] Mahmudur Rahman, Tanay Kumar Saha, Mohammad Al Hasan, Kevin S Xu, and Chandan K Reddy. Dylink2vec: Effective feature representation for link prediction in dynamic networks. *arXiv preprint arXiv:1804.05755*, 2018.

[244] Purnamrita Sarkar, Deepayan Chakrabarti, and Michael Jordan. Nonparametric link prediction in dynamic networks. *arXiv preprint arXiv:1206.6394*, 2012.

[245] Shuo Yang, Tushar Khot, Kristian Kersting, and Sriraam Natarajan. Learning continuous-time bayesian networks in relational domains: A non-parametric approach. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[246] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

[247] Xiaoke Ma, Penggang Sun, and Yu Wang. Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks. *Physica A: Statistical mechanics and its applications*, 496:121–136, 2018.

[248] Nitish Talasu, Annapurna Jonnalagadda, S Sai Akshaya Pillai, and Jampani Rahul. A link prediction based approach for recommendation systems. In *2017 international conference on advances in computing, communications and informatics (ICACCI)*, pages 2059–2062. IEEE, 2017.

[249] Jundong Li, Kewei Cheng, Liang Wu, and Huan Liu. Streaming link prediction on dynamic attributed networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 369–377. ACM, 2018.

[250] Yuchung J Wang and George Y Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

[251] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Neurocomputing: Foundations of research. *JA Anderson and E. Rosenfeld, Eds*, pages 696–699, 1988.

[252] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[253] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[254] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.