Dissertation Type: Applied

DEPARTMENT OF ENGINEERING MATHEMATICS

Control Logic Based Cyber Attacks in Industrial Control Systems

Laurence Browne

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Friday 27th September, 2024

Supervisor: Dr. Sridhar Adepu

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Laurence Browne, Friday 27th September, 2024

Abstract

This paper investigates the use of machine learning techniques to model the operation of, then create attacks against Cyber Physical Systems (CPS). CPS have a physical aspect, which measure and influence the real world through sensors and actuators, and a cyber aspect, which provide control and monitoring functions.

CPS are traditionally used in industrial process but with the advent of the Internet of Things (IoT) they now proliferate modern life- from home automation systems and autonomous vehicles through to the Critical National Infrastructure such as power grids and public transport networks. As such, understanding the threats posed to these system by emerging technologies is a key area of research for both the public and private sector.

This paper focuses on Secure Water Treatment testbeds (SWaT) as variants of these are present in the majority of industrial process and include a range of components which are common to other processes. Initially a cyber attack methodology was applied to identify aspects of the SWaT which are vulnerable when an 'insider' access level is assumed- that is, full knowledge of, and access to all aspects of the system (including any anomaly detection systems).

A cyber-attack can have multiple facets, investigated here are:

- False anomalies to provoke unnecessary maintenance
- Spoofing control signals to increase wear/ impact system operation
- Hiding/ spoofing error messages to allow components to be damaged

Supporting Technologies

Third-party resources used during the project:

- Development of the SVM and GAN models was performed on a AMD Ryzen CPU and NVIDIA A4000 GPU running Ubuntu V22.04.03. Pycharm Professiononal 2023.1.3 IDE was used for the code development
- Real-Time Modelling of the SWaT was performed in a VMWare Workstation Player virtual machine running Ubuntu V22.04.03. The Mininet network simulation tool with a custom MiniCPS SWaT model. Wireshark was used to analyse network traffic.
- I used the *Pandas* and *Seaborn* public-domian Python Libraries.
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Notation and Acronyms

The following are terms used within the report:

PLC	:	Programable Logic Controller
SWaT	:	Secure Water Treatment Testbed
CPS	:	Cyber Physical System
HMI	:	Human Machine Interface
SCADA	:	Supervisory Control and Data Acquisition
Ethernet/IP	:	Industrial Ethernet
ADC	:	Analogue to Digital Converter
DAC	:	Digital to Analogue Converter
OSI	:	
Sensor	:	
Actuator	:	

Acknowledgements

An optional section, of at most 1 page

TBC.

TBC.

Chapter 1

Introduction

Overview of AI in Cyber Physical Systems

The water treatment Cyber Physical System is a collection of sensors which produce continuous signals which are analogous to physical properties of the system such as water level, fluid flow rates and temperature. A Programmable Logic Controller (PLC) monitors these measurements and applies pre-programmed logic to output signals which operate various switches and actuators. The effect of this is that the PLC is a standalone micro-controller which is able to controller an industrial process autonomously. For example, a PLC receiving a low water level signal as an input will send a control signal to a pump or valve which will operate until the input signal is again within range.

The internal logic of the PLC is designed to account for all possible system states and includes parameters such as maximum temperatures, pressures and flow rates. Should the system move outside of these parameters the PLC will take appropriate action and raise an alert. The signals between the PLC and the input and output components are said to be at Level 0 of the OSI model as they either analogue or digital signals in a range suitable for operating switches etc. These signals are typically continuous voltage signals which are converted into a current range between 4 I& 20 mA to avoid transmission losses. This current signal is processed by an external Analogue to Digital Converter (ADC) or Digital to Analogue Converter (DAC) as appropriate as the PLC uses binary encoded values at Level 1 of the OSI model.

A CPS will consist of multiple stages, each performing a distinct process and controlled by it's own PLC. These PLC's communicate with each other in order to pass or request data pertinent to their own stage of the process e.g. request a faster flow of water or indicate a batch is ready to move to the next stage. There are multiple communications protocol's used for this PLC – PLC communication, these are often proprietary to a particular manufacturer but are usually based on the Ethernet/IP industrial Ethernet standard. This configuration is known as Industrial Internet of Things iIOT or Industry 4.0. The updating of the logic within the PLC's is accomplished via a Human Machine Interface which communicates on the same Ethernet/IP ring network as the PLC's. The monitoring of process parameters is performed by Supervisory Control and Data Acquisition (SCADA) sytem which may also play a Historian function and record sytems parameters.

Machine Learning is already applied to these CPS in the form of anomaly detectors. These are usually Support Vector Machine or Deep Neural Networks which have been trained to classify normal and anomalous behaviours. These anomaly detectors may reside on the SCADA system or possinble on a stand-alone processore at PLC level- an example of edge computing. The anomaly detector is often a Support Vector Machine or Deep Neural Network operating as a classifier having been trained on data from normal operation and simulated attacks.

A cyber vulnerability methodology was used to identify vulnerabilities suitable for machine learning techniques to be applied to. The study assumes an 'insider' level of access where the attack has full access to the system- this allows manipulation of the raw signals at level 0 through to the high-level TCP/IP packets which carry the control and monitoring messages.

A SVM Classifier was trained on the sample data to act as a baseline then a Generative Adversarial Network (GAN) was created to generate data for the attack. A GAN consists of a Classifier and a Generator where the error from the classifier is used to improve the generator until it can successfully produce data which can fool the classifier.

This method was applied to the individual Stages of the SWaT system and to system as a whole.

Project Aim and Scope

The SWaT is a microcosm of the cyber physical systems which are present in most aspect of modern life. The data provided by the iTrust Centre for Research in Cyber Security would represent not only an insider level of access, where all system information is known, but also provides data of the system under various forms of attack.

The aim of this project is to devleop an AI based approach which can learn the system the system under normal operation (passively) and then probe it to verify and update it's beliefs by manipulating the system (actively). The aim is to have an intelligent agent which could be implemented at the edge of CPS and either work completely automomously or near lautomomously so that only minimal comunitcation with the attacker was required.

The applications for this approach are useful for both cyber defence and offense. Typically, GAN Neural Network based anomaly detectors are used to flag behaviour which may indicate an attack. These are effective but are vulnerable to gradient based attacks. The downside of these defencees is their black-box nature. Teh detectors is based on an abstract funciton of all system parameters. At a lower level there are rules based approaches which look for the system to remain with in expected parameteres, these again are effective but are reliant on the rules being definined and updated.

The AI agent I'm preposinsing will initially learn the system and be able to create a network diagram of all components, their interactions and normal operationg parameters. This could operate as a dyanamic rule-based anomaly detection system so providing the system/ data generated model like the GANs but with the explainability of the rule based system (clear tracking of which system components were amanipulated).

Chapter 2

Literature Review

Introduction to SWat Testbed, iTrust Centre for Cyber Security

2.1 Introduction to SWaT Testbed, iTrust Centre for Cyber Security

The iTrust Centre for Cyber Security operates a Secure Water Treatment Testbed (SWaT) which is used to “support research into the design of secure, public infrastructure” [1]. SWaT is a Cyber Physical System (CPS) so consists of the physical side which implements a process and a cyber side which performs control, monitoring and security. The testbed produces clean water by using both Ultra Filtration and Reverse Osmosis which is implemented through a six stage, distributed control system which supports wired and wireless communications.

SWaT Stage 1- Raw Water Processing:

SWaT Stage 1- Raw Water Processing: An Alan Bradley Programmable Logic Controllers (PLC) acts as the primary controller for this stage and a second provides redundancy in case of a failure in the primary. Each stage in SWaT uses this dual controller configuration. The naming convention is PLC followed by the stage number, the backup PLC also has ‘b’ appended to the name so for Stage 1 the PLC’s are ‘PLC1’ I& ‘PLC1b’ The PLC manages the flow of raw water from the inlet into the SWaT Stage 2. A motorised valve controls the flow into a storage tank which has four markings (HH, H, L, LL) which indicate maximum to minimum water levels. These markings correspond to numerical water level readings used by the PLC to control the inlet valve and a constant speed pump which feeds water to Stage 2. The numerical water level values used by the PLC are produced by a ultrasonic, water level sensor which uses current signalling (in the 4-20mA range). This analogue signal is digitised for use by the PLC, all tanks within the SWaT use this type of water level sensing. A pH and a Oxidation Reduction Potential (ORP) sensor are present after the constant speed pump and the measurements they produce are sent to the Stage 2 PLC.

SWaT Stage 2- Chemical Dosing/ Pre-Treatment:

Stage 2 controls the addition of three separate chemicals to the water from Stage 1- Sodium Hypochlorite ($NaOCl$), Hydrochloric Acid (HCl) and Sodium Chloride ($NaCl$). These chemical are used to balance the pH and ORP of the water as well as disinfection. Dual dosing pumps are used (presumably for redundancy as with the PLCs) and these add the chemicals into the water feed at a rate determined by the Stage 2 PLC.

SWaT Stage 3- Ultrafiltration:

Stage 3 consists of two water tanks with an ultrafiltration unit in between. Tank T301 holds the water from Stage 2 prior to it being pumped through the filter and tank T401 hod the filtered water prior to it entering Stage 4. The ultrafiltration unit consists of progressively fine, micrometer membranes which remove particulate matter. Stage 3 consists of several motorised valves and pressure sensors which control the flow of water through the filter. The filters become clogged with use so a differential pressure sensor is used to indicate when the pressure across the unit increases- this signal is used by the PLC to instigate a cleaning cycle (back flush of the system). There are additional flow and pressure sensors which monitor the properties of water entering and exiting the filter.

SWaT Stage 4- De-chlorinisation:

In order to prevent oxidation of the membranes within the Reverse Osmosis (RO) unit, chlorine is removed from the water coming from tank T401 using an Ultra Violet de-chlorination unit (UV 401) and also Sodium Bi-sulphate (NaHSO3) from tank T402 if required. An ORP monitor is used to ensure the Chlorine has been removed. The Stage 3 PLC controls this stage of the process.

SWaT Stage 5- Reverse Osmosis:

The RO stage is the most complex as the nano-filters within the RO units (numbered RO 501, RO 502 I& RO 503) are sensitive to particulates or chlorine which were missed by the previous stages. Water which successfully permeates the RO filters is sent to tank T601, this is final product of the system (the clean water). It is recycled by being sent back to Stage 1. Water that does not permeate through the RO filters is sent to tank T602, this is the reject water and is used to clean the ultrafiltration unit in Stage 3. Stage 5 has motorised valves, flow metres, pH and ORP sensors in order to protect the RO filters. It also has a cartridge filter.

SWaT Stage 6- Backwash:

The SWaT testbed is programmed to initiate a cleaning cycle every 30 minutes which is controlled by the PLC in Stage 6. An additional cleaning process of back-flushing the ultra-filtration unit (to remove particulates from the filters) is instigated when the signal from the differential pressure sensor in Stage 3 exceeds a pre-defined value. The water from these processes is taken from the Stage 5 reject tank (T602) and is expelled from the system after use.

SWaT Network

The control network (or cyber component) is split into two main parts- Layer 0 and Layer 1.

The Layer 0 network is where the PLC headers interface with the actuators and sensors (peripherals) in order to interact with the physical processes. Layer 0 is taken to be at a photon/ electron level i.e. sub-bit level, the PLC's are unable to interpret or produce analogue signals so continuous signals undergo either Analogue to Digital Conversion (ADC) or Digital to Analogue Conversion (DAC) as needed within the Remote Input Output (RIO) unit. The PLC run software for the control logic of these peripherals (via the RIO) and is connected by a ethernet based circular/ ring network to the RIO and backup PLC.

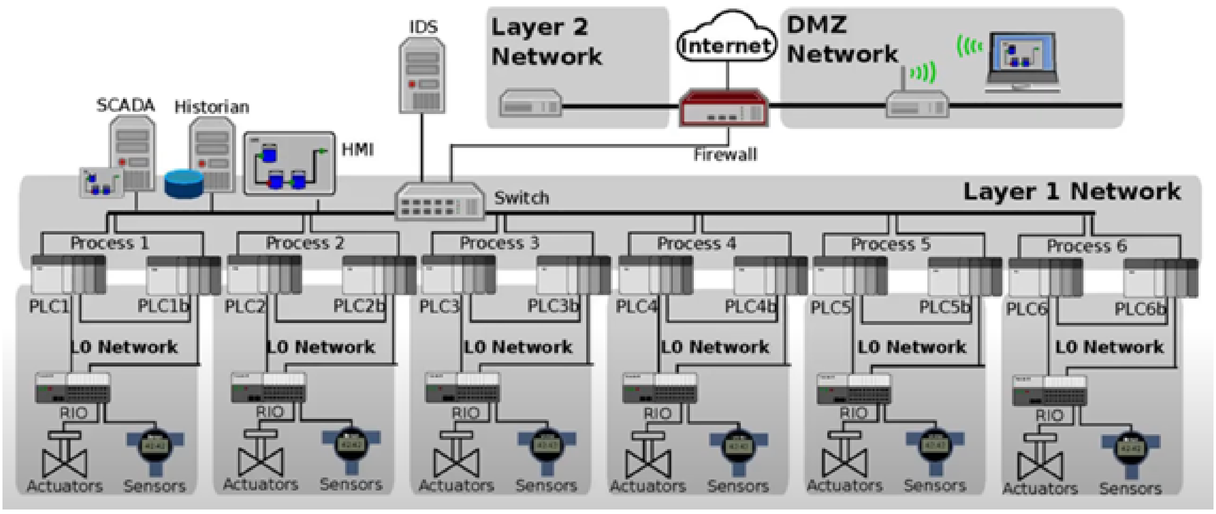
In the Layer 1 Network the PLCs are connected to each other (using the manufactures proprietary protocol), a SCADA system, a network Historian and a Human Machine Interface. These devices are configured by a star network via a central, 24 port switch. This is Layer 1 in the OSI model as it is the physical layer where packets of binary data are passed, rather than the analogue or binary encoded analogue values passed at Layer 1. The protocol used on this Layer 1 network is not TCP/IP but a specific protocol for control systems which is built on top of TCP and allows data such as that for programming/ updating the logic of the PLCs or firmware updates to be passed. This protocol is also used on the ring part of the Level 0 network where the PLCs and RIO communicate (along with other, undefined components which can communicate via ethernet).

The central switch also used 4 ports to export copies of all the network traffic for use in intrusion detection system development. Both network layers can be changed from wired to wireless communication via a switch on the cabinet for each Stage.

2.2 Threat Modeling of Cyber-Physical Systems - A Case Study of a Microgrid System

Shaymaa Mamdouh Khalil a, Hayretdin Bahsi, Henry Ochieng' Dolaa, Tarmo Korõtko, Kieran Laughlinc, Vahur Kotkas

Figure 2.1: SWaT Network Diagram



"Cyber threat modeling is an analytical process that is used for identifying the potential threats against a system" [3]. The article seeks to apply the secure-by-design model from software development to CPS which it states has not been well catered for with current threat models. The paper points out that the CPS are more varied in their makeup than purely software systems so require a input from a wider range of stakeholders such as experts who deal with physical processes. This aligns with the Level 0 in the SWaT testbed whereby the physical aspects of the system are a vector which could be used to attack the system, the paper suggests that software based security models assume the physical aspects of the system (i.e. access to equipment) is secure. The research attempts to develop existing security practices and map these to the IEC 62443 standard which addresses cyber security for automation and control systems. This is implemented in the following 9 stage Threat Modelling Methodology.

Process

Stage 1: Initial Attack Taxonomy Creations Review literature on attacks on similar systems in order to become familiar with the system and it's security issues. Stage 2: Information I& Systems Assets Identification Identify all system assets whether or not they are within the scope of the threat modelling exercise. In CPS there is likely in two distinct information categories- control and measurement. This could take the form of a system architecture diagram. Stage 3: System Mapping into Data Flow Digagram This stage allows the visualisation of assets which produce/ use data but do not have computing capability so would be outside the scope of conventional cyber threat modelling. Stage 4: Security Context Definition Agrees the physical security assumptions such as who is trusted with admin access and the main threat actors. Stage 5: Trust Boundaries Determination

Stage 6: Threat Elicitation and Attack Taxonomy Update Applies STRIDE to each element in the DFD or to information flows which cross a trust boundary. Stage 7: Threat Consequences I& Losses Identification Cyber security experts work with system experts to identify real world consequences of each threat. Stage 8: Threat Prioritisation Highest impact threats prioritised. Stage 9: Security Requirements Selection System requirements required in order to counter identified threats.

2.3 Adversarial Attacks and Mitigation for Anomaly Detectors of Cyber-Physical Systems

Chen et al describe the methods used by CPS to identify anomalous behaviour which is indicative of a cyber attack. They describe that typically a CPS has two forms of defence: Firstly, an anomaly detector which is a Machine Learning model (often based on a neural network model) which is trained on the systems physical data. Secondly, rule checkers (or invariant checkers) are used which check values against the acceptable parameters or known relationships between components in the CPS . Chen et al assume a 'white' box level of access to the anomaly detector, that is a full understanding of it's behavious and accesst to the data it was trained on .It is assumed the rule checker is only a black or grey level of access so it's behaviour must be learnt from the librarian logs etc.

The team 'crafts noise' over the signal between actuators and sensors then use a 'genetic algorithm' to optimise the noise so that both detection systems are deceived to the degree that their classification accuracy is reduced by over 50

The report mentions how attacks on the CPS typically involve spoofing or manipulating the network packets and neural network based detectors are effective at identifying these. This paper seeks to create attack possible when there is 'insider' level access- the attacker knows the anomaly model. The focus of the paper is to create noise which will lead the anomaly detector and rules checkers to misclassify the activity. For example, if the attack can use noise to shrink the difference between the actual value and the predicted value then the anomaly detector will assert more false positives "when a detector misclassifies a real attack as normal behaviour" [2]. Jiaa et al assert that "existing adversarial attacks have limited effectiveness in the presence of rule checkers" but that genetic algorithms based on the white-box gradient based approach can remedy this.

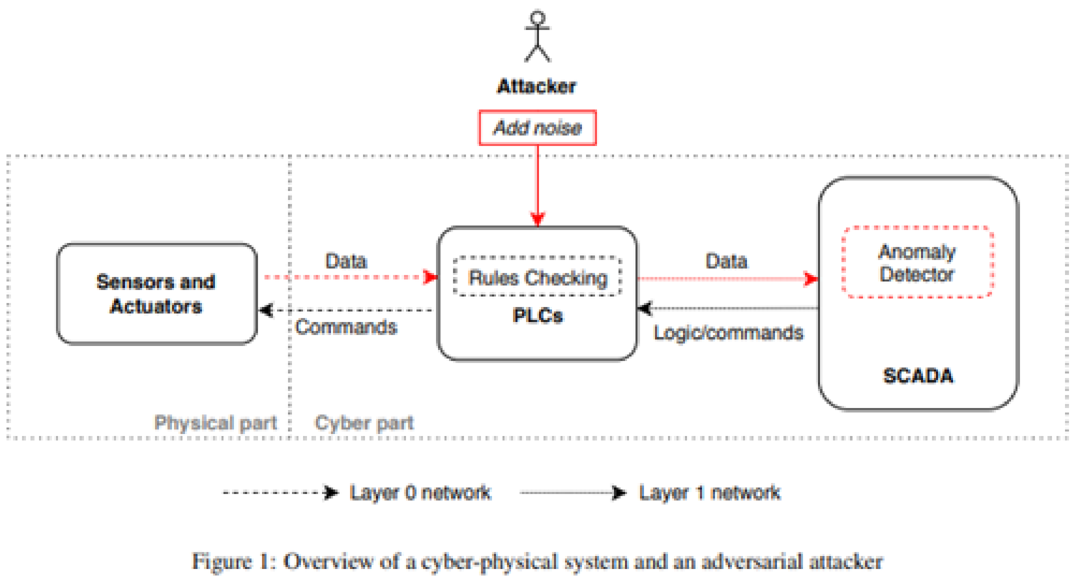
The paper defines a CPS as PLC's which are connected to actuators and sensors which are the interface to the physical world. The PLC run software for the control logic of these peripherals which it is connected to by a circular/ ring network operating at 'Layer 0'. Layer 0 is taken to be at a photon/ electron level- i.e. sub-bit level so continuous or discrete signals. These PLCs are connected to a central SCADA system by a star network operating at layer 1- the physical layer.

It is assumed that rule checkers reside within the PLCs- for example to open a valve using an actuator when a particular sensor value is met. The anomaly detector is assumed to reside on the SCADA system.

The paper describes the two test beds used for the research – the Swat I& WADI plants that model a water treatment and water distribution plants respectively. The SWAT plant is described as having 68 sensors and actuators in total, a number of these are standby in case of failures and were not cond=sidered in the paper. It is noted that the sensors are typically continuous values and the acutators are discrete. This is understandable as the output of the PLC is likely to a motor controlller or relay which handle things like soft start for motors or gradual closing of valves in order to avoid the water hammer effect (me).

'Our approach is inspired by a white-box gradient-based approach [33],' N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: 2016 IEEE European Symposium on Security and Privacy (EuroSI&P), IEEE, 2016, pp. 372–387

Figure 2.2: Adversarial attack Diagram



The SWaT testbed has a historian which records the physical state of the system, this is "a fixed ordering of all the sensor readings and actuator configurations at a particular timepoint" [3]. The report uses the following notation to denote the system state (x) where subscript a and s are used for acutators and sensors respectively.

Figure 2.3: System State Formula

$$x = [x_{a1}, x_{a2}, x_{a3} \ldots x_{s1}, x_{s2}, x_{s3} \ldots]$$

SCADA, Historian and Human Machine Interface workstations sit at higher levels and allow operations such as changing control code/ parameters within the PLCs. The Threat model used is White Box where attacker has access to physical signals at layer 0, full knowledge of the RNN based anomaly detector but can only judge rule checker from Status in the historian. The authors use gradient based methods where by the original attack signals have noise added which is basedon the loss gradient of the RNN. This does not affect the attack but leads to the attack being misclassified.

Chapter 3

Passive System Modelling

3.1 Generating Initial Beliefs

The CPS is a Deterministic System as the operation of each component is determined my rules-based logic applied to the signals it receives from it's related components or control signals from the SCADA???. Though the system is deterministic it can be useful to use probabilistic models initially as these can account for the complex interactions between components. The first stage of this approach is identify all components in the system, this is done by extracting the columns from the SWaT data but would be done by identifying all unique IP addresses in the industrial CPS (as these are usually fixed as no DHCP like normal ethernet).

The source data is collected at 1 second intervals in the historian and over several days, though possible to process all of this data (as there are only ..60??? unique components), the aim is to produce an agent which can use only the limited resources of an Edge device as such, various techniques were investigated to find an efficient way to map the system.

The CPS will work in a cyclical way as the controls work to find an equilibrium where demand for clean water is met by the instructions given to the sub-stages. The process is linear in that polluted water enters the first stage and exit the final stage after passing through every subsequent stage. Within some substages there may be cyclical processes which continue to until the water reaches the required state.

3.2 Data Preparation

The SWaT dataset contains raw values for the system which includes a range of characteristics for each component type. It also includes data for components which are used in case of failure of the primary component. These components are not used in normal operation so their values do not change. As these add no information these were dropped which resulted in a reduction from .... to 37 components.

The dataset had no missing values and initial analysis gave confidence that there were no erroneous outliers. A version of the dataset which used the original values and one which used the min max scaler (where the original value range is scaled to between 0 and 1) were used. It was important to have both version of the data as many methods such as training neural networks are sensitive to magnitude variation so require normalised data. As the system is an industrial process the raw values are useful in operations such identifying similarities in components.

Analysis of the waveform suggested that the data did not suffer from excessive noise- minor perturbations to signal levels were likely due to the characteristics of the electronic components in the system such as the granularity of the pump motor step size when maintaining a PLC set level.

The main, normal operation dataset was too large to use with the Git version control system so was split into 24 hour periods. The original Timestamp index was converted to a seconds-since-midnight value to simplify analysis. To aid initial data exploration a 10 sec mean was taken of all values as this reduced dataset reduced the processing overhead with minimal information loss.

To manage the manipulation of the large number of parameters being created during the study, a Component Class was created which allowed attributes created through the pipeline to added and accessed in a consistent way. Class methods facilitated retrieval of information such as relationship strength between component and common visualisations.

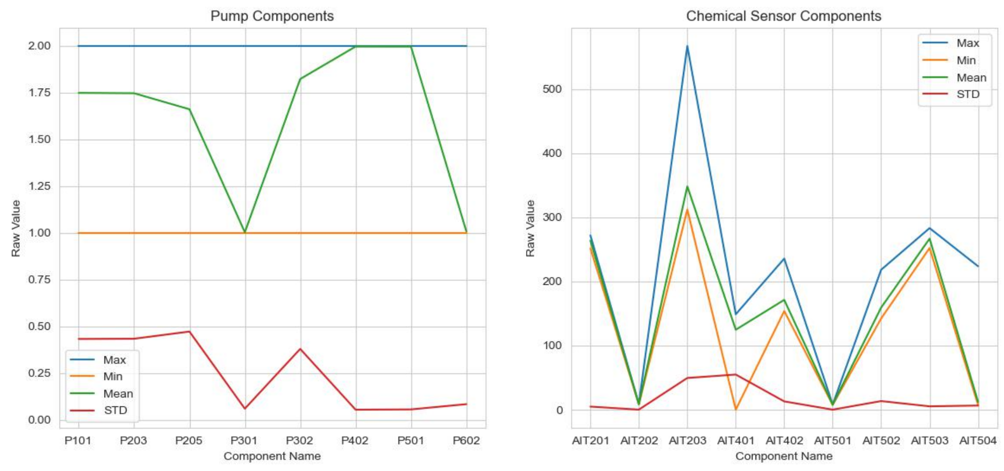
The Component Class was defined inside a module alongside the functions were used to manage the data conditioning. This ensured consistency across dataset and supported the pipeline approach.

3.3 Intial Component Comparison

Initial grouping on components was performed using the raw values from the dataset. As these values represent physical values on the most part, simply comparing the magnitude of values allows discrimination between components.

The value comparison in Figure 3.1 demonstrates that the pump actuator components all have the same minimum and maximum values or 1.0 and 2.0 respectively (though P301 and P602 are notable outliers in regards to the mean value). These characteristics could be indicative of the actuators having discrete values representing on and off, possibly not a perfect square wave due to the electronics design.

Figure 3.1: Component Value Comparison



The chemical sensor components (AIT prefix) have a range of functions relating to water quality such as pH balancing and water hardness measurement. Despite this range of functions and the outliers, it is evident that the range of values are similar (approx 400 times the magnitude of the pump components). In most cases the mean is closely correlated to the minimum value which may indicate the set point of the system (one of the hard-coded rules of the control system). It is evident from the AIT sensor chart that 202 and 501 have minimal variance and so offer little information to the system.

The Coefficient of Variation (CV) is the ratio of the standard deviation to the mean. A low CV indicates that the values are very stable and possibly attributable to non-changing value with a noise signal superimposed.

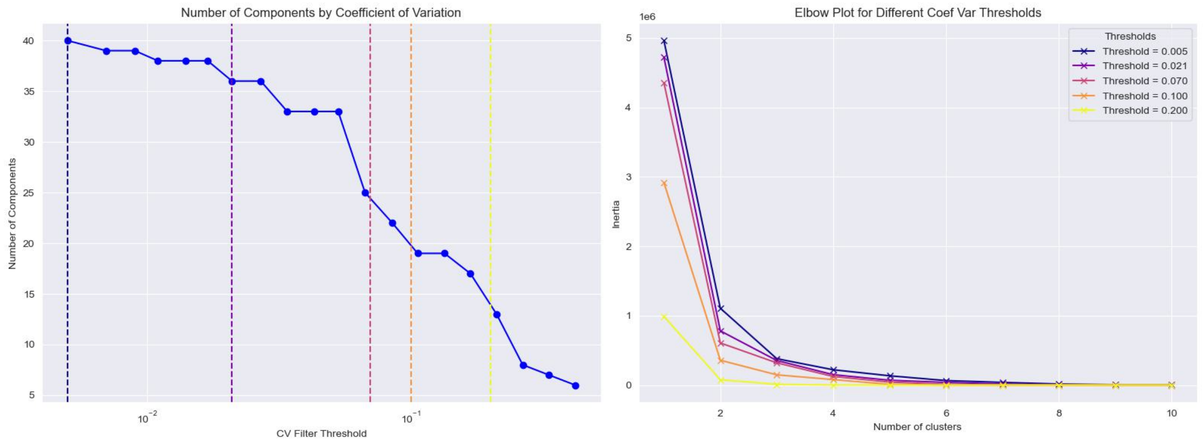
$$\mathrm{CV} = \frac{\sigma}{\mu}$$

The CV was calculated for each component in order to provide a method to filter those which may no influence the other components due to their near static behaviour. Fig 3.2

To evaluate the analysis using rudimentary feature engineering the components were clustered on their Min, Max, Mean and Standard Deviations. As this data is from the raw values the results are (intentionally) skewed by the magnitude of values. An elbow chart of the inertia of the KMeans model suggested between 3 and 6 clusters. Both of these gave a reasonable separation of components into groups but suffered from a single very large cluster representing components which tended to have a small variance. Filtering on the Coefficient of Variation threshold of 0.15 (STD is 15% of the mean or above) allowed values with minimal information content to be removed for the large cluster. Beyond this it impacted the clustering of component which intuitively seemed correct.

Table 3.1 shows some effective clustering using these simple features, cluster 3 correctly distinguishes chemical sensors AIT 201, 203 and 503 from those in cluster 4 (AIT 401, 402, 502 and 504). It can be seen

Figure 3.2: Coefficient of Variation Filtering



in Fig 3.1 that the sensors in cluster 4 have similar characteristics to cluster 3 but a reduced magnitude.

Cluster	Components
Cluster 0	AIT401, AIT402, AIT502, AIT504, PIT501, PIT503
Cluster 1	LIT301, LIT401
Cluster 2	FIT101, MV101, P101, FIT201, MV201, P203, P205, DPIT301, FIT301, MV301 MV302, MV303, MV304, P301, P302, FIT401, P402, UV401, FIT501, FIT502 FIT503, FIT504, P501, PIT502, FIT601, P602
Cluster 3	AIT201, AIT203, AIT503
Cluster 4	LIT101

Table 3.1: Clustered Components

3.4 Component Rate of Change

In order to differentiate control signals from those likely to be sensor signals, the rate of change in the value of the component was calculated. The maximum change seen in any time step was used to calculate how long it would take at this rate to see every value in the data.

$$\text{Change Ratio} = \frac{\text{Component Max Value} - \text{Component Min Value}}{\text{Component Step Size}}$$

This simple formula was highly effective in separating components by their response times and therefore their likely function.

Range	**Tags**
1.0	MV101, P101, MV201, P203, P205, MV301, MV302, MV303, MV304, P301, P302, P602
2.0 - 10.0	AIT504, FIT504, AIT401, PIT502, FIT503, FIT301, FIT201, FIT502, FIT101, FIT601, DPIT301, FIT401, FIT501, AIT201
10.0 - 20.0	AIT501, AIT202, PIT501, PIT503, AIT502
20.0 - 100.0	AIT503, AIT402, AIT203
100.0 - 160.0	LIT101, LIT301, LIT401
NaN	P102, P201, P202, P204, P206, P401, P402, P403, P404, UV401, P501, P502, P601, P603

Table 3.2: Components by Change Ratio

Table 3.2 shows all the motorised valves transition from minimum to maximum value in a single time step, as do the pumps which have a low through-put such as those involved in chemical dosing.

The components which take 2 to 100 seconds are involved in either moving larger volumes of fluid or monitoring chemical properties. These are either subject to larger inertia in the physical bodies or are dependent on the rate of diffusion of chemicals.

The three components in the 100 - 160 second range are water level sensors in the large water tanks.

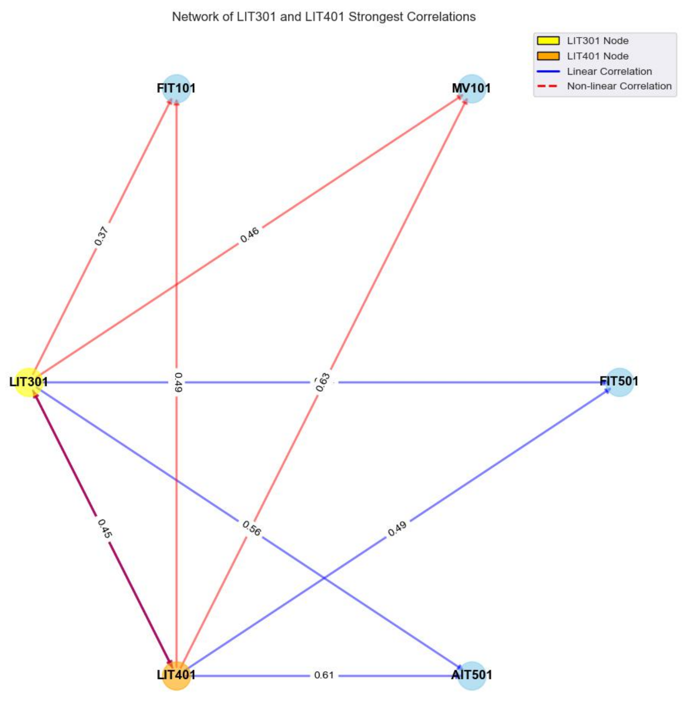
3.5 Component Correlations

The water treatment is a linear process so there will be dependencies within the system (i.e. water can't flow from an empty tank/ a tank being drained will need topping up). Logic dictates that the operation of components within a particular stage will be closely linked though this correlation will take a variety of forms. The switch which activates a pump will be at it's maximum value while the water level with the adjoining tank increases- a positive but non-linear relationship. There will also be inverse correlation between components- as the output tank is drained there will be a positive flow in the pipework.

Linear correlation and non- linear methods were evaluated as the complex interplay of components could lead to non-linear behaviour even with pairwise linear relationships. These methods look for monotonic relationships- the values for two components consistently change together (positive or negative). The Pearson Correlation coefficient measures the linear correlation whereas Spearman's rank correlation measures activity that may or may not be linear.

Fig 3.3 shows the three strongest correlations (linear and non-linear) for tank level sensor components LIT 301 & 401 which were in their own cluster in the previous stage. Both correlation methods return the same components and indicate a strong correlation between the level sensors themselves. The correlation coefficients were calculated using the normalised data (so were not influenced by the magnitude of the values as with the initial clustering) and used the relationship between components rather that the components characteristics as before. Despite this the results generally support the clustering from the previous stage.

Figure 3.3: Component Correlation Coefficient Comparison



P value (statistical significance) value of 5% indicates 38 of 39 components disprove the null hypothesis that the relationship between components is random. This number of related components is achieved by using a Pearson correlation coefficient of 0.01 or above (a very week linear correlation). These Values were calculated from the correlation from the LIT101 water level sensor which is the main water tank at the beginning of the process (and so is understandably linked to all other components to some degree. Restricted the results to correlation coefficients above 0.6 identifies 5 components with strong correlations, all of which are in the same or following stage and are related to the movement of water (pumps or flow sensors). This suggests the P values are not useful is revealing system dependencies.

3.6 Mutual Information

The previous stage indicated that strong, non-linear, relationships are common the system. Mutual information was used to identify which component had behavior could be used to gain information about another component.

To reduce computational load, the Kernel Density Estimation (KDE) was used rather than binned distribution of the components values. KDE adds a chosen distribution (in this case Gaussian) to each data point then sums the values at each point (with the standard deviation/ bandwidth of the distribution providing smoothing and interpolation between measured values). The KDE then has a probability density function which ”for applications that consider continuous variables, KDE provides a more accurate density estimate”[5].

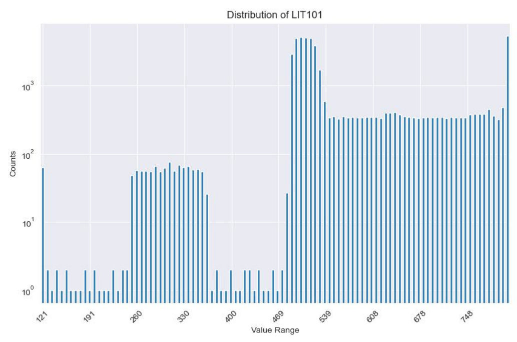


Figure 3.4: Binned LIT101

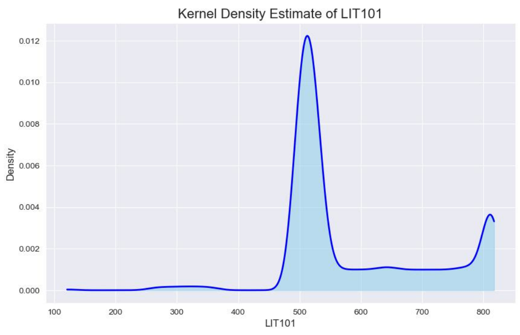


Figure 3.5: KDE Plot for LIT101

The data was scaled otherwise large values would give an artificial impression of strong mutual information. The KDE for all components was calculated separately and these values used as the marginal distributions p(x)p(y) in order to reduce duplication of effort.

$$I(X;Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \, dx \, dy \tag{3.1}$$

Comparison between strongest correlation coefficients and mutual information for water level sensor LIT101 show good agreement. Both Pearson and Spearman correlation coefficients have AIT202 as the strongest correlation. This sensor measures the pH value of the water as it is fed into the water tank in stage 3 (with level sensor LIT301). Presumably the pH value of the water is related to the throughput of the system but this is less useful in understanding the system than FIT504 which is the flow meter at the end of the process (and as the test-bed reuses the water also the feed to the tank monitored by LIT101).

3.7 Assessing System Seasonality by Auto Correlation

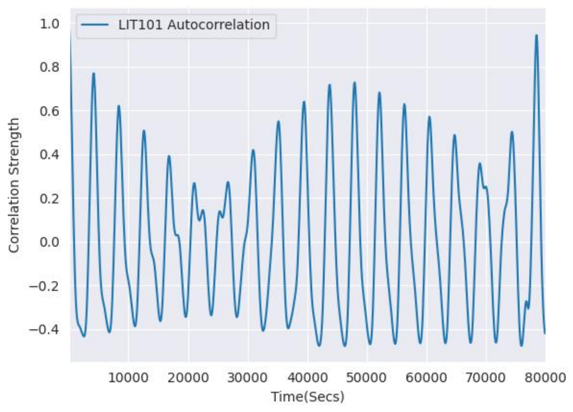
Auto correlating a components value is the process of comparing it to a time-shifted version of itself. The complete dataset was shifted by 100 seconds at a time and the correlation plotted. Fig 3.6 shows very strong correlations at regular intervals of approximately 3000 seconds (hourly) intervals. The regularity of the peaks implies the system simulates demand on an automated schedule. The variation in amplitude corresponds to increased daily demand in morning and early evening. The obvious seasonality would

Mutual Info	Linear Cor	Non Linear Cor
FIT504	AIT202	AIT202
P301	LIT301	LIT301
FIT401	FIT504	FIT504
FIT501	AIT501	AIT501
PIT501	FIT503	FIT503
P402	FIT601	FIT601

Table 3.3: LIT101 Mutual Information and Correlation

allow a single hour window to describe the system operation to a reasonable level of detail. This regular seasonality can be described as the fundamental frequency of the system operation.

Figure 3.6: LIT101 Auto-Correlation Coefficient By Time



3.8 Cross Correlation to Identify Temporal relationships

To approximate these dependencies the covariance between system variables when they were out of phase (advanced or delayed by T seconds) with the other variables. If a stronger covariance value could be found at then it implied a dependency (one may trigger the others).

Using the fundamental system frequency from the auto correlation and phase difference from the Fourier transform it was possible to tailor the correlation parameters to allow fine grain analysis which would be impractical on the full dataset with a small time shift.

Due to the very pronounced auto correlation the small cross correlations were not obvious using graphical methods.

3.9 Fourier Analysis Approach to System interactions

An alternative approach to find system dependencies is the application of Fourier analysis to the system. The treatment plant is designed for constant water supply, as such the processes are cyclical- demand for water from the final tank triggers a cascade of actions which either maintain equilibrium for a constant output or seek to return the system to its ideal state/ set parameters.

Fig 3.6 shows the dependencies in the first stage of the process- the level sensor reading (LIT101) drops as pump P101 removes water. At a fixed set point, flow in the feed pipe begins to re-fill the tank. The FIT101 flow sensor and MV101 mechanical valve signals are exactly in phase.

Fourier Analysis states that any waveform can be reproduced with an infinite number of sine and cosine waves. By filtering these waveform to the fundamental (and some harmonics) it is possible to identify the approximate cycle times in the system. Further more, by comparing the phase of the fundamentals one can see which variables lead or lag the other variables. Again, suggesting the deterministic rules of the system.

Figure 3.7: 1st Stage Component Relationship

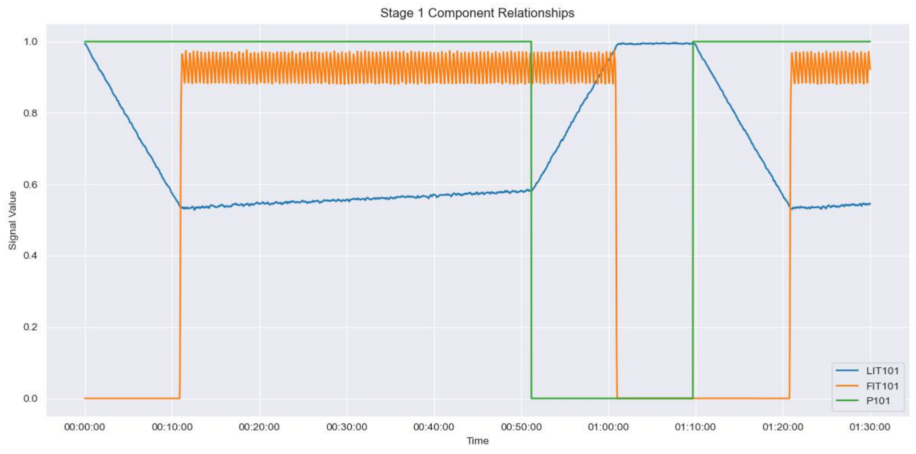


Figure 3.8: Fourier Analysis

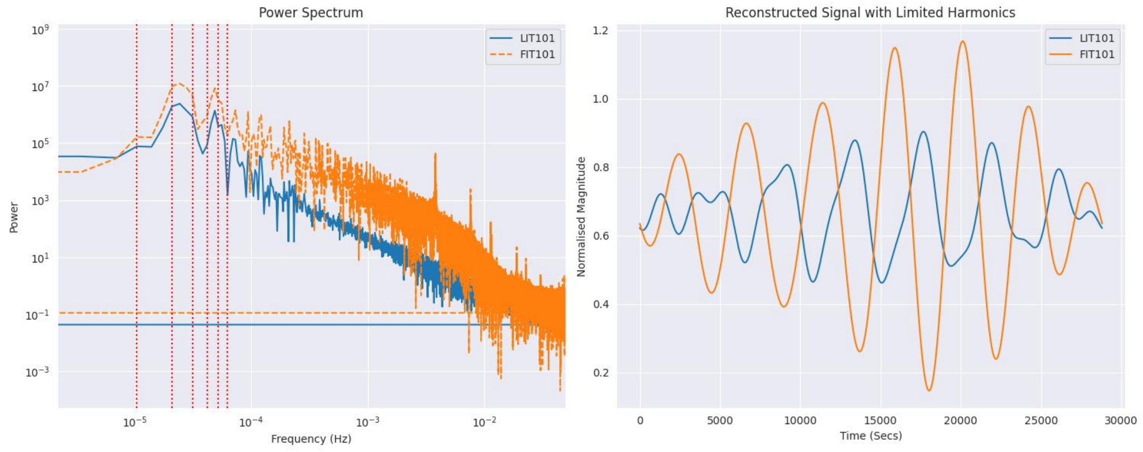


Fig 3.7 Illustrates the similarity in power spectrum- the magnitude of the component sinusoidal waveform which form the complex signal of the component state over time. The horizontal lines are the d.c. components representing the offset of the waveform from zero.

The plot of the reconstructed component waveform uses the first 6 harmonics. For FIT101 these harmonics appear to be in-phase with the fundamental frequency so have the effect of elongating the fundamental sinusoid. LIT101 has harmonics which are out of phase with the fundamental so more complex features are present so some periods have sea=tooth or square wave characteristics. As more harmonics are added the ramp-profile from fig 3.6 would become more evident.

Despite the difference in waveform shape, it is clear that LIT101 and FIT101 are out of phase- maintaining the lagged relationship from fig 3.6. The Fourier transform allows the phases of the fundamental frequencies of each component to be compared with all other components.

Component	Phase Difference	Time Difference
P302	-2.10	-945.17
AIT202	-1.80	-708.81
MV304	-2.54	-147.48
MV301	-0.33	-6.15
P301	2.56	6.24
LIT301	1.47	1013.33
LIT401	2.21	1521.33
FIT101	2.53	1737.01

Table 3.4: Phase and Time Differences for Components

Table 3.3 shows pump P301 which removes water from LIT101 leads the level sensor and the flow in the associated pipe lags by a similar 6 seconds. Also notable is that there is a cascade in the relationship from the level in LIT401, LIT301 and LIT101 as demand up stream (towards the outlet) triggers a response in the dependent components. This illustrates the utility in Fourier transforms for finding temporal relationships in CPS.

The seasonality of the system found via autocorrealtion is approximately 3000 seconds, as such time difference between FIT101 and LIT101 by 1700 could be interpreted as either a lead or a lag.

3.10 Clustering to Identify modes of Operation

The large volume of data obfuscates easily identifiable patterns in the data. By applying clustering algorithms it may be possible to identify distinct phases in the operation of the system.

3.11 Dimensionality Reduction to Identify Influential Variables

The large volume of data likely contains many duplication's of the same system states. By bracketing variable values into a number of states it may be possible to reduces the processing requirements.

3.12 Applying Expert Knowledge- Physics Informed Models

Though the aim of the pipeline is to be able to model a system using only system data, there are knowledge items that it is reasonable for the intelligent Agent to know.

Industrial systems include standard components- level sensors, fluid tanks, motors, switched etc. In addition, the systems will conform to physical laws such volume increase with temperature or pH change when an acidic substance is added. By providing the system with absolute laws and hypotheses in regards to system components it is likely the model can more efficiently describe the system.

This also applies to the types of attack on the system- over I& under value or excessive duty cycle for components.

Several waveform for standard component relationships were generated and these were compared to the relationships identified previously.

3.13 Multivariate Linear Analysis

chain rule with related components form above to find coefficient for each related system + phase difference.

Use this to influence the network diagram generation with sequence dictated by offset and grouping by correlation etc.

3.14 Creating Rules

In the simplest sense the maximum and minimum values for each components but logic tables/ conditional relationships to model common system states. I.e. pump is 0 when tank is full, pump is 1, flow sensor is max when tank ¡= 0.5..... These are then tested in the next stage.

Chapter 4

Passive Assumption Testing

4.1 System Simulation

A variety of models were used to model the system in order to simlulate the interaction the agent would have with the real world system. Assumptions were tested against these models under the assumption that although imperfect, an effective agent would adapt in the same way.

Shaps visualisations were also used on these models to test the effectiveness of the earlier system analysis.

random forest,

multilinear and

4.2 Deep Learning Model

Deep Neural Networks (DNN) are effective on time series data as windows of n seconds can be used with the values from 0 to $n-1$ representing the random variables which are passed to the input layer and the value at n being the dependant variable used as the label. The window is then shifted by one second to create a new set of training data. In the multivariate SWaT data a window is an array $n-1$ (seconds of training data in the widow) by d, the number of components in the dataset (37 in the cleaned data). This array is flattened to produce a one dimensional array $(n-1)*d$ in length for the input layer of the neural network.

The output layer of the DNN correspond to the number of components so is a 1 dimensional array of length d (37). The predicted values are compared to the label (final value in the data window and the overall loss calculated using the mean of squared error between prediction and label.

The weights between layers in the model are initialised using random values then modified based on the loss function (back propagation). The learning rate of the model dictates the step size when updating the model rates and is optimised to ensure the model both finds a solution in a reasonable time but does not take too large steps and overshoot the optimum values.

The DNNs were trained using nVidia GPUs with the CUDA toolkit which allows efficient, parallel processing of the models. As such, rather than a single input layer of $(n-1)*d$ the input was stacked in batches which were processed simultaneously. The Adam optimiser in TensorFlow is based on the Stochastic Gradient Descent (SGD) method. SGD updates the model weights after each individual training record (rather than at the of the whole dataset as with conventional gradient descent). The Adam model updates at each batch of parallel processed records.

A number of DNN architectures were evaluated with the optimum hyper parameters(learning rate, neurons in the dense layers, processing batch size and LSTM units etc.) for each found by iteratively testing each using the Keras Tuner tool.

The dataset is deterministic and even the most simple model of a single dense layer produce reasonable predictions. The most effective model tested used a single LSTM layer and dense layer. The bi-directional version of this model performed slightly worse even though many components in the dataset have strong dependencies.

Evaluation of the model is complex as the aim was to produce a model which incorporated all system values. Many components have discrete values which are not handled well by the regression process. As such the Mean Absolute Error (MAE) used to evaluate the models performance against the validation dataset was 0.18 or an average 18% error in each prediction.

Figure 4.1: Deep Neural Network Performance

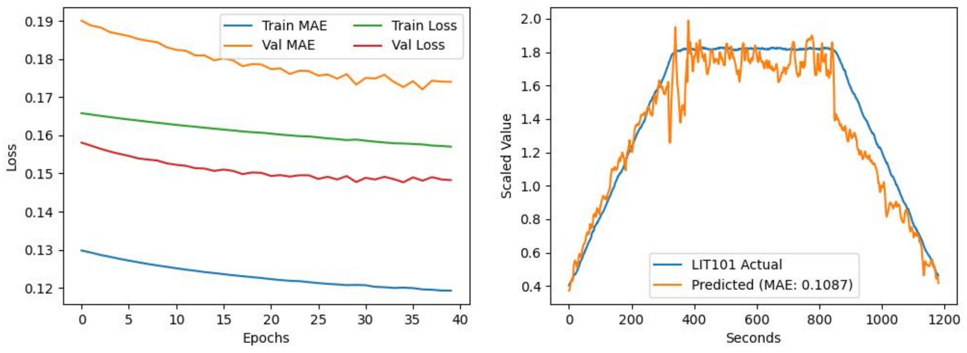


Fig... shows the model loss being better on the validation set over the test set, this is likely due to the data being split at a single time point rather that the windows being randomly selected. The jagged lines for the validation MAE and loss indicate noise in the model due to the choice of learning rate or batch size.

A 20 minute slice (1200 rows) of the data set were run through the model in order ot produce a predicted version of the same window. The plot of actual values for water level sensor LIT101 verses the predicted value illustrates the noise in the model but that it is none the less effective in characterising the general behaviour.

The training dataset included all components, many of which are highly correlated. Further development of the DNN could improve it's performance but this is likely to be limited.

and offers one prediction based on the n

LSTM.

4.3 Testing Phase Relationships

The Fourier analysis identified components with similar activity patterns and the phase difference between the these fundamental frequencies. By performing a time shift of data which is beleived to either lead or lag another component there should be a stronger correlation in the data. Using LIT101 FIT1101 is can be seen from the fig,,, that the level indicator lags behind the flow in the pipe feeding the tank.

4.4 Learning Rate

Learn this by testing agianst the GAN.... assume system fully captured in normal state. Is new activity an alarm? Learn the thresholds for this (gradient based attack??)

4.5 GAN and other anomaly detectors

Train a gan on the attack data and then have running at the same time to tyr and spot the probing of the intelligent agent.

4.6 Testing Phase Relationships

The Fourier analysis identified components with similar activity patterns and the phase difference between the these fundamental frequencies. By performing a time shift of data which is beleived to either lead or lag another component there should be a stronger correlation in the data. Using LIT101 FIT1101 is can be seen from the fig,,, that the level indicator lags behind the flow in the pipe feeding the tank.

4.7 Apply Methods to Generate System Diagram

Combine the regression, phase relationships and other factors from above to create a model which generates a visual representation of the system process- including operating parameters and step sizes.

Use Monte Carlo sampling in chronological order then Stochastic gradient descent to calculate coefficents, work out learning rate for best performance with smapling rate.

Use Nyquist Shannon for sampling rate of Monte Carlo

-
-
-
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Chapter 5

AI Solution

5.1 AI Agent Approach

An intelligent agent views it's world through percepts and affects it through it's choice of actions. "The agent function maps given percept sequence to an action" (Norvig, 2022) and is an abstract mathematical model of the agents behaviors. The actions are decided by the agent program.

It would be possible to produce attack data through a deep learning, neural network such as Generative Adversarial Network (GAN). A GAN consists of a classifier trained on to distinguish inputs from the attack dataset and the normal operation dataset. This 'discriminator' is used to provide feedback to another neural network (the generator) which initially generates random data but learns to generate approximations of real data.

A GAN would rely on the the data from previous, human designed attacks and would not be able to choose the most appropriate attack for the current system condition.

The aim of the reinforcement learning based agent is to start with an understanding of the system which was learnt passively, refine this knowledge by probing it (information gathering via packet injection) then use it's inbuilt knowledge of attack types to choose how to manipulate the system.

5.2 Task Environment

The task environment is defined by the Performance, Environment, Actuators and Sensors description (PEAS).

It is assumed that the task environment is partially observable due to the insider level of access but demands on the system (clean water output) are random so the agent must account for this stochastic behaviour.

.........unknown in regards to the behaviour and relationship of component. Physical laws will be included.

The system is deterministic when there is no demand- for example when returning the system to it's desired set points following a high demand period, but stochastic otherwise. The component processes are sequential and dynamic and are continuous-state and continuous time. The SWaT is a competitive, multi-agent environment because the control system is trying to maintain a system state the attack agent is trying to disrupt.

The system is assumed to be Markovian as the current state fully describes the system- it is' not dependent on the previous states.

5.3 Agent Heuristics

Industrial systems share common components and common failure modes, this knowledge will be incorporated as heuristics. The agent will try to map observed behavior to a known component type then prepare an attack against it.

An example would be identifying a water level sensor and the switch, pump and valve responsible for maintaining it's level. An attack could try to manipulate the associated devices in order move the level beyond it's highest known value. Reaction to this attack should also be recorded and incorporated into the model.

These heuristics are passed to the model from the component object attributes generated in the passive stage and form the basis for the agents actions and rewards. The rewards to use a component identified as being influential will prioritise it's use in the exploration stages.

5.4 Agent Policy

The agent will make rational decisions based on it's knowledge of the system, if the consequence of this decision (reward) supports its understanding then this is a positive performance measure and the policy table is updated accordingly. The aim of the agent will be to reach then surpass the maximum and minimum previously observed values for the component without triggering a response from the anomally detector.

To make the agent more appropriate a penalty is given for the amount of noise on the system- the number of packets required to attain the effect, a packet is required for each component at each time step so using the most influential components only is prioritised. A noisy attack is more likely to be spotted by anomaly detectors in a real world system.

The aim is that the agent is not reliant on it's prior knowledge so small rewards are given for using the heuristics but these are outweighed by the other rewards such as reaching the terminal state (the maximum previously observed value). This ensures the agent has the autonomy to explore the environment in the earlier episodes and find the optimum Q table rather than use erroneous assumptions from the passive stage.

The agent will alter certain parameters (take actions) based on the current state in order to put the CPS into the desired state- whether this is to test it's beliefs or instigate an attack. Though a deterministic system, in the real world there is a random element which is the demands on the system by the user (requirement for clean water). The agent must work with this uncertainly so the system will be considered as stochastic. The Markov Decision Process is used by the agent to account for this.

?? Initially the system will be considered as being in a dormant state- no water demand so state changes will be based on the PLC attempting to retain the system rules. A more advanced agent would work while the system is in use by simulating a demand on the output stage. ??

The problem is a sequential problem as the agent must achieve it's aim using steps sizes within normal parameters so as not to trigger the anomaly detector. The initial beliefs indicate which components have an affect on the desired out come. The agent must produce actions for each of these in order to reach it's goals and confirm whether the beliefs are accurate/ need updating.

The agent has two distinct aims- learn the system and attack the system. These can be considered the exploration and execution stages. The aim of the exploration stage is to keep the system within parameters as much as possible (save testing the limits of each component). The execution stage will attempt to attack a component and devise a policy(??) to achieve this. The rewards are based based on how well the agent predicts the impact of it's action.

The utility function (the effectiveness of its plan to alter the system state) is dependent on a sequence of states and actions known as an environment history [6]. This utility will be the sum of reward for each step in the sequence.

The agent will maximise this utility by creating a policy π which accounts for all system states. The environment history may be different even when starting from the same initial state due influence of outside factors. The policy accounts for this and allows the agent to find the optimum way to move the system into the attack state (without triggering the anomaly detector).

The CPS is is considered as having an infinite horizon as it's operation is continuous and there is no time limitation in the application of an attack- the goal is to reach the desired terminal state (desired component value) whilst avoiding detection. This means the optimum policy is stationary so does not have to account for time, just the system state. If the CPS exhibited seasonality then a non-stationary policy may be more effective.

5.5 LIT101 Agent Example

The Q learning model for the Stage 1 water level *LIT101* is a mapping of an action to every possible *LIT101* state. As this is a continuous value it is binned according to the step size Δ which is the maximum change observed during a single time step t in the dataset. The upper $LIT101_{max}$ and lower $LIT101_{min}$ limits of this range are taken from the dataset and stored as attributes in the Component object along with the step size. This range forms the observation or state space $\mathcal{S}$ that describes every possible state for *LIT101.*

$$\mathcal{S} = \{\mathrm{LIT101_{min}} + i \cdot \Delta \,|\, i \in \mathbb{Z}, \, \mathrm{LIT101_{min}} + i \cdot \Delta \leq \mathrm{LIT101_{max}}\} \tag{5.1}$$

The $LIT101_{max}$ and lower $LIT101_{min}$ are assumed to represent the upper and lower limits of the system, these are used as the terminal states of for the agent as a single step beyond either would constitute a successful attack.

The passive analysis of $LIT101$ indicated that inlet valve $MV101$ and outlet pump $P101$ were suitable components for the manipulation of the water level. These components had binary behaviours of off and on (though with a D.C. offset in the off state) so the action space has four elements:

State	$MV101$ (Inlet Valve)	$P101$ (Outlet Pump)
1	Closed	Closed
2	Closed	Open
3	Open	Closed
4	Open	Open

Table 5.1: Action space with four possible states for $MV101$ and $P101$

A *Q Table* is a lookup table where the rows are the values from the state space $\mathcal{S}$ and the columns each possible action- four in this case. Each cell of the table contains a Q value, which estimates the expected reward for taking a the action in the specified state.

During the training or exploration phase the table is initialised with 0 reward in all cells then agent randomly selects an action for the current state and a reward is calculated. This is the /emphtransition which is described by the tuple (s, a, r, s'). As the exploitation progresses the value in each cell is populated so that for any given state s the best action can be chosen by finding the maximum Q value, which represents the reward at the next state s' if that action is taken.

$$V(s) = \max_a V(s')$$

In order to prioritise the most efficient route a negative reward is given for each action $R(s, a)$, this has the effect of punishing the agent if it, for example, chose to manipulate a component which had less impact than another as the smaller reward gain may be negated by the cost of the actions[4].

$$V(s) = \max_a (R(s, a), V(s'))$$

With $\mathrm{LIT101_{min}}$ = ..., $\mathrm{LIT101_{min}}$ and step size , Δ = ... the following is the *Q table* for the middle of the state space:

States		Actions		
	$MV101$ Closed $P101$ Closed	$MV101$ Closed $P101$ Open	$MV101$ Open $P101$ Closed	$MV101$ Open $P101$ Open
		1	Off	Off
		2	Off	On
		3	On	Off
		4	On	On

Table 5.2: Q Tables for $LIT101$

As *LIT101* had two terminal states or goals and a set of linear values in the state space so there are always two possible routes that will achieve the goal. In this simple agent the negative reward above would suffice to prioritise the optimum route but as the agent complexity is increased, to work for terminal states in other components, a discount factor gamma: γ becomes necessary.

The discount factor is analogous to inflation in the economy, given an equal reward for attaining either of the *LIT101* goals the one which was achievable sooner would offer the higher return.

5.6 LIT101 Agent Tuning

The initial agent took a window of the dataset and took an action on the basis of the LIT101 value in the final, most recent row of the data. The modified state containing the action was passed to the LSTM along with enough of the original window to fully fill the input layer.

This was repeated for the rest of the trial so that when the number of episodes in the trials passed the number or rows in the data window there was no longer any of the real SWaT data in the feed to the LSTM.

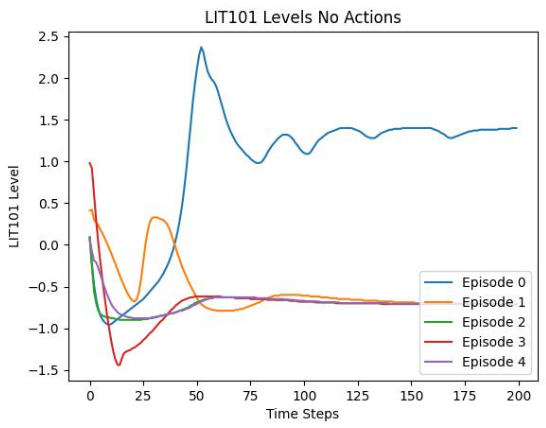


Figure 5.1: LIT101 with No Actions

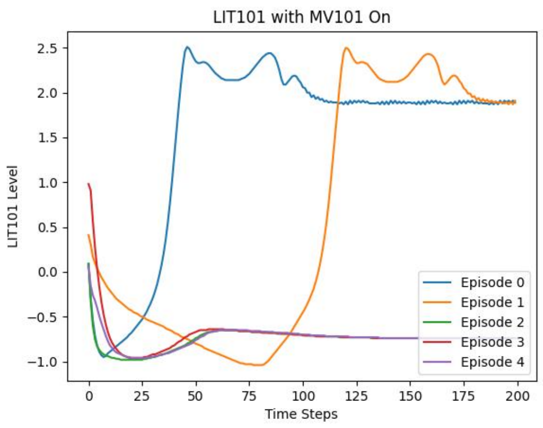


Figure 5.2: LIT101 with MV101 Held High

Fig 5.1 and 5.2 show the effect of the LSTM generating it's own data window without any input from the agent and when the agent turns the input pump MV101 on for the whole trial. The agent was passed 5 data windows randomly chosen from the dataset, each with a different final state of LIT101. There were 100 trials in the episode so from Time Step 100 the data is fully synthesised. It is evident that the model is very noisy both with and without an action from the agent. Fig 5.2 shows the value range when the input pump MV101 is on, though two of the trials are successful the overall behaviour is similar to that in Fig 5.1. Without any actions set the model re-configures it's own inputs at each step, this results in violent transitions until the majority of the data in the window is synthetic. The LSTM has used all system states to create a model though in the real system only a subset of components has the ability to influence the behaviour (mainly the motorised valves and pumps). As such the model may change parameters which are not possible in the real system. By increasing the action space to include more control components the the degrees of freedom for the LSTM are reduced as these parameters will be set by the agent. Though the parameter itself may not influence the target value it will be accounted for in the policy as improving system stability will reduce the number of steps to reach the goal.

5.7 LIT101 Q Table

Fortunately the Q table is built by randomly selecting system states, taking an action then recording the reward. As such it does not require consecutive system values based on the previous actions taken. This allows the agent to take a random window from the real SWaT data, append a copy of the final state with the control component states modified as necessary then pass this to the LSTM. The reward will be calculated from the s' state returned then the following episode in the trail will select another random window from the dataset and repeat. This means the model will perform like that illustrated in fig 4.1 where noise is not allowed to influence the general trajectory of the target value.

The reward was calculated on the maximum value inorder to simplyfy the evaluation.

$$\text{Reward} = \Delta LIT101 \tag{5.2}$$

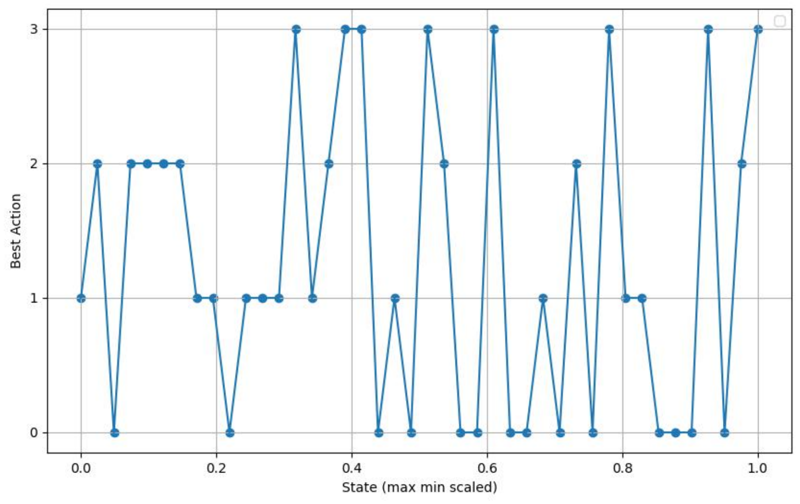
Where

$$\Delta LIT101 = \text{new\_state}[LIT101] - \text{current\_state}[LIT101] \tag{5.3}$$

The Q learning model was run for 10000 episodes (a single action in each) and he plot of the best action at each state (normalised value of LIT101) is shown in fig 5.3 against the normalise value range (state). There is no obvious convergence towards the optimum action (action 1) which indicates the model is too simplistic.

Discounting the binary components, if the full state space of 37 components were each digitised into 100 bins the model would be 10^{74} which is comparable to estimated number of atoms in the known universe.

Figure 5.3: Basic Q Table Performance



Discounting the binary components, if the full state space of 37 components were each digitised into 100 bins the model would be 10^{74} which is comparable to estimated number of atoms in the known universe.

5.8 Advanced Agent Example

MV101, P101, MV201, P203, P205, MV301, MV302, MV303, MV304, P301, P302, P602

Performance Sensors	Environment	Actuators
Attack Types for real world effects Findout which can be spoofed to cause effect	Intrusion detection	Find out which have ef
False alarms Accelerated wear Over level/ pressure/ temp etc.	Scada counteracting actions	

Table 5.3: Task Environment

5.9 Test Model Against Simulated SWaT

5.10 Learning Rate

Learn this by testing agianst the GAN.... assume system fully captured in normal state. Is new activity an alarm? Learn the thresholds for this (gradient based attack??)

5.11 Deep Q Learning

The Q table

-
-
-
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Chapter 6

Analysis

6.1 Pipeline

The work in the previous chapters considered various options for learning deterministice, time series data such as that of a CPS.

The outcome of this is to identify a pipeline which can be applied to an unkonwn system and not only effectively model the system but autonomously manipulate it to a level which would constitute a cyber attack.

The methods investigated overlap each other to some degree so an optimismed pipeline would look like that in fig ....

6.2 Pllication to unkonwn System

A crude model of a CPS was created to test the pipeline against an unknown system. The block diagram for the CPOS and teh pipelines approximation show good......

6.3 next

Chapter 7

Future Work

7.1 Live Testing

Try on live system which includes random demand on the output though maybe the system already includes this??

7.2 Real World Testing

Try on ral system (possibly do basic minicps demo to show effectiveness of a single stage

7.3 Wavelet Analysis

next stage form fourier

The miniCPS was evaluated as a way of simulating the communications packets but this was judged to be outside the scope of this project

7.4 next

-
-
-
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Bibliography

[1] iTrust Centre in Cyber Security. Introduction to swat testbed, 2016. Accessed: 5 April 2016.

[2] Yifan Jiaa and J. W. C. M. P. S. C. J. S. Y. C. Adversarial attacks and mitigation for anomaly detectors of cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 2021.

[3] Shaymaa Mamdouh Khalil and H. B. H. O. D. T. K. K. L. V. K. Threat modeling of cyber-physical systems - a case study of a microgrid system. *Science Direct*, January 2023. Online; Accessed: [Your Access Date here].

[4] Serrano l. Serrano academy - a friendly introduction to deep reinforcement learning, q-networks and policy gradients, 2021. Online; Accessed: Sept 2024.

[5] Robert J May, Graeme C Dandy, Holger R Maier, and T M K Gayani Fernando. Critical values of a kernel density-based mutual information estimator. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):543–558, 2011.

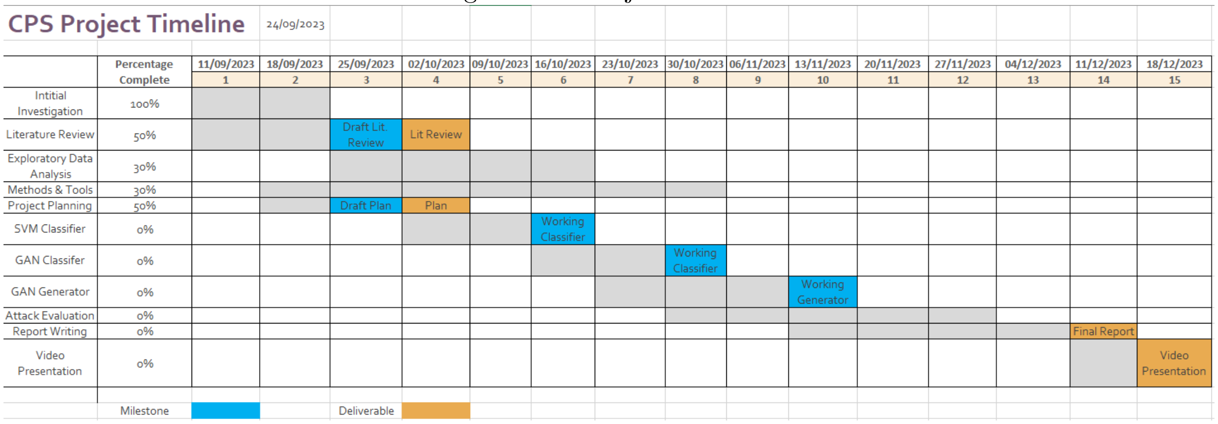
[6] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson, 4th edition, 2020.

Appendix A

Project Timeline

Proposed breakdown of project elements by week:

Figure A.1: Project Timeline



CPS Project Timeline 24/09/2023

	Percentage Complete	11/09/2023 1	18/09/2023 2	25/09/2023 3	02/10/2023 4	09/10/2023 5	16/10/2023 6	23/10/2023 7	30/10/2023 8	06/11/2023 9	13/11/2023 10	20/11/2023 11	27/11/2023 12	04/12/2023 13	11/12/2023 14	18/12/2023 15
Intitial Investigation	100%	▨	▨													
Literature Review	50%	▨	▨	Draft Lit. Review	Lit Review											
Exploratory Data Analysis	30%			▨	▨	▨	▨									
Methods & Tools	30%		▨	▨	▨	▨	▨	▨	▨							
Project Planning	50%		▨	Draft Plan	Plan											
SVM Classifier	0%				▨	▨	Working Classifier									
GAN Classifer	0%						▨	▨	Working Classifier							
GAN Generator	0%							▨	▨	▨	Working Generator					
Attack Evaluation	0%								▨	▨	▨	▨	▨			
Report Writing	0%										▨	▨	▨	▨	Final Report	
Video Presentation	0%														▨	Video Presentation

Milestone	🟦		Deliverable	🟧	

Defining System Parameters

Developing initial understanding of relationship between components of water treatment system including their normal operating paramenters.

- The data
- Real-Time Modelling of the SWaT was performed in a VMWare Workstation Player virtual machine running Ubuntu V22.04.03. The Mininet network simulation tool with a custom MiniCPS SWaT model. Wireshark was used to analyse network traffic.
- I used the *Pandas* and *Seaborn* public-domian Python Libraries.
- I used LaTeX to format my thesis, via the online service *Overleaf*.