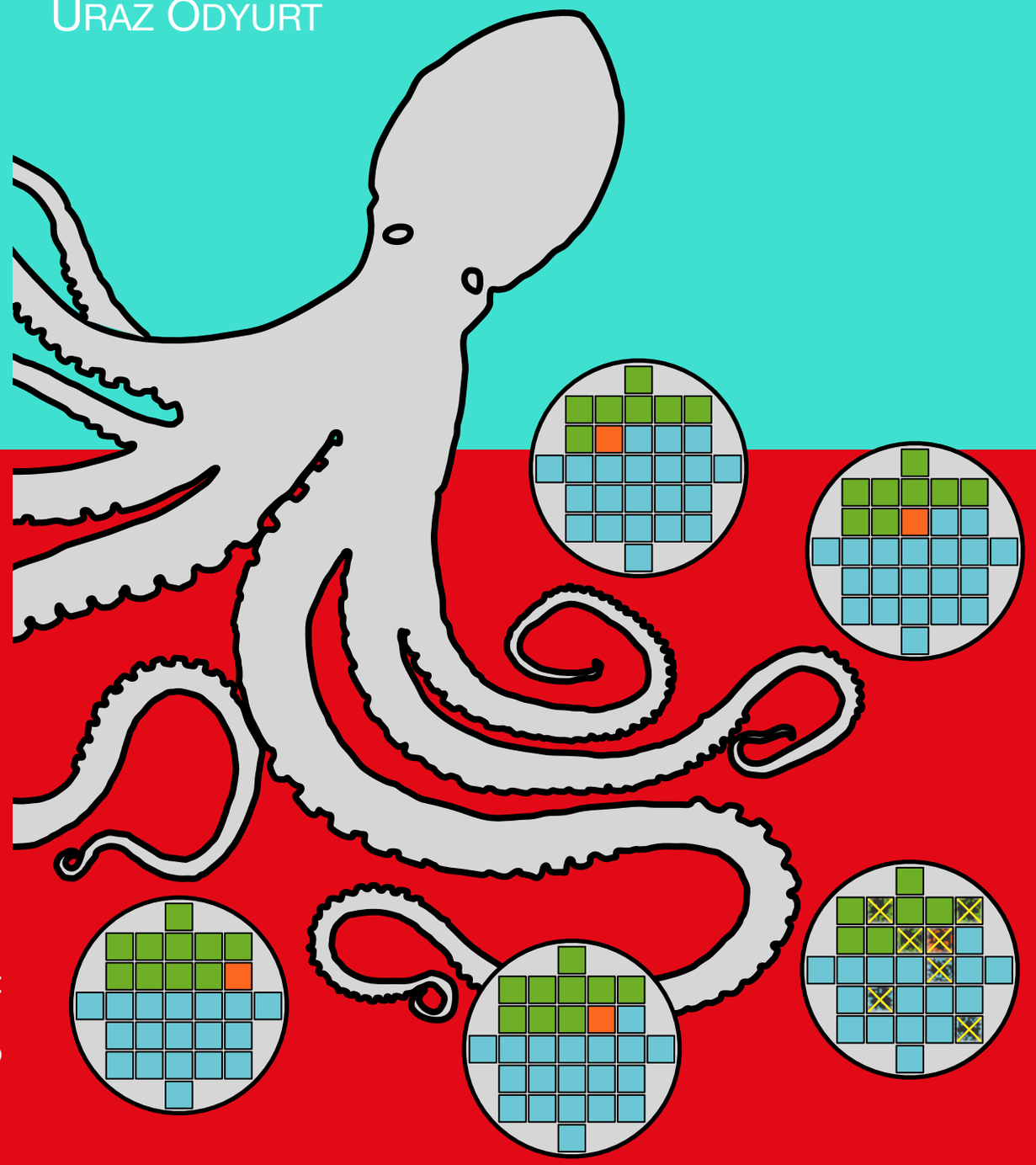
This thesis is based on the author's research work and publications from 2017 till 2021, as part of the Parallel Computing Systems (PCS) research group at the Informatics Institute (IvI) of the University of Amsterdam.
The research work has been carried out within the requirements and goals of the iDAPT project and in close collaboration with ASML Nederlands B.V., under the supervision of prof. dr. A.D. Pimentel and prof. dr. ir. C.T.A.M. de Laat.

A DATA-CENTRIC APPROACH TOWARDS IDENTIFICATION AND PREDICTION OF ANOMALIES IN INDUSTRIAL CPS URAZ ODYURT

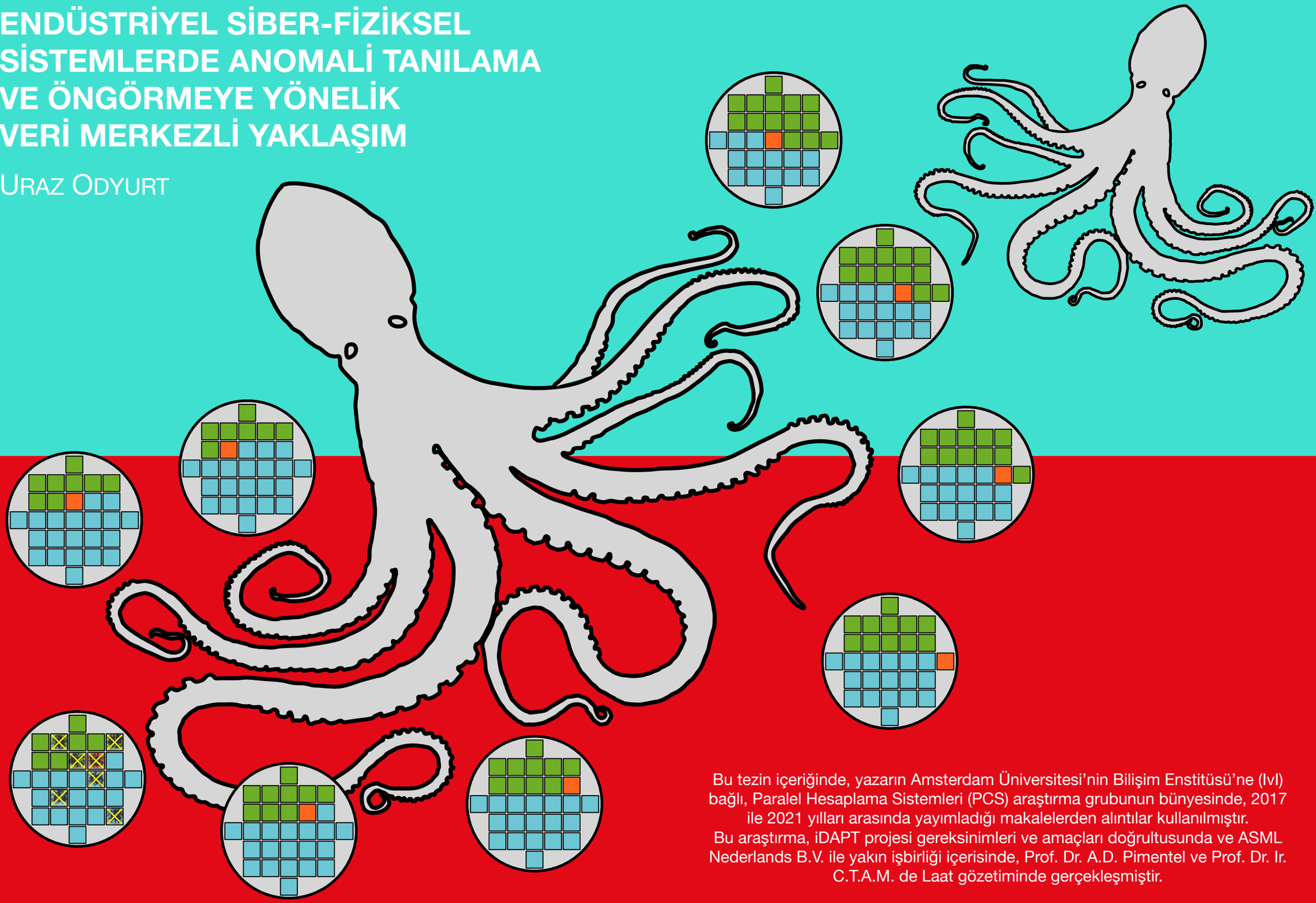
A DATA-CENTRIC APPROACH TOWARDS IDENTIFICATION AND PREDICTION OF ANOMALIES IN INDUSTRIAL CYBER-PHYSICAL SYSTEMS

URAZ ODYURT



ENDÜSTRİYEL SİBER-FİZİKSEL SİSTEMLERDE ANOMALİ TANILAMA VE ÖNGÖRMEYE YÖNELİK VERİ MERKEZLİ YAKLAŞIM

URAZ ODYURT



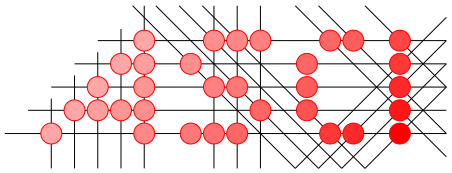
Bu tezin içeriğinde, yazarın Amsterdam Üniversitesi'nin Bilişim Enstitüsü'ne (IvI) bağlı, Paralel Hesaplama Sistemleri (PCS) araştırma grubunun bünyesinde, 2017 ile 2021 yılları arasında yayımladığı makalelerden alıntılar kullanılmıştır.
Bu araştırma, iDAPT projesi gereksinimleri ve amaçları doğrultusunda ve ASML Nederlands B.V. ile yakın işbirliği içerisinde, Prof. Dr. A.D. Pimentel ve Prof. Dr. Ir. C.T.A.M. de Laat gözetiminde gerçekleşmiştir.

A DATA-CENTRIC APPROACH TOWARDS IDENTIFICATION AND PREDICTION OF ANOMALIES IN INDUSTRIAL CYBER-PHYSICAL SYSTEMS

Uraz Odyurt



This work was financially supported by *Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO)*, a.k.a., the Dutch Research Council, under the project number 14208, and was performed in close collaboration with *ASML Nederlands B.V.*, as its main user.



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 428.

Copyright © 2021 Uraz Odyurt.
All rights reserved.

Cover design: Uraz Odyurt
Thesis template: `classicthesis` by André Miede & Ivo Pletikosić
Printed and bound at Ipskamp Printing, Enschede, The Netherlands.

ISBN-13: 978-94-6421-520-5

A DATA-CENTRIC APPROACH TOWARDS IDENTIFICATION AND PREDICTION OF ANOMALIES IN INDUSTRIAL CYBER-PHYSICAL SYSTEMS

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op maandag 6 december 2021, te 14.00 uur

door

Uraz Odyurt

geboren te Istanbul

Promotiecommissie

Promotores:	prof. dr. A.D. Pimentel	Universiteit van Amsterdam
	prof. dr. ir. C.T.A.M. de Laat	Universiteit van Amsterdam
Overige leden:	prof. dr. S.J. Altmeyer	University of Augsburg
	prof. dr. P.T. Groth	Universiteit van Amsterdam
	prof. dr. S.B. Scholz	Radboud Universiteit Nijmegen
	prof. dr. K.B. Åkesson	Universiteit van Amsterdam
	dr. P. Grosso	Universiteit van Amsterdam
	dr. ir. A.L. Varbanescu	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Kostir ro betri heldr en at klǫkkva sé,
hveim er fúss er fara;
einu dægri mér var aldr of skapaðr,
ok allt líf of lagit . . .

Boldness is better than plaints can be,
for him whose feet must fare;
to a destined day has mine age been doomed,
and my life's span thereto laid . . .

— Skírnismál

ACKNOWLEDGEMENTS

Some say the best part of a journey is its ending, when the destination is reached and the goal is achieved. Others say it is the journey itself and what happens during. I say the best deed is the first deed, when the first step is taken and the chaos begins, setting the wheels of time, curiosity, experience and achievement in motion. I would like to thank my promoter and supervisor, prof. dr. Andy D. Pimentel, for doing the best deed by hiring me as a PhD candidate in the iDAPT project and setting the wheels of time, curiosity, experience and achievement in motion for me. I do remember him saying, "you have your own way of doing things and that is why I hired you," which sums up his patience and how he allowed me to develop my understanding and my skills in this field. I also would like to thank prof. dr. ir. Cees de Laat for accepting to be my second promoter and for advising me during our meetings, in both technical and non-technical matters. I remember him saying once, "you are being paid to do extraordinary research, not to fulfil some project requirements," which had a profound impact on my view towards academic work.

Throughout the four years that we have been working on the iDAPT project, I have had the privilege of collaborating with colleagues within the project iDAPT, within the Systems and Networking Laboratory (SNE) and within the Parallel Computing Systems (PCS) research group. I have learned from each and everyone of them and I would like to thank them here, dr. Hugo Meyer, Simon Polstra, Julius Roeder, dr. Benjamin Rouxel, Dolly Sapra, dr. Jun Xiao and many others. I would like to send my regards to Jaap van Ginkel and Karst Koymans for trusting me with the teaching assistance and student project supervision duties for the Security and Network Engineering Master programme and for allowing me to stay involved with the cybersecurity domain. I also would like to mention my colleagues from the Security and Network Engineering Master programme and send my regards to dr. Arno Bakker and Peter Prjevara.

The people at ASML Nederlands B.V., as the main industrial collaborator of our project, had a big role in this achievement. Amongst all of our colleagues at ASML, I would like to thank dr. Ignacio Gonzalez Alonso, dr. Evangelos Paradas, dr. Ramon Schiffelers, Robert van Uden and Erwin Schoonmakers (from Alten Nederland) for helping someone from academia to grasp the differences and the realities of the industry.

I would also like to thank the members of the user committee for the project iDAPT throughout these years, dr. Ignacio Gonzalez Alonso, Theo Engelen, René van Hees, Henk Kant, dr. ir. Bram van der Sanden,

prof. dr. Sven-Bodo Scholz, prof. dr. ir. Jeroen Voeten and ir. Rogier Wester, for their commentary and much needed feedback, bringing in diverse perspectives to the table.

Last but definitely not least, I would like to extend my appreciation to the members of the examination committee, prof. dr. Benny Åkesson, prof. dr. Sebastian Altmeyer, dr. Paola Grosso, prof. dr. Paul Groth, prof. dr. Sven-Bodo Scholz and dr. ir. Ana-Lucia Varbanescu, for putting in the time and the effort to read this thesis. Their feedback has been invaluable for me as an inexperienced researcher.

CONTENTS

1 INTRODUCTION 1
- 1.1 Evolution of computing machinery 1
- 1.2 Data-centric solutions 4
- 1.3 Role of artificial intelligence 6
- 1.4 Cyber-physical system complexity spectrum 7
- 1.5 Synopsis 9
- 1.6 Research questions 9
- 1.7 Outcomes of the research 11
- 1.8 Author's publications and thesis structure 12

2 FUNDAMENTAL CONCEPTS 15
- 2.1 Extra-Functional Behaviour 16
- 2.2 Execution phases 16
- 2.3 Anomalies and faults 18
- 2.4 Behavioural signatures and passports 19
- 2.5 Time series data classification algorithms 22

3 METHODOLOGY 23
- 3.1 Performance anomaly identification 23
- 3.2 Behaviour improvements 25
- 3.3 Information position 26
  - 3.3.1 White box position 26
  - 3.3.2 Grey box position 28
  - 3.3.3 Black box position 28
- 3.4 Communication-centric monitoring, modelling and simulation 29
- 3.5 Digital twins and their usage 30
  - 3.5.1 Digital twin for a communication subsystem 31
  - 3.5.2 Hiccup injection 35
- 3.6 One challenge, two approaches 35
- 3.7 Use-cases and workloads 36

4 EXPERIMENTAL IMPLEMENTATION 39
- 4.1 Software passports workflow 40
  - 4.1.1 Effective sensing 40
  - 4.1.2 Digital twin and behavioural coverage 43
  - 4.1.3 Feature set and classification 47
- 4.2 Power passports workflow 48
  - 4.2.1 Mean passports 49
  - 4.2.2 Feature set and classification 51
- 4.3 Experimental set-up 53
  - 4.3.1 Semiconductor photolithography machine: Synthesising anomalies 55
  - 4.3.2 Image analysis platform: Diversifying scenarios 57

4.4 Alternative identification approach 58
4.4.1 Data set 59
4.4.2 Convolutional neural network structure and search space 61
5 CLASSIFICATION RESULTS 65
5.1 Semiconductor photolithography machine 66
5.2 Image analysis platform 69
5.3 Classic ML vs Advanced DL 69
5.3.1 Quantitative comparison 72
5.3.2 Qualitative comparison 74
5.4 Discussion 75
5.4.1 Explainable output 79
5.4.2 Limitations of the study 80
5.4.3 Implications for industrial deployment 80
6 RELATED WORK 85
6.1 Classification algorithms 86
6.2 Interest in deep learning 87
6.3 Electrical metrics 88
6.4 Power of regression modelling 89
6.5 Modelling and simulation 89
7 CONCLUSION 91
7.1 Looking back at research questions 93
7.2 Future extensions 94
7.3 Final thoughts 95

BIBLIOGRAPHY 99
PUBLICATIONS 109
SUMMARY 111
SAMENVATTING 113

LIST OF FIGURES

Figure 1.1 The evolution of mobile phones from their commercial release until the year 2014 is shown. (Image source Wikimedia, public) 1

Figure 1.2 Plotting the progression of software size in terms of non-comment source code lines per every major space mission. Note that how human missions have larger codebase compared to robotic missions, pointing out the increase in software size when dealing with life-threatening scenarios. This plot is generated based on the data from [18, 34]. The plot also shows the amount of code in modern cars to provide a perspective on how fast CPS software is growing. 3

Figure 1.3 Depicted is the high-level view towards performance anomaly detection, identification and the envisioned countering of their effects. The focus of this thesis is highlighted in yellow, with a partial highlight on the topic of digital twin, for its implications on actuation is out of the scope of this thesis. 6

Figure 1.4 A state-of-the-art semiconductor photolithography machine is shown as an example complex industrial CPS, running numerous computing nodes with heterogeneous architectures and interacting with the physical domain. (Image courtesy of ASML Nederlands B.V.) 8

Figure 1.5 Depicted is a brief view of the order of tasks included in a wafer processing workflow, executed by a semiconductor photolithography machine. Different levels of repetition of tasks are observable, i.e., at the die exposure level, where the wafer is stepped through and exposed one die at a time inside the stepper unit (blue repetition area), at the wafer batch processing level, where a number of wafers are exposed using the same reticle (green repetition area), and at the reticle exchange level, where a new reticle is selected for the processing of an upcoming batch of wafers. 10

Figure 1.6 Shown is an overview of the topic-wise flow of chapters within this thesis, with solid arrows representing direct transitions and dashed arrows indicating connection of the content where there is no direct transition present, with black and blue dashed arrows indicating usage of fundamental concepts and collection of material for the conclusion from other chapters, respectively. Chapter numbers and publications relevant to each chapter are indicated as well. 14

Figure 2.1 Showcasing the repetitive nature of application execution for industrial CPS, including example atomic phases such as *Phase A* and *Phase B*, as well as an example combo phase, *Phase AB*. The figure also depicts the effects of different types of anomalies related to timeliness, some localised and some with cascading effects, delaying the subsequent phases. We can observe how certain anomalies result in localised effects, e.g., when Phase A is delayed or overtakes another independent Phase A, while other could result in lasting consequences, e.g., when Phase A is not concluded in time, shifting the start of the dependent Phase B. 17

Figure 2.2 The comparison of the quality of fit between (a) cumulative and (b) non-cumulative representations of data is depicted here, using quadratic regression functions and dummy data. A perfect fit means that $R^2 = 1$ and as it can be seen from the values of R^2, the cumulative case results in a much closer fit when limited to quadratic functions. Note how fluctuations in collected data points are softened when the data is transformed to the cumulative representation. 20

Figure 3.1 The high-level approach towards anomaly detection and identification is depicted. The imagined feature engineering is expected to involve more than a single step and will form the preprocessing of data, when implemented as a workflow. 24

Figure 3.2 The high-level approach towards countering the effects of detected and identified anomalies, involving actuation policies, the validation of their positive effect through a digital twin and their dispatch by means of known activations on the industrial CPS, are visualised. We also imagine that there will be a database of known and effective policies for previously encountered anomalies in place. 25

Figure 3.3 A diagram showing the discrete-event simulation environment as the digital twin of the semiconductor photolithography machine node is given. The diagram is considering the topology from the publish-subscribe broker's perspective and parsed traces that are converted into *write*, *read* and *compute* events. 32

Figure 3.4 Different event scheduling policies applicable during a simulation are shown, with (a) Policy 1, for close synchronisation of events based on initialisation times within the trace and (b) Policy 2, which considers the idleness part of computations when scheduling events. Policy 2 is especially interesting for observing the changes resulting from tweaking of the original behaviour, through actuation. 33

Figure 4.1 Our Classic ML workflow for the first use-case, a semiconductor photolithography machine, is shown. We can observe different data processing steps, organised in three main flows for software passport generation, for software signature generation and for data set generation. One important detail present in this figure is the use of the digital twin instance for hiccup injection purposes. 41

Figure 4.2 Plotting the behavioural coverage evaluation, using CPU utilisation differences for multiple workloads, i.e., by comparing total system vs the communication-centric view. Considering that we are monitoring 1/6th of the total application processes, we can see that the amount of observation loss is consistent for different workloads, while at the same time, the observation loss is rather close for both `top` and `getrusage`. 45

Figure 4.3 Plotting CPU utilisation differences for multiple workloads, observed vs simulated results, considering that the observed view is from our communication-centric monitoring using `getrusage`. We can observe that when it comes to CPU utilisation, simulations are rather close to what is collected via our partial monitoring view. Note the rather small scale of the y-axis. 46

Figure 4.4 Plotting process lifetime differences for simulations of different workloads, considering both Policy 1 and Policy 2. 47

Figure 4.5 Our experimental platform, electrical EFB metric collection set-up and the data processing pipeline for power signatures and power passports are presented (note the independent power supply to the fan). This diagram elaborates the data set generation (preprocessing) flow from large amounts of trace data for the Classic ML approach, leading to a training data set. 50

Figure 4.6 This diagram visualises the anomaly identification flow for the Classic ML approach, using much smaller trace batches with a previously trained classifier. 51

Figure 4.7 Different execution phases for an image processing task are drawn, i.e., the *atomic image operation* for loading of an image, the *atomic neural operation* for running the image through a neural network algorithm and the *combo cycle operation*, encompassing the first two atomic phases. 52

Figure 4.8 The diagram elaborating the mean passport generation flow is given. We are using the full set of x values (independent variable) from all available passports for the same metric and generating equal size sets of y values (dependent variable), by means of regression-based imputation. 53

Figure 4.9 Example mean passport plots are provided for (a) Image operation atomic phase and for (b) Neural operation atomic phase. Grey curves indicate the passports that the mean passport in blue is based on. While the procedure is the same for functions with any degree, all regression functions in these examples are quadratic. 54

Figure 4.10 An example plotting (a) the quadratic regression modelling result for a reference software passport, using cumulative CPU time as the EFB metric of choice, and (b) the corresponding violation after synthesising an anomaly. Aside from the quantifiable goodness-of-fit test results, here we can visually observe the deviation. 56

Figure 4.11 The two flows involved with the Advanced DL approach, (a) data set generation flow from large amounts of trace data, leading to a training data set, and (b) an anomaly identification flow, using much smaller trace batches with a previously trained classifier. Note the reduction in number of steps and the overall complexity of this approach compared to the Classic ML workflow. 60

Figure 4.12 The composition of the CNN model with the highest achieved accuracy through our grid search is depicted. The depth, i.e., number of channels per input, different layers and output are given in blue colour. 62

Figure 5.1 Depicted are normalised confusion matrices for (a) DT, (b) RF, (c) GaussianNB, (d) k-NN, (e) LinearSVC and (f) KernelSVM classifiers, considering the anomaly categories, Normal, Persistent Benign (p-benign), Persistent Harmful (p-harmful), Transient Benign (t-benign) and Transient Harmful (t-harmful) as labels, with the data set based on the combination of three metrics, i.e., CPU time, reads count and writes count, with the wafer processing phase. Best accuracies belong to DT and RF classifiers. 68

Figure 5.2 Depicted are normalised confusion matrices for (a) DT, (b) RF, (c) GaussianNB, (d) k-NN, (e) LinearSVC and (f) KernelSVM classifiers, considering the anomaly categories, Normal, NoFan and UnderVolt as labels, with the data set put together for the electrical current metric and for the cycle op. phase. 71

Figure 5.3 The set of highest accuracies resulting from our grid search over different hyperparameter choices are visualised in a parallel coordinates plot. In the above plot, *Slice size* stands for the input slice size from our sliding window implementation, *Slice shift* stands for the amount of shifting over data between consecutive slices, *Conv. count* stands for convolutional layer count, *Max conv.* stands for the width of the broadest convolutional layer, *FC layer* stands for the fully connected layer size, *Epochs* stands for the number of training epochs, *LR* stands for the applied learning rate, *Decay step* stands for the epoch intervals before a decay application on the learning rate with 0 denoting the absence of decay, and *Accuracy* stands for the achieved model prediction accuracy. 73

Figure 5.4 Showcases the compactness of the Decision Tree (DT) graph for the semiconductor photolithography machine use-case, using the Classic ML workflow and considering all three metrics, during the wafer op. phase. 76

Figure 5.5 Showcases the compactness of the Decision Tree (DT) graph for the image analysis platform use-case, using the Classic ML workflow and considering current as the metric, during the cycle op. phase. 77

Figure 5.6 The high-level procedure view to check the validity of post-update behavioural passports, before releasing a new service pack or software update is provided. By considering the pre-update passports as reference and the post-update passports as signatures for comparison, the workflow can be used as is to evaluate these signatures. 81

Figure 5.7 Depicted are typical executional periods, i.e., large scale phases, present in large and complex industrial CPS. It would be highly advantageous to detect and identify anomalies in every period as each has their own set of potential anomalies. Since passports for one period are not applicable to the rest, each period needs its own application of our methodology. 83

Figure 7.1 Potential future directions to the research work presented in this thesis are depicted, with extensions highlighted in yellow. 96

LIST OF TABLES

Table 4.1 Three categories of hyperparameters considered during the grid search to find the most accurate CNN model, alongside their considered variations during the search are provided. 62

Table 5.1 Overall prediction accuracy percentage of Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM) classifiers based on the data sets generated from different individual metrics, as well as their combination. The highest accuracies belong to DT and RF, when all three metrics, i.e., CPU time, reads count and writes count, are considered. 67

Table 5.2 Overall prediction accuracy percentage of Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM) classifiers, based on the data sets generated from different combinations of metrics and phases, are listed. The highest accuracies belong to RF and DT with the combination of current metric and the cycle op. phase. 70

Table 5.3 Elapsed times in seconds during different stages of the Classic ML and the Advanced DL workflows are compared. We can observe how the Classic ML workflow is rather fast when it comes to training and validation. On the other hand, the much faster preprocessing and the availability of GPU acceleration for the Advanced DL workflow stands out. 72

ACRONYMS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPD	Change Point Detection
CPS	Cyber-Physical System
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DL	Deep Learning
DT	Decision Tree
EFB	Extra-Funtional Behaviour
FC	Fully Connected
FCFS	First Come, First Served
GaussianNB	Gaussian Naïve Bayes
GB	Gigabyte
GPU	Graphics Processing Unit
IC	Integrated Circuit
ID	Identifier
I/O	Input/Output
k-NN	k-Nearest Neighbours
KernelSVM	Kernel Support Vector Machine
LinearSVC	Linear Support Vector Classification
LR	Learning Rate
LSTM	Long Short-Term Memory
ML	Machine Learning
OS	Operating System
PID	Process Identifier

RAM Random Access Memory

ReLU Rectifier Linear Unit

RF Random Forest

RMSD Root-Mean-Square Deviation

RMSE Root-Mean-Square Error

1 INTRODUCTION

1.1 EVOLUTION OF COMPUTING MACHINERY

The computing machinery is ever-present in today's modern society. The digitisation of our daily routines has been in progression full steam ahead. From personal computers to the Internet, online shopping to virtual meetings, smart phones to smart watches, one can list countless examples of the trend. Take the example of handheld cellular mobile phones for instance. These pocket-sized devices have come a long way in their usage and ubiquity since their commercial introduction back in the year 1984. Almost 37 years later, when compared to early versions, handheld cellular mobile phones have been improving in every aspect, as can be seen in Figure 1.1. One can think of aspects such as weight, size, battery life, but perhaps the most prominent of all, computerisation. Mobile phones are no longer just digital handheld devices, but computing platforms not that far from personal computers.



Figure 1.1: The evolution of mobile phones from their commercial release until the year 2014 is shown. (Image source Wikimedia, public)

Evolving any digital system, whether a handheld device, a consumer appliance, or a commercial machine, into a computing platform, will have two immediate outcomes. Computing platforms are capable of running software and software is not a static entity. By the virtue of expandability and the ability to add novel functionality through software, systems can perform an ever-increasing set of tasks. We have witnessed such characteristics in modern mobile phones, i.e., smart phones, where each new application, a.k.a., "app", adds a new set of capabilities to the hosting platform.

From personal to organisational activities, we not only rely on computing machinery, but we depend on them. This dependence is especially prevalent in the industry and manufacturing sector. The very aforementioned trend with respect to the role of software is also the case for industrial machinery. Current industrial evolution has brought forth the emergence of *Cyber-Physical Systems (CPS)* in industrial applications. Today, ubiquitous deployment of CPS in production and machinery, means that our economy and lifestyle is heavily reliant on the proper functioning of these *industrial CPS* [53]. The risk carried along with anomalies in industrial CPS is an economic one and in certain critical deployments, even life-threatening. Any form of deviation from the behaviour intended in the design of the system, or simply put, any form of malfunction for CPS, leading to undesirable results, is considered as an anomaly. For a couple of examples from day-to-day systems, think of the case of unintended acceleration for cars, or think of a coffee machine, pouring more than expected, overflowing a cup. Accordingly, the goal of this thesis is to provide methods to predict anomalies before they happen, based on the available information from the system at hand.

As industrial CPS evolve, control subsystems are computerised more than ever. Consequently, the role of software is ever-increasing in these systems. The continuous rise in the steering role of software alongside the heterogeneous, distributed and multi-node design of industrial CPS, has created a sharp rise in industrial CPS complexity. Although model-based design practices [39] have boosted design capabilities, CPS complexity remains hard to tackle. Not every aspect of system's operation and not every corner case is covered at design time [17]. A complete exploration of the *behavioural space* has simply become too expensive. There is a need for behavioural tracking of industrial CPS during their operation, while their sheer complexity means that following a single observable is not revealing enough.

Take the example of flight software for space missions. As it is argued in [18], most of the mission issues in the recent years are related to software. Both manned and unmanned missions have shown steady exponential growth in source lines of code for major missions from 1968 to 2005. We can observe this trend in Figure 1.2. Similar trends are

present for military aircraft embedded software, taking over a bigger portion of the total available functionality [18]. The software controlling a modern commercial airliner has 7 million lines of code.

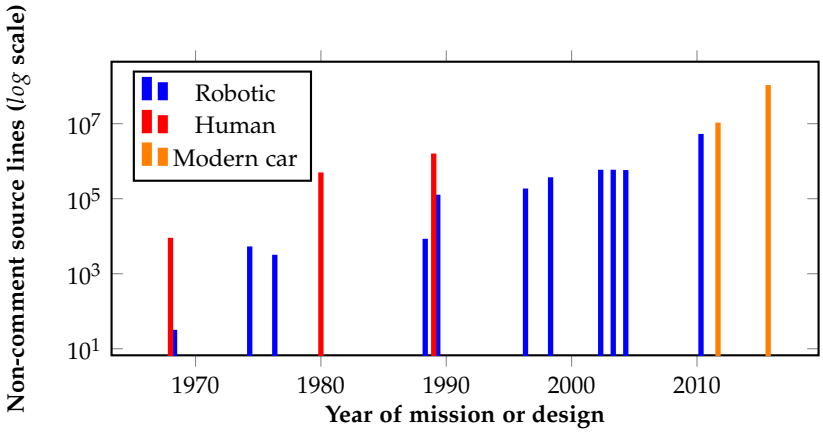


Figure 1.2: Plotting the progression of software size in terms of non-comment source code lines per every major space mission. Note that how human missions have larger codebase compared to robotic missions, pointing out the increase in software size when dealing with life-threatening scenarios. This plot is generated based on the data from [18, 34]. The plot also shows the amount of code in modern cars to provide a perspective on how fast CPS software is growing.

Another defining angle in the motivation of this thesis is the risks involved with large software development processes. A large software body goes hand in hand with the elevation of risks such as bugs, or unwanted behaviour, spawning from aspects such as concurrency, correct order of tasks, or resource management. Again, not every operational corner case can be covered at design time. The time and budget requirements applicable to space programs however, are seldom justifiable for earthly industrial applications. Such pragmatic realities leave us with an alternative choice, i.e., online methods to support scrutiny, analysis and decision-making at runtime, enabling resilience and self-healing capabilities. By the virtue of the methodology and the techniques presented in this thesis, we strive to facilitate the realisation of this endeavour.

Focusing on industrial CPS, we observe a few exploitable characteristics. Accordingly, the foundation for the following discourse in this thesis is the *inherent repetitiveness* and *inherent data-richness* of modern industrial CPS. As such, from whichever perspective they are looked at, industrial CPS operation and the information related to industrial CPS operation can be compartmentalised. This view will pave the way to the notion of execution phases, explained in Section 2.2. What brings such characteristics about is the unique construction of these systems.

CPS are an amalgamation of machinery, sensors, embedded computing and communication subsystems. The types of CPS incorporated in the industry, industrial CPS, have all of the above pieces tuned towards a specific task, making them *purpose-built*, with a contained domain of tasks and operational duties. That is what makes them repetitive. Especially, a plethora of sensors are incorporated in today's industrial CPS. These sensors include both hardware and software varieties and more importantly, what they have in common is large amounts of generated data. Though it is no trivial task to manage this data and make sense of it, if done with tact, it is an almost unlimited source to reveal many of the behavioural secrets of a system. As such, it is only logical to take advantage of this data-rich ecosystem and move towards *data-centric solutions*, which will in turn allow employment of techniques based on artificial intelligence.

1.2 DATA-CENTRIC SOLUTIONS

The large amount of data generated by production machinery sensors, has implications that are twofold. On the one hand, such a characteristic results in a need towards treating these systems with data-centric methodologies. Ergo, data-centricity is a virtue that could enrich the analytical capabilities over such systems. On the other hand, the data-centric approach has to be performed in a cautious manner, as it is rather easy to end up in a slippery slope, where there is either too much data to process in a timely fashion, or there is too much data dependence, or the data collection itself becomes too much of a resource-consuming overhead for the system. In extreme cases such as the Tesla case [71], excessive data collection results in hardware failure and ultimately, bricking of the whole car. In other words, we should optimise the amount of data needed to achieve results, while at the same time, the solution should not require extremely detailed data collections. As will be shown, both risks can be circumvented by considering:

- Communication-centric monitoring and modelling of the system and
- Breaking up the operations into reoccurring units of execution during the runtime of an industrial CPS.

When dealing with production grade systems, the same reality regarding the amount of available data, presents itself in a different manner. It is often rather challenging to come up with data collections that include what is needed, or rather what is desired, for the intended analysis. For instance, having design-specific knowledge is a big contribution to the analysis, facilitating the selection of ideal break points

and the ideal data collection points. The very same design-specific knowledge could very well be in the domain of trade secrets of the subject system. Thus, and we would like to emphasise this point, it is very important to have an optimal and flexible data collection strategy. At the same time, the solution has to make due with the available amount of data and achieve the best result possible. In Section 3.3, we will discuss different information positions and later on in Chapter 4 we show in our experimental implementations, how we can deal with poor information positions by adopting alternative solution workflows.

As mentioned with respect to repetitiveness, the discussions and the methodology provided in this thesis are fundamentally based on the characteristic that industrial CPS are *inherently repetitive*. Having considered repetitiveness and from a data-centric perspective, we show how the system's execution timeline can be compartmentalised into distinct *execution phases* and how the data contained in each phase can lead to the generation of a representative construct, a *behavioural signature*, which for certain reference executions is considered to be a *behavioural passport*. Here, a reference execution, a.k.a., a golden execution, is an execution in which the system's behaviour is considered to be normal. The data collected during such executions is the source to build behavioural passports from. The same exact procedure is the case for behavioural signatures, except that we do not know if the execution can be considered as normal or anomalous. Furthermore, we describe how *Extra-Functional Behaviour (EFB)* and different metrics reflecting EFB can be captured. EFB reveal beyond the functional specifics of a system, especially beyond the behaviour encoded within its software code. EFB can be captured by collecting metrics such as CPU time, or electrical power, amongst many others. Such metric data is the source when generating behavioural signatures and passports. The consideration of metrics revealing the EFB of industrial CPS is especially important as industrial CPS portray non-deterministic behaviour. This is mainly due to their continuous interaction with the physical domain. As such, just looking at the functional behaviour falls short.

The high-level view of our anomaly detection and identification approach is depicted in Figure 1.3. We strive to detect and identify performance anomalies for industrial CPS in an online fashion. We also strive to facilitate acting upon these detections and deployment of actuations on the industrial CPS at hand, with the aim of thwarting performance anomalies, or reducing their detrimental effects. Without going into too much detail at this point, one major role for presence of a digital twin is exploration and validation of actuation policies. Actuation policies are formulae intended to alter the behaviour of the system, which are applicable through collections of activations on the system. To make sure that the behaviour-altering effects of a policy are what we have aimed for, we can first efficiently validate it on the digital

twin instance, representing the actual system. Although we provide the foreseen methodology and part of the workflow for the actuation step, i.e., a digital twin, however, the actuations and their deployment will be outside the scope of this thesis.

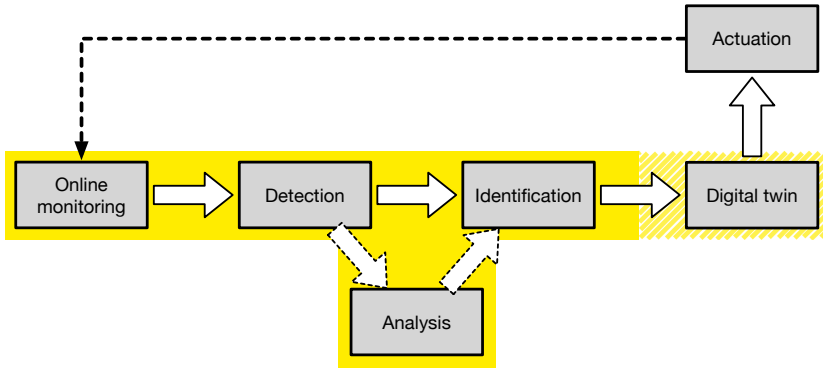


Figure 1.3: Depicted is the high-level view towards performance anomaly detection, identification and the envisioned countering of their effects. The focus of this thesis is highlighted in yellow, with a partial highlight on the topic of digital twin, for its implications on actuation is out of the scope of this thesis.

1.3 ROLE OF ARTIFICIAL INTELLIGENCE

We would not stand an earthly if we ignore Artificial Intelligence (AI) as the culmination of data-centric solutions. We have witnessed the emergence of solutions based on *traditional Machine Learning (ML)* and more advanced models, i.e., *Deep Learning (DL)*, for a plethora of problems for quite some time now. These methods have become an integral part of any data-centric method of choice. The industry in particular, reaps the benefits of such solutions in production systems.

DL models come in many forms. Considering the fact that the bulk of the monitoring data from industrial CPS is represented as time series, our attention is especially drawn to *Convolutional Neural Networks (CNN)* alongside traditional ML models. As the traditional ML and DL together provide more than just one way to solve a given problem, it is of utmost importance to pick the right solution and employ the right supporting workflow. For industrial systems in general and industrial CPS in particular, the extent of resource consumption and timely operation could very well mean the difference between success and failure, depending on the relevant requirements. In other words, it is not just about the accuracy of answers to problems, but also how fast and how efficiently they can be found.

Within this thesis, we do explore the balance between these factors for our methodology, addressing the challenge of anomaly detection and identification in industrial CPS. In this context, we will be comparing two ML-based solutions, namely, *Classic ML* and *Advanced DL* workflows, developed as alternative approaches. The Classic ML workflow incorporates regression modelling and traditional classification algorithms, e.g., decision tree and random forest, whereas, the Advanced DL workflow incorporates limited data preprocessing steps, taking advantage of CNNs. We will see how the Advanced DL workflow is different in its requirements for the amount of domain specific knowledge, expertise and understanding of the system that is necessary for the Classic ML workflow. Our Advanced DL method is a truly black box approach, requiring no insight into the data or the internals of the system, but at the same time, has its own limitations.

1.4 CYBER-PHYSICAL SYSTEM COMPLEXITY SPECTRUM

Industrial CPS come in different sizes and with different levels of complexity. Examples can be systems as complex as factory production lines, radars, or a semiconductor photolithography machine (Figure 1.4), or something as simple and contained as an embedded platform, and everything in between, interacting with the surrounding environment. With the goal of covering this vast spectrum, we have opted for cases from the extremities. Our first demonstrator is a semiconductor photolithography machine from ASML, for which the main computing node has been studied. The opposite case, is an embedded platform, used for detection of cars in images taken with the system's camera. Although for both cases we follow the same methodology, the exact workflows and data manipulations along the way are, to a limited extent, platform-specific.

As a definitive example showcasing industrial CPS, let us take a closer look into a semiconductor photolithography machine and its major tasks to understand the types of inherent repetitiveness present in these systems. More specifically, let us look at the wafer processing flow in these machines.

A semiconductor photolithography machine applies *Integrated Circuit (IC)* design patterns to silicon wafers through the photolithography process. A *silicon wafer*, a.k.a., silicon substrate, is a circular slice of single crystalline silicon. The main purpose of the photolithography process is to reflect the IC design from a reference pattern, a *reticle*, onto numerous dies on a wafer. Each *die* will be developed into an exact copy of the same IC design residing on the reticle. The process involves high-energy light beams going through the reticle and chemically interacting with sensitive material applied to wafers. In other words, wafers become exposed and will be taken for further processing steps



Figure 1.4: A state-of-the-art semiconductor photolithography machine is shown as an example complex industrial CPS, running numerous computing nodes with heterogeneous architectures and interacting with the physical domain. (Image courtesy of ASML Nederlands B.V.)

outside the photolithography machine, such as chemical etching, very much like developing a photograph. Without going into too much detail, the main order of tasks during the wafer processing workflow can be elaborated as follows:

1. Initially, a desired IC design pattern, a reticle, is selected and loaded.
2. The machine is also loaded with preprocessed wafers and one is fetched for exposure.
3. The loaded wafer is scanned, mainly for metrology, i.e., positioning purposes, as well as structural fault detection.
4. The wafer is exposed in consecutive steps inside a stepper unit, with each step exposing a single die on the wafer, until the whole wafer is exposed.
5. The wafer is released and moves to further processing and testing stages outside the photolithography machine.

Considering the order of tasks, there are different levels of repetition present. Item four involving die exposure is a repetitive task within the stepper unit and the number of repetitions depends on the number of dies to be exposed on the wafer. Another level of repetition is items two to five, which are repeated for every new wafer. Wafers are most often exposed in batches as these machines are intended for mass

production. There is yet another level of repetition present, covering all of the aforementioned tasks and it starts with selection of a new reticle. The wafer processing workflow and the main repetition cycles present in it are depicted in Figure 1.5.

1.5 SYNOPSIS

The research project we have embarked on, is aimed at the need to address unpredicted anomalies in industrial CPS. The project was conceived around a semiconductor photolithography machine. In this context and for the specific example of a photolithography machine, missing an internal deadline at any of the data processing, wafer scanning, or wafer exposure stages, is considered a typical anomaly. Such an anomaly, in most cases will result in a rejected wafer, or reduced machine yield. We can also consider more granular anomalies, for instance, a software component anomaly for missing a deadline during a data processing step. Such sensitivities relevant to timeliness are valid for most industrial CPS.

It is highly advantageous to *detect* when the system behaviour leaves the normal state and steps into the anomalous domain. It is also highly advantageous to *predict* the type of anomaly that the system is going to experience by *identifying* the *anomalous trend*. Being privy to such information in a timely fashion, it is conceivable that certain anomalies could be circumvented, or their deprival could be diminished, either automatically or with human intervention. Though it may be perceived as trivial at first, however, the actual challenge to tackle lies in the efficiency and accuracy of predictions, while relying on a constrained amount of data. This is both a limitation and a necessity, for we will not have the liberty of arbitrary data collection, because of a multitude of reasons. At the same time, processing vast amounts of data is equally detrimental to a timely solution.

1.6 RESEARCH QUESTIONS

Considering what we have introduced so far, the challenges ahead can be expressed in the form of research questions to ponder over. Given that modern industrial CPS are data-rich ecosystems, we would like to delve into the matter by answering the following:

RESEARCH QUESTION 1

How can we follow behavioural diversity in industrial CPS through the variations embedded within sensory data, in an efficient manner?

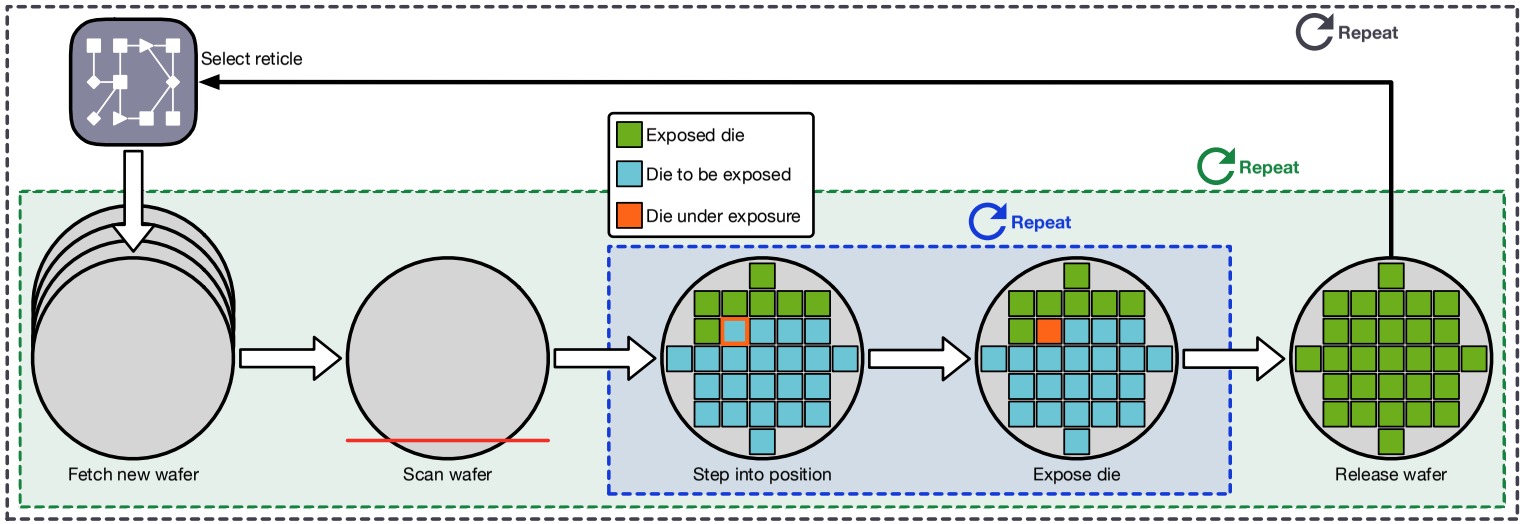


Figure 1.5: Depicted is a brief view of the order of tasks included in a wafer processing workflow, executed by a semiconductor photolithography machine. Different levels of repetition of tasks are observable, i.e., at the die exposure level, where the wafer is stepped through and exposed one die at a time inside the stepper unit (blue repetition area), at the wafer batch processing level, where a number of wafers are exposed using the same reticle (green repetition area), and at the reticle exchange level, where a new reticle is selected for the processing of an upcoming batch of wafers.

RESEARCH QUESTION 2

How can we demystify such embedded variations by only taking a partial, but yet, a descriptive view of the sensory data, to detect, identify and predict anomalous behaviour?

RESEARCH QUESTION 3

What are the different approaches towards the identification of such anomalous behaviour? What are the implications for production systems implementing such approaches?

1.7 OUTCOMES OF THE RESEARCH

The following is a compact listing of the outcomes of the research activity, leading to the conception of this thesis. Accordingly, we present the following novelties:

- A data-centric methodology towards performance and physical anomaly detection, identification and prediction during the operation of industrial CPS, which has been elaborated in Chapter 3;
- A communication-centric monitoring and modelling approach to dramatically limit the collection of EFB data and to facilitate composition of a digital twin, which has been elaborated in Sections 3.4, 3.5, 4.1.1 and 4.1.2;
- Purposeful compartmentalisation of industrial CPS runtime activity into executional units of operation, playing a fundamental role in both analysis and generation of behavioural representations for repetitive systems, which has been elaborated in Section 2.2 with its implications affecting concepts of anomalies, behavioural passports and communication-centric modelling;
- Approaches towards composing executional unit representations, i.e., behavioural signatures and behavioural passports, as the constructs reflecting the behaviour of a system in a compact and definitive fashion, which has been elaborated in Sections 2.4 and 4.2.1, as well as Figure 4.10;
- The demonstration of our data-centric workflow with proof-of-concept set-ups for two use-cases from the industry, covering the full spectrum of industrial CPS from complex to embedded, addressing both performance and hardware-oriented anomalies, with per use-case elaborations provided in Sections 4.1 to 4.3;
- And the comparison of two different machine learning approaches in the final phase of our methodology, along with data-centric workflows supporting each, revealing advantages and disadvantages of their usage, elaborated in Sections 4.4 and 5.3.

We also include elaborations and implementations on the following techniques:

- Software-based and hardware-based, internal and external probing techniques, relevant to the collection of EFB metrics;
- And the use of representative constructs in feature extraction and identification of behaviour, resulting in anomaly detection and prediction with high accuracies.

1.8 AUTHOR'S PUBLICATIONS AND THESIS STRUCTURE

The author of this thesis, has based it on the following closely related publications, with him as either the first or the second author. Here we provide only the titles, alongside the role the author has played during the composition of every publication. For complete bibliographic details, citation links can be followed. The list is provided in chronological order.

1. "Work-in-Progress: Communication-Centric Analysis of Complex Embedded Computing Systems" [56] (P1)
2. "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems" [50] (P2)
3. "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems" [55] (P3)
4. "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
5. "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
6. "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
7. "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Following the above enumeration, as the first author in publications P1, P3, P5, P6 and P7, the author of this thesis has been involved in all aspects, including data curation, formal analysis, investigation, methodology, software, validation and writing of the original draft. With regards to publications P2 and P4, the author of this thesis had a less prominent role in software, validation and writing of the original draft, but he was equally involved in the rest of the activities.

This introduction is followed by the fundamental concepts that are needed throughout this thesis, given in Chapter 2. This will pave the way for our methodology, introduced in Chapter 3. The realisation of our methodology comes next, as Chapter 4 elaborates our experimental implementations for both use-cases covered in this thesis. As our methodology ultimately leads to classification of behaviour, Chapter 5 covers the classification results from our use-cases, in great detail. This chapter also includes our discussions. Hereafter, we provide the related work in Chapter 6, going through the body of knowledge in close relation with the concepts and the techniques we have considered in this thesis. Ultimately, conclusive remarks are presented in Chapter 7, which also revisits our research questions listed in the current introductory chapter.

Accordingly, a visual diagram of the covered subjects within this thesis and their supporting relations is provided in Figure 1.6. The diagram indicates publications directly relevant to each chapter, as well as subject-wise listing of the contents of each and every chapter. Aside from this structure, related publications are listed at the beginning of every chapter, except this introductory chapter and the Conclusion.

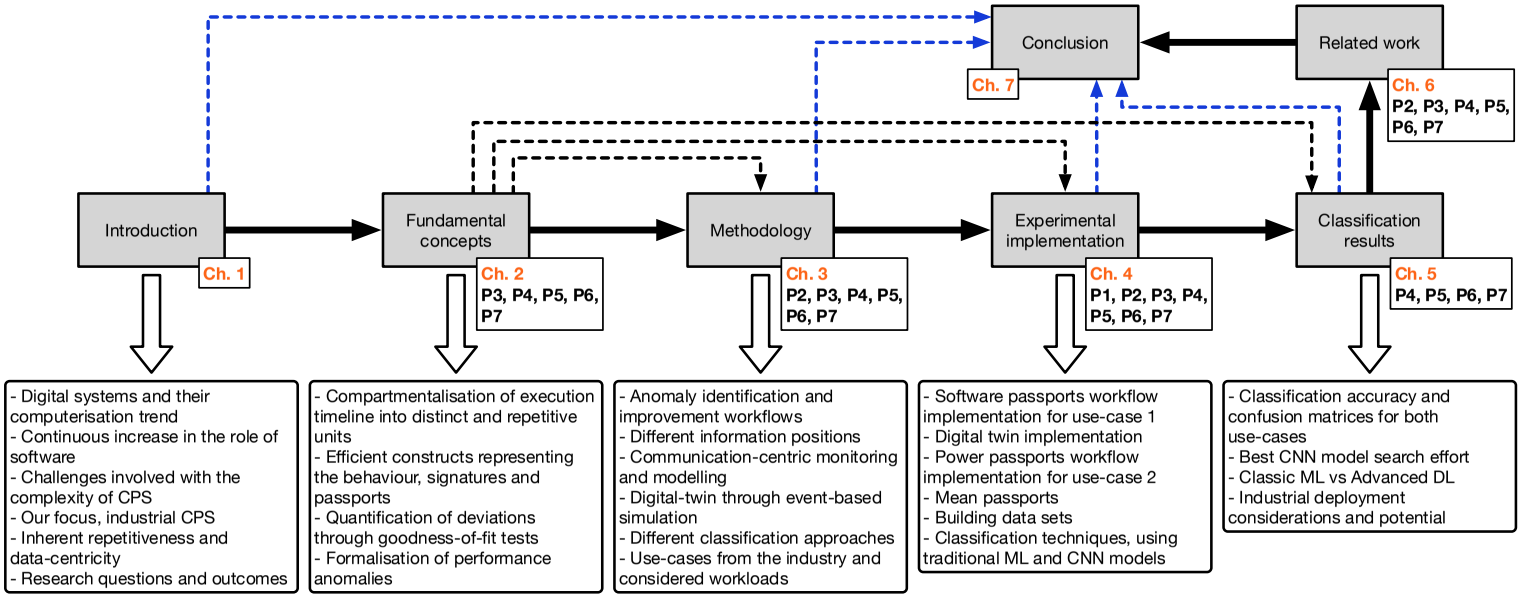


Figure 1.6: Shown is an overview of the topic-wise flow of chapters within this thesis, with solid arrows representing direct transitions and dashed arrows indicating connection of the content where there is no direct transition present, with black and blue dashed arrows indicating usage of fundamental concepts and collection of material for the conclusion from other chapters, respectively. Chapter numbers and publications relevant to each chapter are indicated as well.

2 FUNDAMENTAL CONCEPTS

The following includes fundamental concepts that are constantly referred to and mentioned throughout this thesis. Comprehension of the chapters to come, does require a priori knowledge of these concepts.

The contents of this chapter are mainly based on, but not limited to, the previously published conference and/or journal publications of the author. The publications of interest for Chapter 2 are:

- "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems" [55] (P3)
- "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
- "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
- "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
- "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Looking at the relevant body of knowledge, it must be pointed out that the fundamental concepts and techniques employed in this thesis, are not novel when considered as stand-alone instruments. We do go over the related work covering the use of similar or the same techniques, such as, classifiers based on traditional machine learning, classifiers based on deep learning, regression modelling, system modelling and simulation methods, and the use of side-channel metrics, in Chapter 6. Although many of our references relate to the general theme of anomaly detection and identification, the novelty of our methodology and implementations lies in the use of these pieces within a unique combination. We have specifically focused on the exploitable angles, inherently present in the design of industrial CPS.

2.1 EXTRA-FUNCTIONAL BEHAVIOUR

Extra-Functional Behaviour (EFB) convey a computing system's behaviour and as the name suggests, EFB are not directly derived from functional aspects of the system. Examples are, execution time, different latencies, throughput, power and energy consumption, amongst others. Metrics reflecting EFB are not only dependent on functional behaviour, but also on environmental circumstances, such as the platform itself, the input to the system and operational conditions. Environmental circumstances, being important variables for CPS, necessitate the role of EFB metrics in their monitoring and analysis.

Looking at EFB from a different perspective, one can observe that they reflect the effects of environmental circumstances, the platform and the input, which may be in physical form depending on the application, all directly related to physical aspects of a CPS. As such, we can consider EFB and the information provided by EFB as an interface, binding the digital and physical realms within the operational domain of a CPS, together.

2.2 EXECUTION PHASES

Execution phases are basically repeated units of execution, which are especially noticeable in industrial CPS operations. Repeated tasks can be broken down to their subtasks and higher granularity can be achieved to describe repeated units of execution. We call the smallest of these units an *atomic execution phase*. There may be (usually are) combined atomic phases that are also repetitive. Such repetitions involving two or more consecutive atomic phases are called *combo execution phases*. Figure 2.1 depicts these concepts under normal and anomalous execution conditions. For any particular system, analysis will reveal the best choice(s) from available atomic and combo phases.

Formally, an arbitrary phase p is denoted as a tuple of its start, t_s, and end, t_e, times within the execution timeline, i.e., $p = (t_s, t_e)$. Each execution phase category, atomic and combo, is considered as a partially ordered set, where the generic ordering condition is $t_{e_k} < t_{s_{k+1}}$, making the ordering strict. This means that phases within one category should not overlap. As such, we can describe atomic and combo phases as

$$P_{\text{atomic}} = \{p_i : p \in T_{\text{atomic}}, i \leq n, i \in \mathbb{N}^*\} \text{ and}$$
$$P_{\text{combo}} = \{p_j : p \in T_{\text{combo}}, j \leq m, j \in \mathbb{N}^*\},$$

respectively. Here, T is the set of available phase types, n is the number of atomic phases and m is the number of combo phases. Considering the simple example from Figure 2.1, we have $T_{\text{atomic}} = \{A, B\}$ and $T_{\text{combo}} = \{AB\}$.

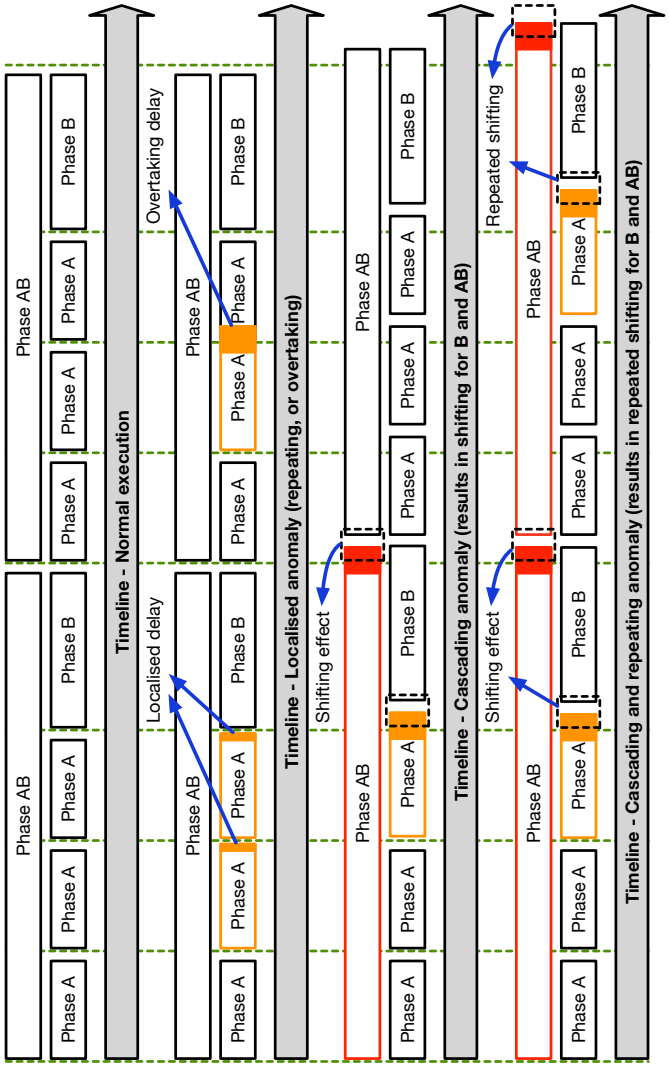


Figure 2.1: Showcasing the repetitive nature of application execution for industrial CPS, including example atomic phases such as *Phase A* and *Phase B*, as well as an example combo phase, *Phase AB*. The figure also depicts the effects of different types of anomalies related to timeliness, some localised and some with cascading effects, delaying the subsequent phases. We can observe how certain anomalies result in localised effects, e.g., when Phase A is delayed or overtakes another independent Phase A, while other could result in lasting consequences, e.g., when Phase A is not concluded in time, shifting the start of the dependent Phase B.

As a tangible example, looking back at the wafer processing workflow introduced in Section 1.4 and depicted in Figure 1.5, one can consider the exposure of a single wafer as an atomic phase and accordingly, the processing of a batch of wafers as a combo phase. This means that all the operations involved in the exposure of a wafer, i.e., "stepping into position" and "expose die", for all dies, fall under this atomic phase's umbrella. As a more granular alternative choice, a single die exposure can be considered an atomic phase, making a single wafer's exposure activities a repeating combo phase.

2.3 ANOMALIES AND FAULTS

Anomalies in general can manifest themselves as unreliable or subpar performance behaviour. The manifestation might also be in other forms of unexpected and harmful system behaviour and not necessarily performance-related. The two important terms to consider here are performance anomaly and fault [55]. A *performance anomaly* is any *readily* detectable deviation in the system's performance behaviour. For instance, if a computational job takes significantly longer than it should to be fulfilled, a performance anomaly has occurred. Detecting the actual *fault* causing the performance anomaly however, requires *insight* and analysis of the internal interactions of the system. As such, a performance anomaly is the result of a fault, but at the same time, not all faults will necessarily lead to a performance anomaly. Following the notation of execution phases and considering t_s and t_e as expected start and end times vs t'_s and t'_e as realised ones, we can expect to have the following anomalous situations:

$$t_{e_k} < t'_{e_k} < t_{s_{k+1}} \text{ and } t'_{s_{k+1}} = t_{s_{k+1}} \tag{2.1}$$

$$t_{e_k} < t_{s_{k+1}} \leq t'_{e_k} \text{ and } t'_{s_{k+1}} = t_{s_{k+1}} \tag{2.2}$$

$$t_{e_k} < t_{s_{k+1}} \leq t'_{e_k} < t'_{s_{k+1}} \tag{2.3}$$

Considering the relation between t'_{e_k} and $t_{s_{k+1}}$, we see that Equations (2.2) and (2.3) break the ordering condition, i.e., $t'_{e_k} \geq t_{s_{k+1}}$. These instances will be further discussed below.

TRANSIENT ANOMALIES A transient or localised anomaly is visible for a short period of time, or occurs only a few times. In such cases, either the fault is a short-lived one, or its impact to the overall execution is rather contained. For instance, with regards to timeliness, transient anomalies could be seen as delayed tasks in a contained part of the execution timeline. We can also think of tasks without any dependency that do overlap and end up sharing available resources, as visualised in Figure 2.1 for the localised anomaly with independent consecutive

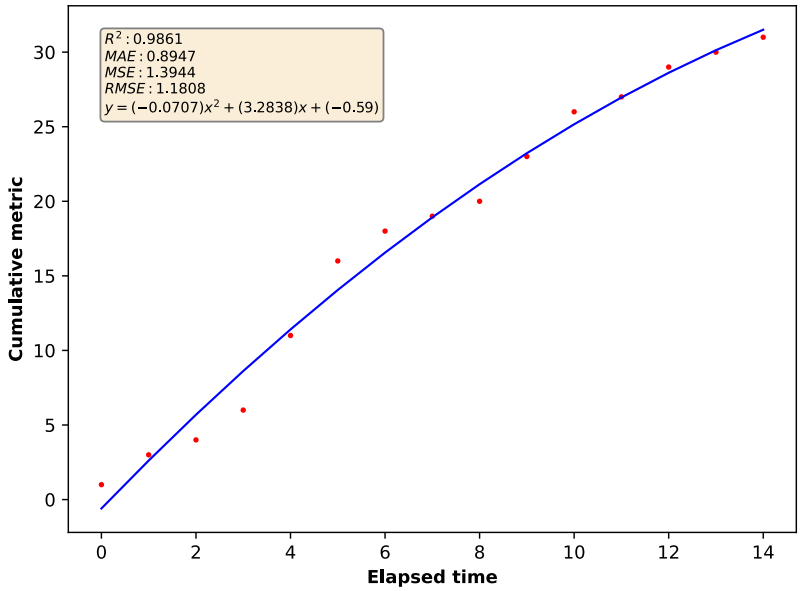
Phase As. In this context, Equations (2.1) and (2.2) point to localised and overtaking delays, respectively.

PERSISTENT ANOMALIES In contrast to transient anomalies, a persistent or repeating anomaly is of the type that either will keep reoccurring, or will create cascading delays for all subsequent tasks. This could be the result of, for instance, dependencies between tasks, or excessive amount of delay. Another factor is the placement (time) of anomaly. There will be no room for compensation for an anomaly occurring towards the end of a phase. The overall cost to the execution timeline is higher for such anomalies, as depicted in Figure 2.1 for cascading and repeating anomalies. The cascading effect is the result of Phase B being dependent on Phase A, as well as each Phase AB being dependent on its predecessor. Both situations result in a shifting effect, as formalised in Equation (2.3).

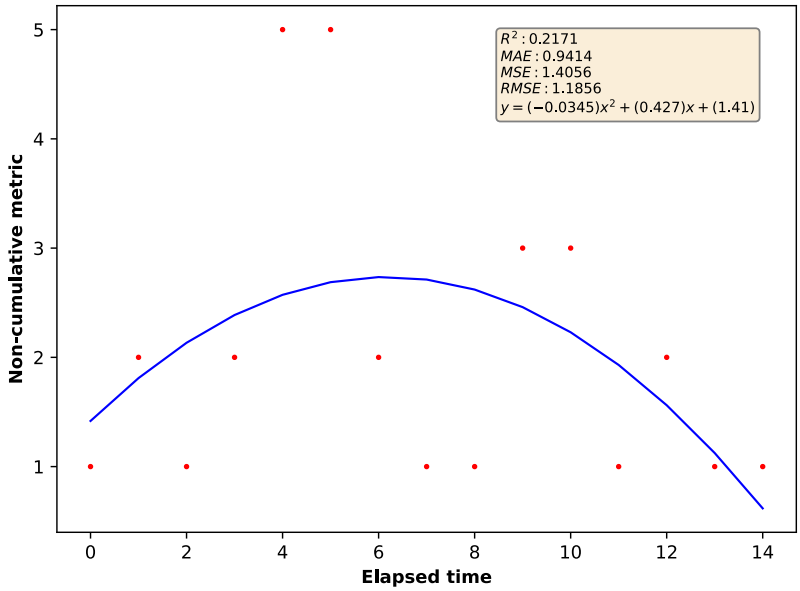
2.4 BEHAVIOURAL SIGNATURES AND PASSPORTS

Signatures and passports [55] are data-centric representations of EFB, for they are composed using the time-series data, collected during the execution timeline of a software running a system, or any executed application for that matter. A behavioural signature is the representation of an execution phase, modelling the trend of EFB metrics collected during that phase. Our modelling technique of choice for phase data is *regression modelling*. A behavioural passport is a reference signature, collected under normal and reference execution conditions, which is used in comparisons. As such, collected values for a metric of choice during an execution phase, for instance CPU time, will be transformed into cumulative data points in time, resulting in a time series collection. Using regression modelling, the closest function interpolating these data points will be generated and is used as the representation for that particular execution phase. The motivation behind opting for cumulative representation is to generate a regression, interpolating the set of data points as closely as possible. As depicted in Figure 2.2 using dummy data for demonstration, cumulative representation softens the fluctuations of data and creates a monotonically increasing time series, which in turn results in a monotonically increasing regression function.

Note that after the cumulative transformation, we practically end up with a different set of points and in turn, a different regression function. The comparison of R^2 values is intended as the comparison of the fit quality, hence goodness-of-fit. A close fit for the non-cumulative points is also achievable, but with a much higher degree for the regression function. We are also aware that a cumulative transformation will also cumulate the amounts of errors, i.e., unobservable error, per data point.



(a) Cumulative data points



(b) Non-cumulative data points

Figure 2.2: The comparison of the quality of fit between (a) cumulative and (b) non-cumulative representations of data is depicted here, using quadratic regression functions and dummy data. A perfect fit means that $R^2 = 1$ and as it can be seen from the values of R^2, the cumulative case results in a much closer fit when limited to quadratic functions. Note how fluctuations in collected data points are softened when the data is transformed to the cumulative representation.

Since we only have a sample and not the total population, this amount will be non-zero. However, regression modelling already introduces residuals, i.e., fitting deviations, per data point. As we would like to limit the degree of the regression function modelling the data, the amounts of unobservable errors are minute in comparison to residuals. Throughout this thesis, quadratic regressions prove to have sufficient fingerprinting accuracy for our data.

Regression-based fingerprinting of behavioural trends allows us to make use of statistical goodness-of-fit measures. As our measures of interest, the *coefficient of determination* (R^2), as well as the *Root-Mean-Square Deviation* ($RMSD$) or *Root-Mean-Square Error* ($RMSE$) provide quantified measures of deviation. These measurements can be utilised in two manners. The first use is to evaluate how close the generated regression function is interpolating the considered data points. In fact, such measurements motivate our choice of cumulative data representations, as well as the sufficiency of quadratic regressions throughout this thesis. The second use is to evaluate how well a behavioural passport fits the data points of a behavioural signature, or in other words, how much the collected points deviate from the passport. One can consider R^2 and $RMSD$ outputs as is, or base the evaluation on the percentage of deviation for R^2 and $RMSD$. Here, we provide the formulas for R^2 and $RMSD$ as a reminder, in Equations (2.4) and (2.5) respectively, such that

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \text{ and} \tag{2.4}$$

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-(k+1)}}. \tag{2.5}$$

In these equations, $\hat{y}_i$ represents the estimated response for the ith observation, $\bar{y}$ represents the sample mean, y_i represents the observed response and n the number of data points. Regarding $RMSD$, $n-(k+1)$ is the degrees of freedom, with k being the degree of the polynomial regression and we have to consider the intercept as well, hence, $k+1$. As such, $n-3$ is the degrees of freedom for a quadratic regression function [36, 60]. When it comes to deviation, if we consider the comparison between behavioural passports and behavioural signatures, y_i represents the data points collected for generation of the deviating (supposedly) behavioural signature, while $\hat{y}_i$ will be the values estimated by the behavioural passport.

Note that signatures are calculated per metric and per phase. For applications with multiple processes, different phases will potentially include activity data from multiple processes. This means that there will be signatures not only per metric and per phase, but also per process. Process identifiers can be used as an indicator to differentiate

between signatures based on the same metric and within the same phase.

Considering the use-cases included in this thesis, we have opted for both internal metrics, e.g., CPU time, reads count, writes count, and external metrics, e.g., electrical current, resulting in our selection of *software signatures*, *software passports* and *power signatures*, *power passports*, as our naming scheme of choice, respectively.

2.5 TIME SERIES DATA CLASSIFICATION ALGORITHMS

As we are following a data-centric approach in our methodology, we are employing machine learning and classification algorithms [49, 58]. When it comes to classification of time series data, both deep learning algorithms, i.e., neural networks, and traditional classification algorithms, e.g., decision tree, are viable options. We are especially interested in explainable results and the ability to backtrack achievements, making connections to the original data. This level of understanding is necessary when developing a methodology. Traditional classification algorithms are much more suitable for this goal, but on the other hand, they do require *feature engineering*. Accordingly, we have experimented with Decision Tree (DT) [69], Random Forest (RF) [7], Gaussian Naïve Bayes (GaussianNB) [23], k-Nearest Neighbours (k-NN) [16], Linear Support Vector Classification (LinearSVC) [4] and Kernel Support Vector Machine (KernelSVM) [15] classifiers for our demonstrators.

As mentioned, classification algorithms are not limited to traditional machine learning. When it comes to time-series data, especially forecasting use-cases, Long Short-Term Memory (LSTM) networks are effective [29]. Although our observations come in the form of time-series data, they consist of individually isolated batches. This makes CNN-based solutions relevant for our purposes. We will be exploring the viability of CNNs as an alternative to traditional algorithms in addressing our questions and we will be comparing them to what is on the offer from traditional machine learning.

3 METHODOLOGY

The following includes descriptions of our general approach towards the questions at hand. This general approach involves definition of methods towards achieving experimental solutions, as well as the depth of access a particular solution may require.

The contents of this chapter are mainly based on, but not limited to, the previously published conference and/or journal publications of the author. The publications of interest for Chapter 3 are:

- "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems" [50] (P2)
- "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems" [55] (P3)
- "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
- "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
- "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
- "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Considering the high-level view of our performance anomaly detection and actuation approach given in Figure 1.3, let us present an elaborated version here. The focus has been to take advantage of the current data-rich industrial CPS ecosystem.

3.1 PERFORMANCE ANOMALY IDENTIFICATION

Our methodology can be divided in two main workflows, the first of which addresses the requirement for performance anomaly detection and identification. Depicted in Figure 3.1, this anomaly detection and identification workflow is completely covered in this chapter.

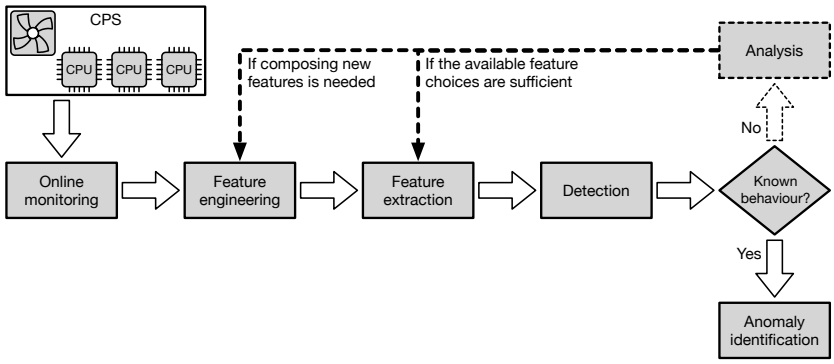


Figure 3.1: The high-level approach towards anomaly detection and identification is depicted. The imagined feature engineering is expected to involve more than a single step and will form the preprocessing of data, when implemented as a workflow.

There are numerous sources of data collection in a modern industrial CPS. These include hardware-based sensors, as well as metric collection tools within the software running the system. Additional low overhead software probes and serial hardware sensors can also be introduced, as it is the case in both of our demonstrators. Considering the approach given in Figure 3.1, this online monitoring, happening during the operation of the system, will produce large amounts of data. The feature engineering that follows, involves different data manipulation steps. Examples are parsing, extraction of important pieces of data and especially fundamental for us, compartmentalisation. The latter is where we divide the data according to execution phases. Following feature extraction, this data set will transform into a feature set. Some features are readily taken from the data set, while others are to be calculated, e.g., through generation of signatures and passports.

Detection is the step where deviation of the behaviour at hand from the reference behaviour is considered. In practice, there is a close relation between "Detection", "Known behaviour" and "Anomaly identification" blocks. We leave implementation details for sections ahead. Upon suspicion of new and unseen anomalous behaviour, the "Analysis" block comes into play. At this point, such an analysis is considered to be a manual effort, but we do foresee that certain analyses could be performed automatically. The goal is to update the feature set and retrain the employed classifier, allowing it for automatic detection of this newly introduced behaviour. Depending on the maturity of the feature set for this specific behaviour, there may or may not be a need to compose new features from the data set.

3.2 BEHAVIOUR IMPROVEMENTS

Detection and identification of anomalous behaviour will be followed with another workflow aimed at countering the anomaly itself, or its effects, keeping the industrial CPS fulfilling its purpose. This workflow, given in Figure 3.2, is the subject of our ongoing research. However, certain pieces of the puzzle are already in place, namely, a digital twin. Not every anomaly can be countered though.

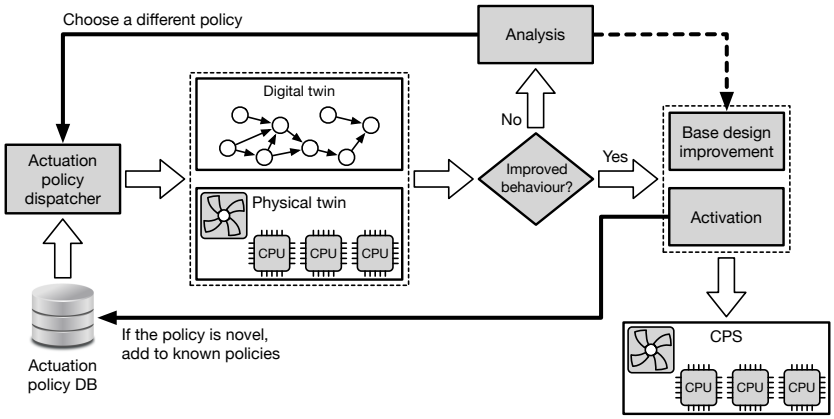


Figure 3.2: The high-level approach towards countering the effects of detected and identified anomalies, involving actuation policies, the validation of their positive effect through a digital twin and their dispatch by means of known activations on the industrial CPS, are visualised. We also imagine that there will be a database of known and effective policies for previously encountered anomalies in place.

We envision well-defined actuation policies for the anomalies that can be acted upon. The actuation policy chosen on the basis of the identified anomaly will first be applied to a twin of the system. For the most part, the anomaly detection and identification workflow from Figure 3.1 is also applicable to the twin. We would like to evaluate if the behaviour of the twin will be identified as normal, or if the deviation from normal is being reduced, in comparison to the anomaly itself. Upon an acceptable improvement, the actuation policy will be activated on the anomalous industrial CPS. The "Actuation policy dispatcher" may also come up with novel actuations, beyond what is previously known and available. Such policies, if resulted in improvements, will be added to the database. Whether an actuation policy results in an improvement or not, it could contribute to the base design of the system and fundamentally eliminate the anomaly from happening. For instance, when a software bug is discovered, or a corner case related to the order of calls in different processes is revealed, the base design can be improved. Although, analysis has to be performed to discover

such causes behind a policy's failure, correlate these causes with the design and develop rectifications for them. Similar to the anomaly detection and identification workflow, we expect the analysis following an unsuccessful actuation policy to be a manual one.

3.3 INFORMATION POSITION

Our methodology clearly relies on monitoring data collected during the operation of an industrial CPS and this collected data goes through quite a few transformations and preprocessing steps, until the anomaly identification is achieved, hence data-centricity. The proper application of these steps also relies on accurate metadata collection. For instance, considering our definition of execution phases from Section 2.2, a phase is defined by its start and end timestamps, $p = (t_s, t_e)$, making the collection of these timestamps an important part of our workflows. Phase boundaries do not necessarily have to be precise, but a relatively close vicinity is needed. The amount of inaccuracy we could get away with is use-case dependent. Accordingly, from a general perspective, we imagine three information positions to describe the amount of internal information we are privy to. These three positions from the highest amount of information to the least are *white box position*, *grey box position* and *black box position*.

As part of this thesis, we have experimented from a white box information position, a grey box information position, as well as a black box information position. For our first use-case we have adopted a white-box position with access to internals of the systems and availability of metadata. Further details on the use-case and our implementation techniques are given in Section 4.1. Our second use-case has been subjected to approaches from both a grey box position and a black box position. Details are provided in Sections 4.2 and 4.4, respectively.

3.3.1 *White box position*

A white box information position refers to a set-up in which there is almost unrestricted access to the internals of the system, e.g., via invasive probing from within the software. Note that by unrestricted access to the internals, what we mean is both literal access to the code and functional elements within the system, as well as access to the knowledge and operational details of it. In other words, a white box approach means that there is interaction with the software running the industrial CPS to collect behavioural metrics during its runtime. From a practical point of view, whether the interaction is with the Operating System (OS), through standard/custom tooling, or through probing of the workload handling application on top of the OS, it will

render the monitoring approach as white box. In such an approach, the application is analysed to discover its internal operations, e.g., I/O and computation operations. For a multi-process application, the analysis is performed on a per process basis. Note that these internal operations may or may not match the execution phases. There could very well be multiple processes that are active throughout different phases and different sets of operations belonging to these processes are involved in each phase.

Invasive probing is a powerful technique for gathering precise information. By implementing probes collecting system-specific and application-specific EFB metrics at the beginning and end of each operation, we will be able to collect the footprint of these metrics for that specific operation. For instance, as shown in Pseudocode 1, by collecting the amount of CPU time consumed by a process at the beginning and at the end of a computation operation, we will be able to calculate the amount of CPU time consumed by that operation for that computation.

Pseudocode 1: Description of software probes, with CPU time collection for a function performing computational work as an example to showcase the invasive probing technique, is provided. CPU time is given as an example metric and the approach can be applied to any metric, collectable from within the software.

```
Function main():
    initialisations;
    do some work...;
    compute(input, other arguments);
    do some other work...;
Function compute(input, other arguments):
    probe and save used CPU...;
    collect_trace(processID, event, timestamp_start, CPU_used);
    computation occurs...;
    collect_trace(processID, event, timestamp_end, CPU_used);
    return
```

One major consideration for this type of probing is the amount of effort required. Upon dealing with a complex industrial CPS with a vast software footprint, as it is the case nowadays, it would be a tough task to add probes in every process and every function within each process. Too many probes will also invoke the unintended computational burden as a side effect, affecting the actual behaviour of the system. After all, these probes do consume a minute amount of available resources. We will explain how one can relatively efficiently utilise invasive probing for industrial CPS through *communication-centric monitoring and modelling*.

3.3.2 *Grey box position*

Looking at an industrial CPS from a grey box information position sets more restrictions in place. Most notably, no internal probing, invasive or otherwise, will be possible. This level of restricted access could be due to various reasons, such as lack of access to the OS, the software component, the required operational knowledge, or strict controls over monitoring overhead. EFB metrics of choice for such cases are to be chosen from external ones, i.e., where the collection of EFB metrics would not interfere with the operation of the platform under scrutiny. These metrics are usually collected in a passive fashion and through hardware probes added to the system. Such collected metrics can in fact be considered as a form of side-channel information source. This information position is considered as grey box, for there will still be reliance on metadata information, in addition to the collected EFB metric readings. For instance, there will be a need for collection of execution phase boundaries.

The lack of interaction with cases that are being treated as grey box brings forth another limitation. Behaviour improvement is harder to achieve when following a grey box approach. We will show the example implementation of this information position, using electrical EFB metrics, while dealing with our image analysis platform use-case. When it comes to electrical metrics, one can use external power meters and power data loggers. Nowadays, more advanced embedded platforms have electrical metric collecting probes included in their design, reducing the need for additional hardware.

3.3.3 *Black box position*

To achieve a solution from a black box information position, the only acceptable source of data is collections by means of external monitoring, i.e., without interfering with the systems operation, which was also the case for the grey box information position. Additionally, for a black box position, no supporting metadata information from the internal state or operational details has to be in play. Such a black box information perspective has been applied to our image analysis platform use-case by considering a solution based on deep learning algorithms. Further details regarding the implementation of this solution can be found in Section 4.4.

Considering what we described for the three possible information positions, it is always desirable to work with less data and to work with less structural information for the data, if results can be achieved with similar quality. Here, quality refers to the accuracy of identification for anomalies, plus how quickly the identification can be performed. This also encompasses both design effort perspective and implementation ef-

fort perspective. A more advanced knowledge of the structural specifics for a given platform and the data relevant to that platform means that the analyst has to consider them in the feature engineering effort. Such knowledge and data access should be taken advantage of by producing clever manipulations of the data to construct revealing features within the data set, except if the solution can inherently discover interesting features on its own. We will perform a comparison between solutions using grey box and black box information positions in Section 5.3.

3.4 COMMUNICATION-CENTRIC MONITORING, MODELLING AND SIMULATION

As already mentioned in the description of the white box information position, for complex industrial CPS with an extensive software footprint, it is vital to limit the data collection effort, while keeping it effective. As such, we strive for effective condition monitoring, providing a partial view of the system that is representative enough for behavioural trend monitoring and identification purposes.

One of the main advantages of looking at an execution timeline from the perspective of phases, i.e., compartmentalisation into repeated execution units, is that we can focus on the most useful units of the execution timeline. Looking back at what we depicted in Figure 2.1, the considered phases out of all available ones in the execution timeline, e.g., just Phase A, are not required to be continuous. In other words, gaps are allowed in condition monitoring data. Such a selection of most useful phases results in elimination of others and in turn, elimination of software processes or functions associated with them. In this fashion, we are already reducing the behavioural surface we are to monitor.

Most multi-node and multi-process, i.e., distributed, industrial CPS include a message passing subsystem, predominantly in the form of a message broker based on the *publish-subscribe* software architecture pattern. To reduce the observed behavioural surface in a complex CPS, one can limit the observation to this message passing subsystem and specifically, to the message broker process. Although the observed behaviour will be a subset of the full system behaviour, it will be a valid representation, sufficiently reflecting the trend and changes [50, 56]. In such a *communication-centric* monitoring, this is achieved by collecting traces from purposefully planted probes in the code of the target communication subsystem. Such an invasive probing provides us with enough data to support modelling and simulation efforts. Collected information consist of EFB metrics that are read and processed. In its final form, the tracing data consist of communication, read and write, and computation events. Communication events contain information such as, senders, receivers, message sizes, and timing information.

Computation events contain information such as, CPU utilisation, CPU waiting times, process IDs and timing information.

We employ high-level modelling and trace-driven simulation of the CPS with the purpose of reasoning about the system behaviour. Accordingly, we have a detailed view of how different resources in the system are being used at different stages of the execution of a workload, e.g., CPU status and buffer utilisation. Moreover, the usage of high-level simulation models allows us to explore the effects of possible actuation mechanisms or countermeasures against performance anomalies in a safe environment, before applying these actuations to the real system.

High-level modelling and simulation are rather intertwined notions, as a simulation is the execution of a calibrated model. This is done through replaying of the previously collected traces by the simulator, while considering the high-level model's specifics, such as interconnections and high-level behaviour of different elements. In Sections 4.1.1 and 4.1.2, we describe how trace data is used to automatically synthesise and calibrate our high-level model. The model is a simplified, but descriptive enough representation of the studied system, hence high-level. Having the knowledge of involved actor types, i.e., data producers (writers), data consumers (readers), and data brokers (communication libraries), complimented with interconnection information, i.e., topology, is sufficient to construct the high-level model. Calibrating this model for simulation is done by considering metric recordings, collected during the system's operation. The aforementioned tracing format, i.e., read, write and compute events, is especially suitable for a discrete-event simulation.

3.5 DIGITAL TWINS AND THEIR USAGE

Collecting traces under anomalous behaviour might not be a straightforward task when it comes to real production-grade industrial CPS. The causes behind such a limitation can be numerous, but regardless, there will be a need for synthetically generated anomalous traces. To be able to resemble anomalous behaviour and generate synthetic anomalous traces, the role of a *digital twin* is a crucial one. This digital twin, which in our case is based on a simulation of our communication-centric model, should sufficiently replicate the behaviour of its physical exemplar. Additionally, using a digital twin will dictate the application of a harmonisation procedure on the data coming from the real system. The argument here is that since both types of traces, normal and anomalous, are going to be the basis for comparisons, they have to be harmonious, i.e., collected from the same platform, which in this case, is the digital twin. The effects of synthetically introduced anomalies will not be local and there is great potential for cascading effects. As a naive example, delaying a process that should send a message to another, will certainly

delay the reception at the latter, affecting its behaviour. We can achieve the conduction of such effects by using the very same digital twin as a reference platform. As such, all traces will be harmonised by passing through the digital twin. More details on this process will be given in Section 4.1.

Another important purpose of having a digital twin is its role in the actuation workflow. As seen in Figure 3.2, any chosen actuation policy will first be applied to a digital twin, in order to evaluate its effects on the representative twin's behaviour. Only if there was an improvement in a desirable direction, the policy will be activated in the real system. It is not a strict requirement for the twin to be a digital one. As discussed earlier, for platforms approached in a black box fashion with a smaller footprint, a *physical twin* can be a fair and even more suitable option. A physical twin is especially advantageous when interacting with an industrial CPS in the form of a distributed embedded system, composed of numerous identical nodes. In such set-ups, a single physical twin can verify actuations aimed at any of the anomalous nodes, i.e., anomalies and actuations localised per node.

3.5.1 *Digital twin for a communication subsystem*

Considering a white box information position and deployment of communication-centric monitoring and modelling, a digital-twin in the form of a trace-driven simulation is conceivable. The discrete-event simulation environment including real processes, simulated processes, interaction between simulated processes and the resource manager model, as well as the topology of the communication subsystem model is depicted in Figure 3.3.

Note that only processes that are using the communication module are being traced, e.g., *Real process i*, will not have a contribution in the collected behavioural metrics. The outcome of tracing to be used for the modelling and simulation, contains information such as, event ID, process ID, event initiation timestamp, event end timestamp, event type, CPU utilisation, and for communication events only, message size and destination process ID. For our trace-driven discrete-event simulation, we use the OMNEST simulation framework [77].

HARDWARE RESOURCE MANAGEMENT In the current version of our implementation in the OMNEST simulation framework, the resource manager model, i.e., hardware architecture, is modelled as a module receiving requests from different simulated processes and storing the requests in a ready queue. CPU resource management is performed using a First Come, First Served (FCFS), or a Round Robin scheduling policy. CPU time is distributed as tokens that are given to the simulated processes and returned when the processing is finished. The number of

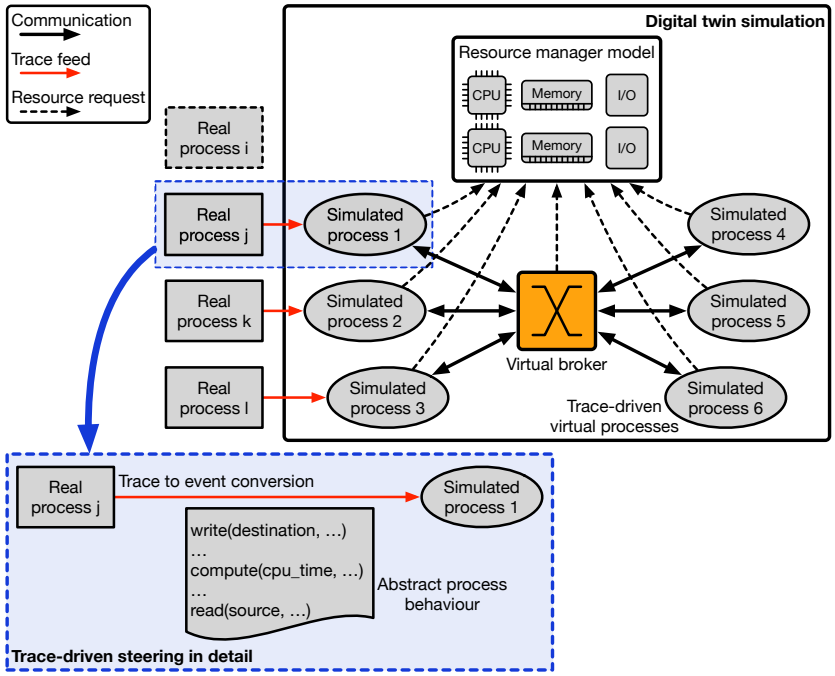


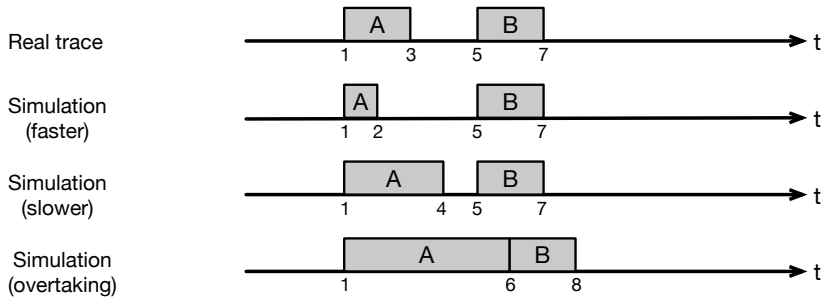
Figure 3.3: A diagram showing the discrete-event simulation environment as the digital twin of the semiconductor photolithography machine node is given. The diagram is considering the topology from the publish-subscribe broker's perspective and parsed traces that are converted into *write*, *read* and *compute* events.

tokens depends on the number of CPU cores included in the simulated system.

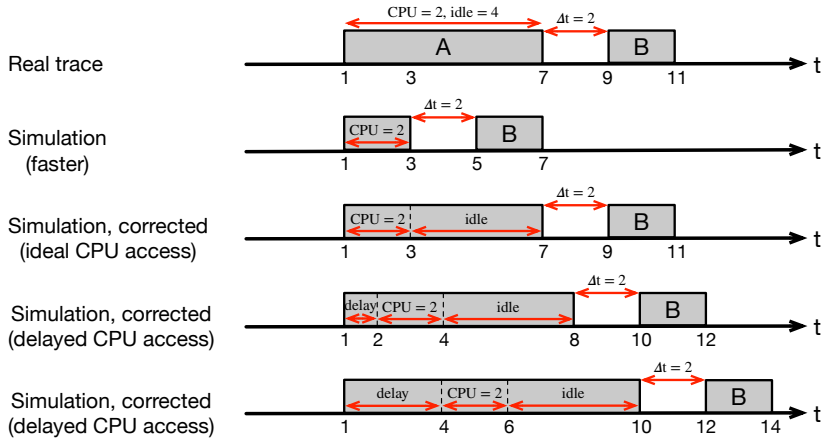
TRACE-DRIVEN VIRTUAL PROCESSES Trace-driven virtual processes are generic simulated processes, automatically generated based on the number of executed application processes in the real system, i.e., observed processes using communication-centric monitoring. Simulated processes are trace-driven and the simulation engine executes one event at a time. Process models are automatically inferred from the monitoring traces, avoiding the need to manually derive these models, which is typically very complex, or even infeasible to do for industrial CPS. Traces are parsed and converted to three types of events, namely, *write*, *read* and *compute*. These events are consumed by simulated processes and resource requests are created depending on the content of the trace, e.g., used CPU, type of event and timestamps. For instance, a process requiring a certain number of CPU cycles to perform the operation specified in the event, will create a CPU request. This request will arrive at the ready queue of the simulated CPU and will be served according

to the CPU scheduling policy implemented in the resource manager model.

EVENT SCHEDULING POLICIES On the side of the software application model, an *event scheduling policy* is present. The simulated processes read the trace events in order and one at a time, scheduling the next event according to a number of factors considered in the event scheduling policy. These factors are, current simulation clock, t_{sim}, the initiation timestamps of events encoded in the trace, t_{ini}, gaps between events in the trace, Δt, CPU utilisation and the idleness of events. Describing our event scheduling policy alternatives in Figure 3.4, real timestamps are given under the "Real trace" timeline, while simulated timestamps are provided under "Simulation" timelines.



(a) Event scheduling policy 1



(b) Event scheduling policy 2

Figure 3.4: Different event scheduling policies applicable during a simulation are shown, with (a) Policy 1, for close synchronisation of events based on initialisation times within the trace and (b) Policy 2, which considers the idleness part of computations when scheduling events. Policy 2 is especially interesting for observing the changes resulting from tweaking of the original behaviour, through actuation.

Figure 3.4a depicts our closely synchronised *Policy 1*, i.e., replay, with the first line showing two consecutive events from a real execution, running on the same processor. When simulating events A and B, simulated event time[1] could take longer or shorter to complete. Whether event A finishes earlier or later compared to its real version, the next event's starting time will be its own t_{ini} from the real trace, resulting in a close synchronisation with the real trace. The only exception is when A takes long enough to overtake B's t_{ini}. As such, B will start as soon as A ends, i.e.,

$$t_{ini_B} = t_{end_A}.$$

Note that for such an overtaking to happen, both tasks A and B should run on the same processor, already as mentioned.

Figure 3.4b shows how gaps between events, Δt, and process idleness during an event are accounted for, especially when scheduling *computation* events. First off, *Policy 2* does not initiate events strictly based on their real t_{ini}, but instead, it prioritised the gaps between events, regardless of faster or slower completion of the preceding one. As such,

$$t_{ini_B} = t_{end_A} + \Delta t.$$

Another consideration is the duration of an event and its associated CPU time. In most cases, an event includes process idleness, which is always added to the simulated event in the form of a wait. This *idle* duration is added after the event's computation is over and the CPU token is released. However, CPU availability might not occur at the very start of the event, resulting in the addition of another *delay* to the event, right before the computation. Considering such a delayed CPU access, the next event will start at

$$t_{ini_B} = t_{ini_A} + A_{\text{delay}} + A_{\text{compute}} + A_{\text{idle}} + \Delta t.$$

This second event scheduling policy focuses on increasing flexibility and less rigid follow-up of the real trace when releasing events to the simulation engine. Here, the policy avoids the use of initialisation timestamps specified in traces. While Policy 1 focuses on strict replay of the original observations, Policy 2 tries to maintain a more fluid behavioural accuracy, as closely as possible to the real execution, by not forcing the events to be scheduled at timestamps mimicking the real trace. Accordingly, Policy 1 is suitable for close replications of the original behaviour, whereas Policy 2 is a better choice to explore the impact of different changes to the system, e.g., as a result of actuation mechanisms.

1 Not the simulation itself

3.5.2 *Hiccup injection*

As one of the uses mentioned in Section 3.5 for a digital twin, we are considering synthetically generated anomalous traces, through injection of faults in the original traces. Our fault injection implementation focuses on affecting the duration of events by modifying the end-time of events [55], i.e., adding small hiccups. Our software allows us to completely configure the percentage of processes to be affected, the percentage of events to be affected, the type of event, i.e., communication or computation, the type of performance anomaly to be introduced, i.e., transient or persistent, as well as the overlap between the processes selected to be modified while introducing transient or persistent anomalies.

The three aforementioned types of events involve process idleness and CPU access delay. Duration of an event, t_E, which is the elapsed time from its initialisation, t_{ini}, till its end, t_{end}, is made up of CPU access delay, E_{delay}, CPU time, E_{compute}, and idleness, E_{idle}, such that

$$t_E = t_{end} - t_{ini} = E_{\text{delay}} + E_{\text{compute}} + E_{\text{idle}}.$$

We are considering additional synthetic delays, d_{synth}, as increases to the original duration of the event, t_E, which means that we are increasing the combined duration of CPU access delay, E_{delay}, and idleness, E_{idle}, resulting in an increased event duration, $t_{E_{\text{synth}}}$, such that

$$t_{E_{\text{synth}}} = E_{\text{compute}} + (E_{\text{delay}} + E_{\text{idle}} + d_{\text{synth}}).$$

Note that CPU access delay and idleness result from different conditions and we are not able to distinguish between them using the deployed tracing mechanism. While CPU access delay is simply a wait before CPU availability, idleness could be a result of I/O waits, or functional and data dependencies to other processes. Nevertheless, our synthetic manipulation of traces does not depend on the distinction between the two, since we will be adding to the overall delay of an event, i.e., anything other than E_{compute}.

3.6 ONE CHALLENGE, TWO APPROACHES

Different flavours of Artificial Intelligence (AI), whether traditional Machine Learning (ML) or more sophisticated Deep Learning (DL) algorithms and models, are good fits when dealing with large amounts of data. Given that modern industrial CPS provide this large amount of monitoring data generation capability through software and hardware probes, the use of ML is not a preference, but a necessity. Depending on the type of ML algorithm, certain amount of preprocessing is needed

to transform the data into consumable forms. We have considered two alternative realisations of our methodology from Figure 3.1, namely, Classic ML and Advanced DL. As it will be shown in Chapter 4, major parts of this preprocessing will be implemented rather differently, resulting in alternative characteristics and performance.

3.7 USE-CASES AND WORKLOADS

Following the discussion on industrial CPS complexity spectrum from Section 1.4, we apply our methodology to, and illustrate the results of this application, for two distinct use-cases. Both of these use-cases that we have developed demonstrators for, portray diverse levels of complexity, covering far ends of the industrial CPS complexity spectrum. Both use-cases have been taken directly from the industry. As our first use-case, the semiconductor photolithography machine, we have adopted a white box information position, in which we are able to add probes within the software to collect EFB metrics. We are also able to collect or estimate metadata information, such as execution phase boundaries within the execution timeline. Accordingly, the resulting representations for this use-case are named as *software signatures* and *software passports*.

For our second use-case, an image analysis platform, the system has been treated from both grey box and black box information positions, in separate demonstrations. We have applied our methodology from the grey box information positions while behaviour is monitored through electrical metrics, collected via external probing. Alongside metric data, we are also collecting phase-related metadata. The resulting behavioural representations for this use-case are named as *power signatures* and *power passports*. Though there are subtle differences between different cases within the early data manipulation steps, generated constructs for both have been treated with the same classifiers. With the black box information position, we have opted for our approach based on DL with Convolutional Neural Networks (CNN). This choice heavily reduces the need for the presence of metadata, while eliminating the need for behavioural signatures and behavioural passports.

The behaviour of industrial CPS is dependent on three factors. These are, the platform, including the hardware and the software running it, environmental specifics and the workload. For both use-cases, aside from the platform and environmental specifics, which are static throughout our experiments, the workload is the most defining source of variation. We have done our utmost by considering diverse workloads, both as solitary batches of jobs and queues of batches to be processed in order by these platforms. For the case of the semiconductor photolithography machine, we have considered batches of 2, 5 and 10 wafers, as well as a queue of 2, 5 and 10-wafer batches

bundled together. For instance, one of the differences when considering a queue of batches is that the platform will perform preprocessing of the 5-wafer batch while the exposure for the 2-wafer batch is in progress. For the case of the image analysis platform, which is less complex, we consider solitary 30-image batches, as well as queues of 10 such batches to push the system with heavier workloads.

4 EXPERIMENTAL IMPLEMENTATION

The following includes experimental and applied implementations based on previously devised methodology and real-world data collections from industrial use-cases. The full extent of implementation details specific to our use-cases are presented.

The contents of this chapter are mainly based on, but not limited to, the previously published conference and/or journal publications of the author. The publications of interest for Chapter 4 are:

- "Work-in-Progress: Communication-Centric Analysis of Complex Embedded Computing Systems" [56] (P1)
- "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems" [50] (P2)
- "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems" [55] (P3)
- "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
- "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
- "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
- "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Although the implementations of behavioural signatures and their generation in both of our proofs-of-concept follow the previously elaborated methodology, differences remain. We will describe each case separately. The *software passport* workflow and the *power passport* workflow refer to the implementations of our methodology for the semiconductor photolithography machine and the image analysis platform, respectively.

4.1 SOFTWARE PASSPORTS WORKFLOW

As noted in Chapter 3, our software passport workflow has been applied to ASML's semiconductor photolithography machines, as one of the real-world demonstrators for this thesis. The workflow is depicted in Figure 4.1 and follows a white box approach. This complex industrial CPS includes communication subsystems based on the *publish-subscribe architectural pattern*. The subsystem connects application processes from different software components, which in turn run on distributed computing nodes. The depicted software pipeline has been created to extract per process, per metric and per execution phase information from the simulated traces. Regression modelling techniques are used to transform the extracted data into regression functions that are most accurately approximating the performance of processes over time.

For this use-case, we are working with the main computing node in the photolithography machine, which is in charge of the wafer processing and wafer exposure tasks. Considering our definition of execution phases from Section 2.2 and the description of the wafer processing workflow from Section 1.4, we have taken a single wafer's exposure operations, wafer op., as the phase of choice for our methodology to be applied over. The selected phase, wafer op., is considered to be a *combo* phase for full processing and exposure of a single wafer. We consider this operation as combo, since it can be broken down to different sub-operations, which will be more fine-grained and could be considered as atomic phases. Nevertheless, the fact that it is a combo phase does not make a difference for our implementation, as our data-centric method is agnostic towards such abstractions. The understanding coming along with the analysis of different phase granularities does however help the researcher or engineer to have a basis, in case the chosen phase was not the best option.

4.1.1 *Effective sensing*

Considering the "Trace data" block from Figure 4.1, we strive to capture a partial view of the system's behavioural trend, small enough for monitoring and data manipulation, but also complete enough to have a semi-accurate view of the behaviour. The sensory data required for this goal is gathered online, through invasive probing, i.e., by adding probes in the code, in a communication-centric fashion. Communication-centric monitoring allows us to limit the introduced overhead, as the software probes are implemented at the system communication module, i.e., message broker. When communication takes place, involved processes are traced indirectly from within the broker calls. The overhead is much less, compared to a fully invasive tracing of each and every software process, at the cost of a reduction in the amount of information cap-

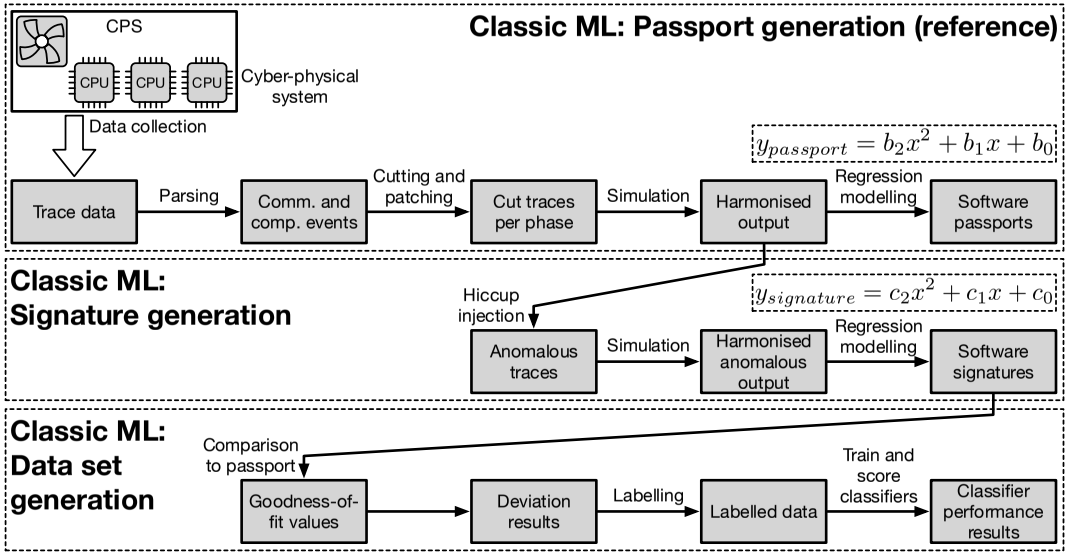


Figure 4.1: Our Classic ML workflow for the first use-case, a semiconductor photolithography machine, is shown. We can observe different data processing steps, organised in three main flows for software passport generation, for software signature generation and for data set generation. One important detail present in this figure is the use of the digital twin instance for hiccup injection purposes.

tured. More precisely, we will miss the information on processes that are not using the communication module. The current information captured by our tracing tool includes per process data such as, event start and end timestamps, CPU utilisation, message sizes, message ids, messages sources and destinations, and memory utilisation. Following the description of software probes for a white box approach given in Chapter 3, Pseudocode 2 elaborates our probing for communication and computation events.

Pseudocode 2: A high-level description of software probes is given, specifically for collection of communication and computation events at the publish-subscribe broker. Using such techniques, we can collect metrics for all callers of the broker functions, from within the communication subsystem.

```
Function main():
    trace_comp_start(processID, event, timestamp_start);
    do some work...;
    communicate(payload, other arguments);
    do some other work...;
    communicate(payload, other arguments);
    ...
Function communicate(payload, other arguments):
    probe and save used CPU...;
    trace_comp_end(processID, event, timestamp_end, CPU_used);
    trace_comm_start(processID, event, timestamp_start,
     size(payload));
    communication occurs...;
    probe and save used CPU...;
    trace_comm_end(processID, event, timestamp_end, CPU_used));
    trace_comp_start(processID, event, timestamp_start,
     CPU_used);
    return
```

When a process starts its execution, a `trace_comp_start` call saves the id of the process, the type of event and the start timestamp. Notice that this is the initial reference point, ergo the CPU time is not recorded. The process continues its normal execution until it calls the communication library, containing the probing code, i.e., `communicate`. When entering the communication library, the `trace_comp_end` event saves the timestamp (`timestamp_end`) and the amount of resources that were used by the application until that point, while the communication event is traced using `trace_comm_start` and `trace_comm_end`. Then, collected traces are processed to calculate time spent in communication and computation related tasks. Computation events may contain not

only CPU usage information, but also idle time, I/O time, etc., while communication events can also involve some CPU utilisation.

One of the tools available for invasive probing of metrics such as CPU time or memory usage, is the `getrusage` system-call. One might also opt for other tools or techniques, depending on what is available for the metric of interest and the platform. In our case, we are specifically focusing on CPU time, as well as read and write counts.

We have mentioned in Section 2.4 that behavioural passports and signatures are collected per metric, per phase, and if multiple processes are present, per process. This may seem like too many passports to keep track of. However, when passports are combined with the communication-centric monitoring strategy, from the perspective of a broker, an unexpected advantage is revealed. When dealing with data producers and data consumers and by considering read event counts and write event counts during a phase as metrics of interest, we are actually considering mutually exclusive metrics. In fact, the absolute majority of such processes are either strictly data producers, or strictly data consumers. In other words, the publish-subscribe software architecture enforces an inherent limit over the available metrics per process, reducing the number of passports to be maintained. Note that this is the case for passports based on univariate regression.

4.1.2 *Digital twin and behavioural coverage*

As mentioned in Section 3.5, there is a need for a digital twin when generation of anomalous traces is not straightforward. This is observable under "Classic ML: Signature generation" segment in Figure 4.1 as the "Hiccup injection" arrow. For our semiconductor photolithography machine use-case, we have to rely on a digital twin to generate anomalous traces. Accordingly, a digital twin in the form of a trace-driven discrete-event simulation, implemented using the OMNEST simulation framework [77], which is suitable for this purpose, comes into play. We have assessed our aforementioned methodology from Section 3.5 using this simulation. The three main metrics that we have considered to evaluate the quality of our proposed methodology are:

1. the behavioural coverage of our communication-centric monitoring and modelling,
2. the accuracy of simulations regarding CPU utilisation and
3. the accuracy of simulations considering process lifetimes.

In order to perform the simulations, traces from a fully functional ASML photolithography machine, having the same software components as a production machine, and thus the same behaviour, minus all the moving robotic parts, were used. This machine is used by ASML

to perform machine throughput qualification as a regular basis. Traces were collected by introducing probes in one of the main communication libraries, following the publish-subscribe architectural pattern. These traces were post-processed, and in turn, used as input for the simulation model developed in OMNEST. The collected traces correspond to four different workload scenarios[1] using a single recipe (pattern). In real operation, recipes are optically projected onto a silicon wafer covered with a film of light-sensitive material. The load scenarios consist of:

1. applying the recipe to expose 2 wafers, workload *2w*,
2. applying the recipe to expose 10 wafers, workload *10w*,
3. sequentially applying the recipe to expose a combination of 2-wafer and 10-wafer batches, workload *2+10w* and
4. sequentially applying the recipe to expose a combination of 2-wafer, 5-wafer and 10-wafer batches, workload *2+5+10w*.

The difference when processing diverse batches of wafers lies in the fact that different application processes are triggered depending on the number of wafers to be processed, as well as the number of the queued batches.

Figure 4.2 shows how the CPU utilisation trend of the total system is closely matched with the combined CPU utilisation of the observed processes, i.e., the ones using the communication subsystem. We have compared both accumulated utilisation values captured via the UNIX `top` command and from our tracing events, collected via resource usage system-call, `getrusage`, with total system utilisation values from `top`. The absolute difference between CPU utilisation of the processes involved with the communication subsystem and total CPU utilisation of the full system represents the amount of undetected behaviour. The figure depicts the undetected behaviour with small amounts of dispersion, of which 0.7% to 8.7% are outliers, indicating a matching behaviour throughout the execution time. Note that the behaviour captured, i.e., monitored, from the communication subsystem involves only 1/6th of all application processes. The blue box plot is based on a graph resulting from the absolute difference between full system CPU utilisation and observed CPU utilisation for every point in time, all from system parameters collected using `top`. It shows median values of 10.10%, 11.35%, 11.35% and 11.30%. The red box plot is generated similarly, with the only difference that the observed CPU utilisations are based on recorded communication events' data using `getrusage` and shows median values of 10.50%, 11.94%, 12.01% and 11.91%.

1 These workloads do not represent all possible workloads for photolithography machines. Nevertheless, these are sufficiently diverse for our experiments.

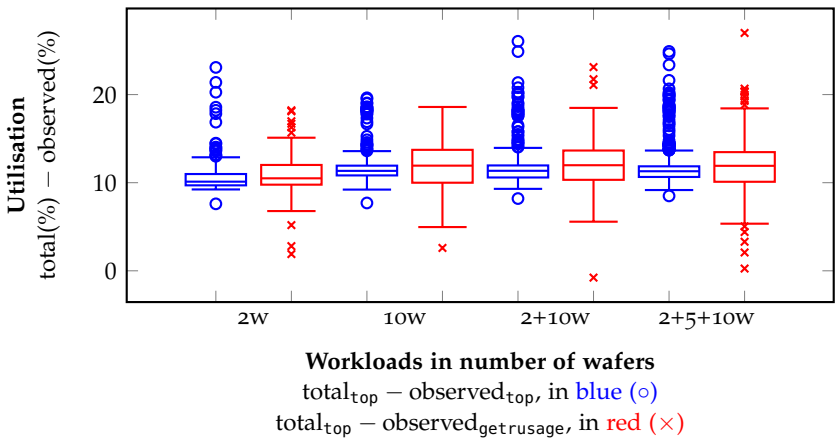


Figure 4.2: Plotting the behavioural coverage evaluation, using CPU utilisation differences for multiple workloads, i.e., by comparing total system vs the communication-centric view. Considering that we are monitoring 1/6th of the total application processes, we can see that the amount of observation loss is consistent for different workloads, while at the same time, the observation loss is rather close for both `top` and `getrusage`.

Now that we have evidence supporting the consistent behavioural view of our communication-centric monitoring, let us consider simulations based on these traces. Figure 4.3 shows the absolute CPU utilisation difference obtained considering the two different simulation policies explained in Section 3.5.1. Median values for Policy 1 are 0.0149%, 1.6439%, 1.0684% and 1.0697%, while for Policy 2, these values are 0.0146%, 1.6119%, 1.0232% and 1.0108%. These results are achieved considering load scenarios 2w, 10w, 2+10w and 2+5+10w, respectively. For both simulation policies, the absolute CPU utilisation difference ranges between 0.5% and 2.3%, showing that the simulated CPU utilisation closely tracks the monitored CPU utilisation. Actual CPU utilisation trends are kept confidential at the request of ASML and graphs depicting them cannot be presented.

As our third metric, aforementioned in the beginning of this section, Figure 4.4 depicts per-workload lifetime difference when simulating processes with the two previously covered event scheduling policies. The y-axis represents process lifetime difference for different workloads. The lifetime difference is calculated by subtracting the lifetime of an observed process from the input traces, with a simulated process from the output traces. Here, the output trace refers to the exact same way of collecting metrics, but from the execution performed on the digital twin. Ideally, the lifetime difference should be as small as possible, indicating low deviation. Separate box plots are drawn for our workloads

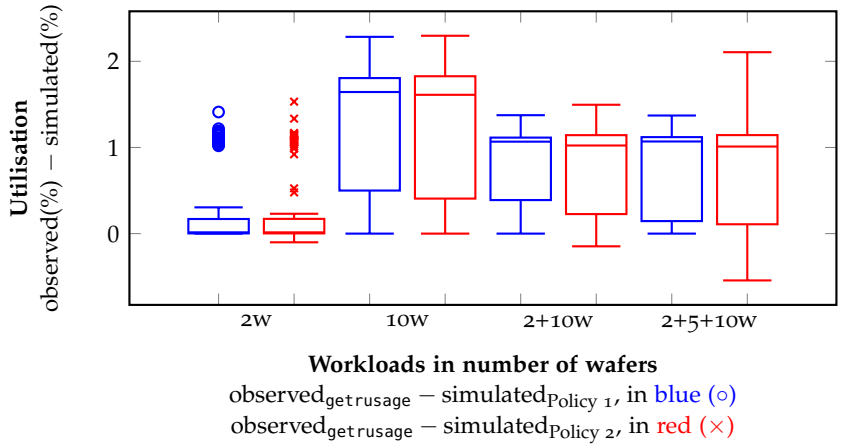


Figure 4.3: Plotting CPU utilisation differences for multiple workloads, observed vs simulated results, considering that the observed view is from our communication-centric monitoring using `getrusage`. We can observe that when it comes to CPU utilisation, simulations are rather close to what is collected via our partial monitoring view. Note the rather small scale of the y-axis.

in Figure 4.4. There are very few specific processes where the lifetime difference can be considerable, such as the 221% lifetime difference observed in one of the processes, when using Policy 2 for the 2-wafer workload experiment. These major differences are present when processes have very short lifetime or contain one small set of events to simulate, in some cases only one event. Policy 2 presents, in most cases, a higher process lifetime difference, due to the fact that events are not scheduled according to initialisation timestamps recorded in the traces, and only idleness of computation events is taken into account, as shown in Figure 3.4b. For the case of Policy 1, some differences are higher because the last scheduled event can be a computation event, where the idleness is not considered. This may considerably increase the process lifetime difference.

Even taking into account that some events have a high lifetime difference, the average total execution time difference, considering the two presented policies, ranges between -0.0007%[2] and 0. Simulating between 63 thousand and 480 thousand events, captured during 10 to 15 minutes of real machine execution, takes about 10 to 15 seconds. This achieved simulation performance is from a single core of an Intel® Xeon® Gold 6146, running at 3.20 GHz.

2 A negative value indicates underestimation.

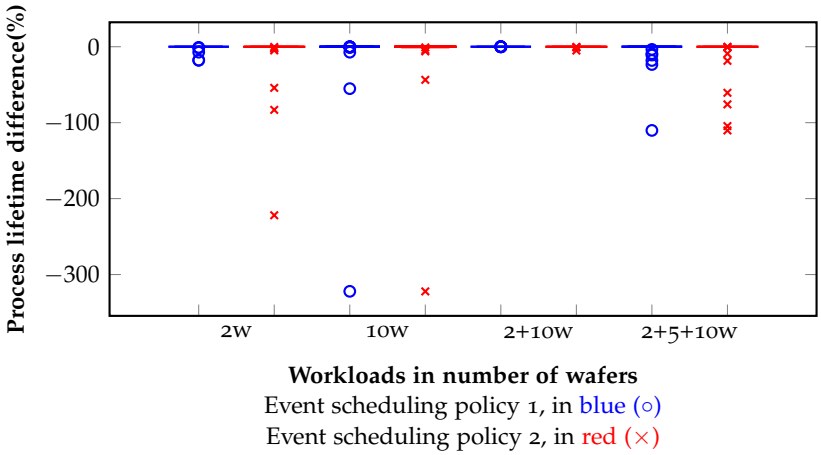


Figure 4.4: Plotting process lifetime differences for simulations of different workloads, considering both Policy 1 and Policy 2.

4.1.3 *Feature set and classification*

Having regression functions as representations of reference units of execution (phases) allows us to compare new sets of sample data against these functions using goodness-of-fit tests, depicted as the "Goodness-of-fit values" block under "Classic ML: Data set generation" segment in Figure 4.1. In our case, coefficient of determination (R^2) and Root-Mean-Square-Deviation ($RMSD$) are taken to quantify and compare the amount of deviation. The comparison considers the collected sample related to a specific input against the reference passport.

The classification algorithms we have are Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM)[3], with the first two having the best results. The feature set involved in the classification and brief descriptions of these features are as follows:

- Operation: Considered phase, i.e., wafer op.
- Metric: Considered metric, i.e., CPU time, cumulative reads (reads count) and cumulative writes (writes count) in our experiments for this use-case.
- $D_{\text{pid, phase, event}}(R^2)$: Percentage difference between the R^2 value and the R^2 value of the reference passport per process, i.e., PID,

3 The hyperparameters used with Scikit-learn library for each classifier besides the default values are as follows: $criterion = entropy$ for DT; $n_estimators = 100$ for RF; $n_jobs = -1$ for k-NN; $max_iter = 50000$ for LinearSVC; $kernel = poly$ and $degree = 5$ for KernelSVM.

and per event, e.g., for a single wafer op. phase, we will have *pid_event* columns such as, *23_compute*, *23_read* and *23_write* for PID 23, corresponding to deviations of computation, read and write behaviour, respectively.

- $D_{\text{pid, phase, event}}(RMSD)$: Percentage difference between the $RMSD$ value and the $RMSD$ value of the reference passport per PID and per event, e.g., for a single wafer op. phase, we will have *pid_event* columns such as, *23_compute*, *23_read* and *23_write* for PID 23, corresponding to deviations of computation, read and write behaviour, respectively. Note that if a PID is strictly a producer of data, it will not have a read column, or for the case of a strict consumer, a write column will not be present.

- Label: Normal, Anomaly 1, Anomaly 2, etc. (Persistent Benign, Persistent Harmful, Transient Benign and Transient Harmful for this use-case, as will be discussed in Section 4.3.1)

A total of 66 different processes (PIDs) are involved in this use-case's phase data. We are running the classification with different metric choices to compare which individual metric proves to train a more accurate classifier. We also do this for the combination of all three metrics. Though we have tried the percentage difference amounts for both R^2 and $RMSD$, the results included in this thesis for this use-case, are based on the percentage difference for $RMSD$ values.The results for different classifications are provided in Chapter 5. We do clean up the data set columns by removing all zero columns, which proves to result in slightly better classifications. We also split the data into 70% training and 30% test data. Our data analysis pipeline has been written in Python 3.7 and for our regression and classification needs, we rely on Scikit-learn 0.23.1 machine learning library [63].

4.2 POWER PASSPORTS WORKFLOW

Our set-up for a proof-of-concept, specific to the image analysis use-case, alongside the preprocessing, the feature extraction, labelled dataset generation and the classification pipelines are given in Figures 4.5 and 4.6, respectively. This set-up consists of an ODROID-XU4 computing device, implementing the ARM big.LITTLE computing architecture. We are running a stripped-down Linux distribution and the main running application is a neural network-based image analysis software, detecting if cars are present in images. The platform is capable of receiving images from either a camera, or a storage device and in our case, images are provided via a storage device. Note that for this use-case, the combination of the proof-of-concept set-up and the data processing pipeline has less complexity, mainly due to the absence of a digital twin,

i.e., a simulator. Here, there is no need for such a twin, for we are able to include real anomalous conditions, eliminating the requirement for synthesising anomalous traces. There will still be a need for a digital or a physical twin when it comes to evaluating actuation policies though.

For our electrical EFB data collection, we rely on the Otii Arc power data logger unit [65]. The data logger collects electric potential in Volts and electric current in Amps with the sampling rates of 1 kHz and 4 kHz, respectively. Timestamps for each data collection is also recorded alongside these metric values. As shown in Figure 4.5, data collection is followed by its compartmentalisation (cutting) and application of regression modelling next, resulting in power signatures/passports. The goodness-of-fit values are acquired by comparing a signature to a passport.

Given that we have electrical potential, electrical current and time readings, we consider the three metrics, *current* in Amps, *power* in Watts and *energy* in Milliwatt-hours, for generation of signatures. The image analysis running on the target platform has two main operations, i.e., reading images from a storage and applying a neural network detection algorithm on them. Thus, we have considered the following atomic and combo phases for current, power and energy readings:

- Image op.: An *atomic* phase for the image loading operation,
- Neural op.: An *atomic* phase for the neural network operation,
- Cycle op.: A *combo* phase for a full image cycle, including image loading and neural network operations.

As depicted in Figure 4.7, we compartmentalise the processing of an image into execution units, based on the mentioned phases. The parsing is followed by a cutting step, breaking the parsed data per phase.

4.2.1 *Mean passports*

In addition to individual power passports, we are also generating *mean power passports* as a unified representation for many reference executions with different input data. Mean passports are generated per metric and per phase type. When it comes to mean power passports, their generation is not a straightforward task, as there needs to be matching timestamps. Basically, we are calculating the mean of many regression functions, which can be written as,

$$y_{\text{mean}}(x) = \frac{f(x) + g(x) + h(x) + ...}{n},$$

with x being the independent variable, time, and n the number of functions.

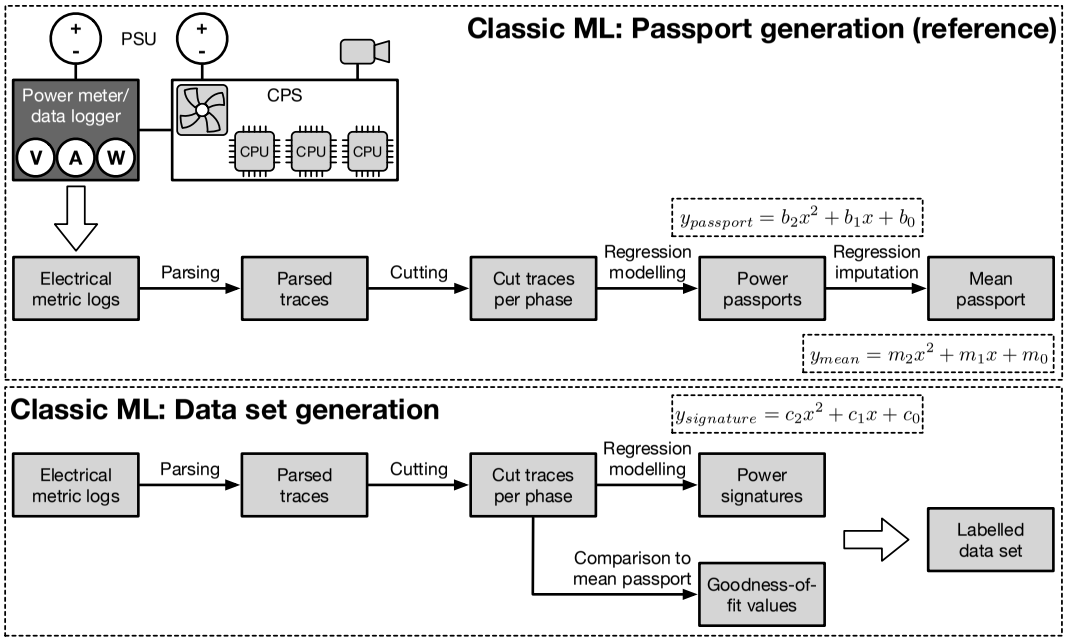


Figure 4.5: Our experimental platform, electrical EFB metric collection set-up and the data processing pipeline for power signatures and power passports are presented (note the independent power supply to the fan). This diagram elaborates the data set generation (preprocessing) flow from large amounts of trace data for the Classic ML approach, leading to a training data set.

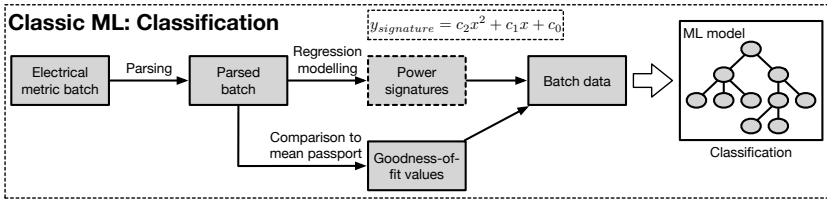


Figure 4.6: This diagram visualises the anomaly identification flow for the Classic ML approach, using much smaller trace batches with a previously trained classifier.

Either we have to do listwise deletion and remove independent variable readings which do not exist in all phases, or we have to perform data imputation. The latter is much more preferable, for there are executions that take slightly longer and we do not wish to disregard valid data collection points residing at the end of longer executions. In principle, this applies to other unmatched points as well, regardless of their location within the execution time frame. We have already generated regression models for data collections as their representations and as such, we can perform regression-based imputation by predicting the dependent variable values (metric) for missing independent variable values. This arrangement is especially convenient, since we already have generated and we use these very same regression models in comparisons and goodness-of-fit tests. There will be no extra bias other than what already exists as part of regression models. Figure 4.8 depicts this process.

Two example mean passports generated for the Image op. atomic phase and the Neural op. atomic phase are plotted in Figures 4.9a and 4.9b, respectively. These examples are based on the electrical current data as the EFB metric.

4.2.2 *Feature set and classification*

Similar to the previous use-case, coefficient of determination (R^2) and Root-Mean-Square-Deviation ($RMSD$) are taken to quantify and compare the amount of deviation. The comparison considers the collected sample related to a specific input against the universal mean passport. Obviously the metric and the phase should be the same. Therefore, there is no need for special considerations on the mean passport leg of the comparison, as there is only one per metric and per phase.

Same classification algorithms as the previous use-case are used, namely, DT, RF, GaussianNB, k-NN, LinearSVC and KernelSVM[4], with the first two classifiers, DT and RF, showing the best results. These

4 Same hyperparameters as the previous use-case were considered.

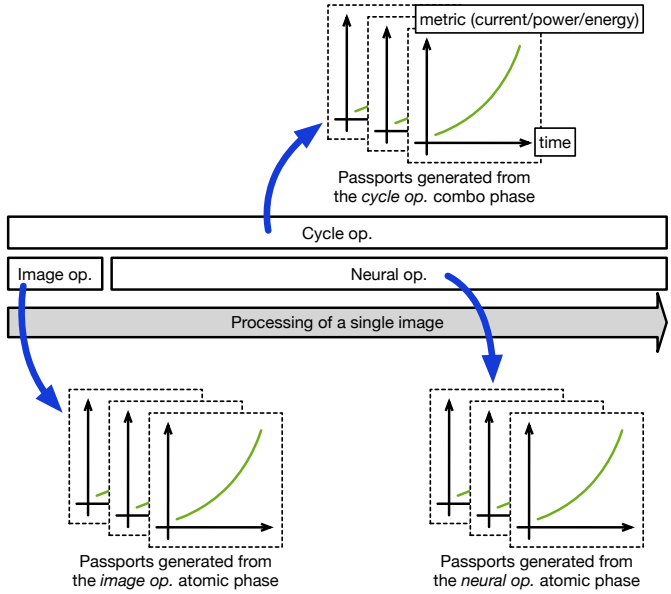


Figure 4.7: Different execution phases for an image processing task are drawn, i.e., the *atomic image operation* for loading of an image, the *atomic neural operation* for running the image through a neural network algorithm and the *combo cycle operation*, encompassing the first two atomic phases.

classification results are detailed in Chapter 5. The feature set involved in the classification and brief descriptions of these features are as follows:

- Operation: Considered phase out of image op., neural op. and cycle op.
- Core type: Utilised CPU core, big or little
- Metric: Considered metric out of current, power and energy consumption
- Execution time: Execution time for the phase (this is the one piece of information, turning the view into grey box, instead of black box, since we have the boundaries of phases in time)
- Coefficient 2: Coefficient for the second degree term, x^2, of the regression function
- Coefficient 1: Coefficient for the first degree term, x, of the regression function
- Intercept: Intercept value of the regression function

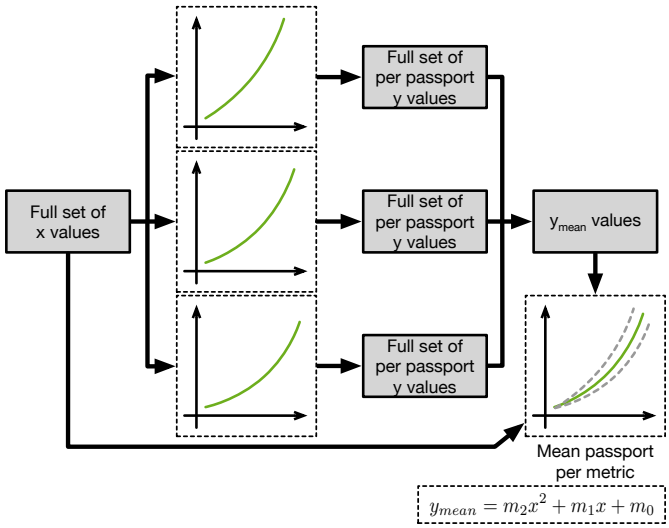


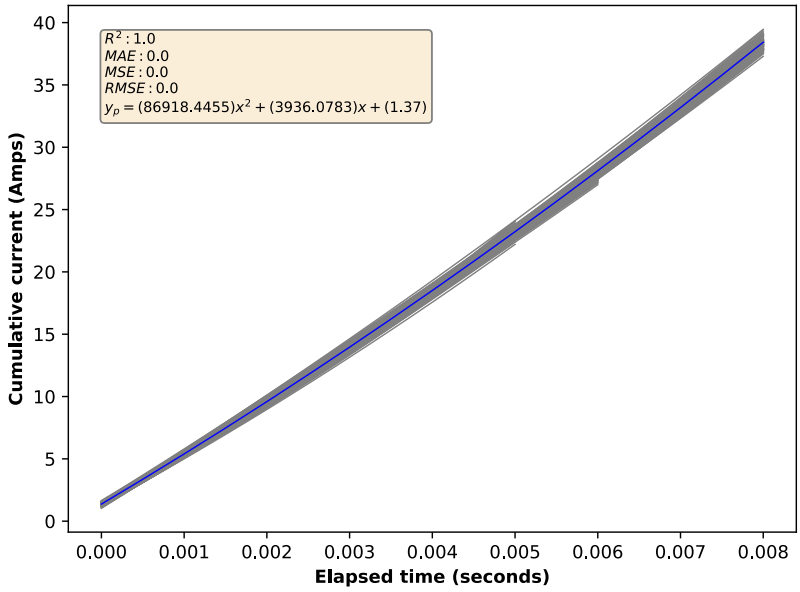
Figure 4.8: The diagram elaborating the mean passport generation flow is given. We are using the full set of x values (independent variable) from all available passports for the same metric and generating equal size sets of y values (dependent variable), by means of regression-based imputation.

- R^2: Goodness-of-fit value for sample points from one image against the reference mean passport
- $D_i(R^2)$: Absolute difference between the R^2 value and the R^2 value of the reference mean passport
- $RMSD$: Goodness-of-fit value for sample points from one image against the reference mean passport
- $D_i(RMSD)$: Absolute difference between the $RMSD$ value and the $RMSD$ value of the reference mean passport
- Label: Normal, Anomaly 1, Anomaly 2, etc. (NoFan and UnderVolt for this use-case, as will be discussed in Section 4.3.2)

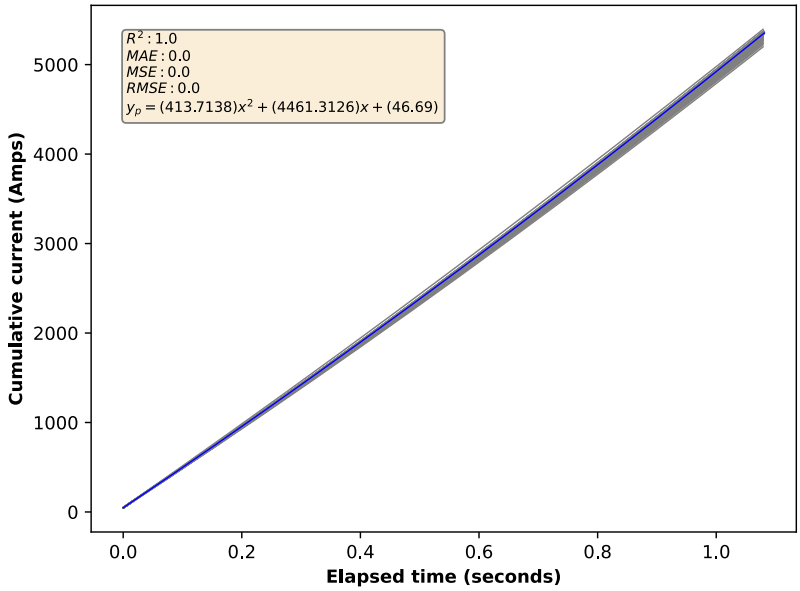
Here, we also split the data into 70% training and 30% test data. Note that the Operation, Core type and Metric columns are not actively used in classification, but are present to break the data set into subsets based on them. The results of trainings based on these subsets are provided in Chapter 5.

4.3 EXPERIMENTAL SET-UP

Let us go through the actual experiments performed for each use-case. Considering that we are following a data-centric approach, the main



(a) Mean passport for the Image op. atomic phase



(b) Mean passport for the Neural op. atomic phase

Figure 4.9: Example mean passport plots are provided for (a) Image operation atomic phase and for (b) Neural operation atomic phase. Grey curves indicate the passports that the mean passport in blue is based on. While the procedure is the same for functions with any degree, all regression functions in these examples are quadratic.

goal is to come up with multiple scenarios and relevant data sets to have enough variation for training and testing in our AI-based solution.

4.3.1 *Semiconductor photolithography machine: Synthesising anomalies*

In the set of experiments conducted with the ASML semiconductor photolithography machine, we have considered one execution phase corresponding to a repetitive task performed by the machine, wafer processing operation. Photolithography machines are always used for exposing batches of wafers with the same chip design, hence repetitive task. The simulation's output traces corresponding to the baseline execution were used to generate software passports for each process involved in the phase. In this fashion, the simulator, i.e., the digital twin of the system, is being considered as a reference platform. An example is provided in Figure 4.10. Here, Figure 4.10a depicts the cumulative CPU usage of one process during one phase (dots in red), alongside the polynomial regression that approximates its trend, i.e., the software passport. Similarly, Figure 4.10b provides a software signature, resulting from a synthesised anomaly.

In order to generate a proper data set containing different types of performance anomalies, we have composed several hiccup injection scenarios. The hiccup injection protocol, resulting in these scenarios is elaborated in the following steps:

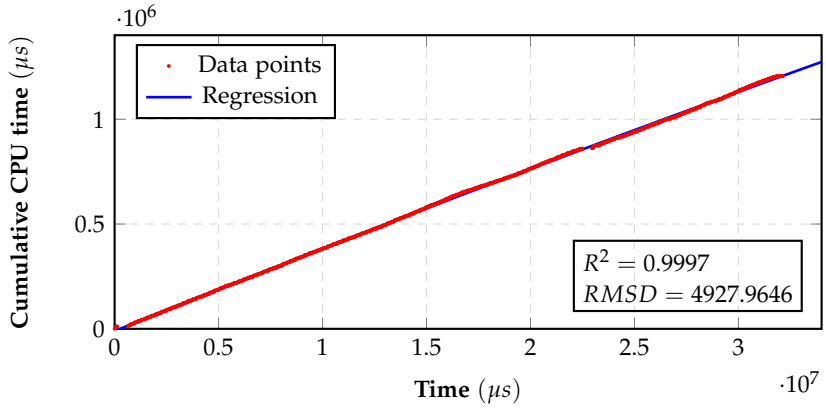
1. From the total number of available processes, 15%, 30% and 45% of them were selected randomly to introduce modifications in their traces.

2. From the processes selected in the previous step, two main performance anomaly groups, persistent and transient, were created. Software processes in the traces were randomly assigned to each group, while a 10% overlap between the two groups was ensured. For example, if we have the option to inject hiccups in twenty processes and we have to generate two groups with 10% overlap, the resulting groups could look like,

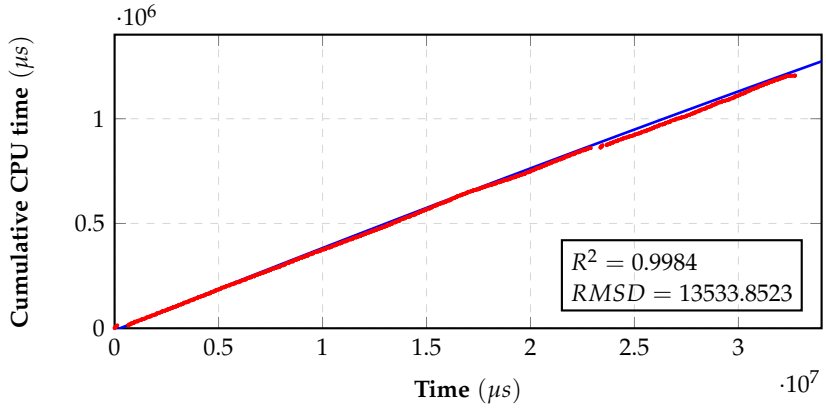
$$\text{Persistent group} = \{0, 1, 2, 5, 7, 8, 10, 11, 12, 13\} \text{ and}$$
$$\text{Transient group} = \{0, 3, 4, 6, 9, 14, 15, 16, 17, 18\}.$$

   The overlap percentage is configurable and its purpose is to create a more realistic scenario, where there is no clear separation between the set of processes that could cause persistent anomalies and the ones that could cause transient anomalies.

3. Computation, read, write, or all types of events are selected from the designated group of processes in the previous step. For each



(a) Cumulative CPU time software passport



(b) Cumulative CPU time software signature

Figure 4.10: An example plotting (a) the quadratic regression modelling result for a reference software passport, using cumulative CPU time as the EFB metric of choice, and (b) the corresponding violation after synthesising an anomaly. Aside from the quantifiable goodness-of-fit test results, here we can visually observe the deviation.

of the event types selected, either 10%, 15%, 30%, or 45% of them are modified, adding some overhead to the event's elapsed time.

4. The overhead applied to each event is either of 5%, 15%, 30%, or 45% of their total event elapsed time.

5. Lastly, the above steps were repeated five times to create sufficient data for different categories.

Using the protocol described above, we created 1920 different scenarios, where each scenario contains traces corresponding to the monitored software processes. These modified traces were simulated, using the

digital twin environment described in Section 4.1.2, to determine the impact on the phase execution time. Since our potentially anomalous execution scenarios and the relevant data are synthetically generated, we need an indicator to detect if the injected hiccup has turned the execution into an anomalous one. This indicator is the execution time of the phase. While considering the original phase execution time as the baseline, for the 0-2%, 2-4% and above 4% amounts of increase in the phase execution time, we consider the behaviour to be Normal, Benign and Harmful, respectively. Combined with the groups of processes that are used for hiccup injection, we will end up with the categories, *normal*, *persistent benign*, *persistent harmful*, *transient benign* and *transient harmful*.

4.3.2 *Image analysis platform: Diversifying scenarios*

The input data for the image analysis application are provided on a storage device. We have considered two different sets of images, first one being proper images with meaningful scenery. This batch includes images with and without a car depicted in them. The second batch includes images that do not depict any particular shape and have purely randomised pixels, introducing variation in the input. In this fashion, we could evaluate if the composition of an image is a factor for our workflow. Each batch is used in two different executions, one involving a single round of image analysis and the second, involving ten rounds of image analysis, meaning the same batch is processed ten times, sequentially. We have chosen the number of rounds arbitrarily and with the aim to have a long enough execution, reflecting the effects of anomalies. Each batch includes 30 images, making the workload for ten rounds of processing as 300 images. We have also executed every combination of conditions twice, by assigning the application to either a big, or a little core on the platform. The list of performed data collections are as follows:

- Case 1: 1 round of execution, regular images, little core (30 total images)
- Case 2: 10 rounds of execution, regular images, little core (300 total images)
- Case 3: 1 round of execution, regular images, big core (30 total images)
- Case 4: 10 rounds of execution, regular images, big core (300 total images)
- Case 5: 1 round of execution, randomised images, little core (30 total images)

- Case 6: 10 rounds of execution, randomised images, little core (300 total images)
- Case 7: 1 round of execution, randomised images, big core (30 total images)
- Case 8: 10 rounds of execution, randomised images, big core (300 total images)

The structuring of our workloads for the aforementioned data collection cases, fits the principle of repetitive task execution for industrial CPS rather well. Keep in mind that although these tasks are repetitive, but the underlying non-determinism is still present, as the behaviour of the system has subtle variations per execution, even with the same exact input. This could for instance result from environmental conditions, as these systems are installed on site and most often outdoors. Now that different cases are defined, we have considered two different anomalies, affecting the performance and the reliability of the system.

MALFUNCTION OF THE COOLING SYSTEM For this anomaly, henceforth called NoFan, we have disabled the cooling fan of the platform's CPU block. Keep in mind that in our set-up, the fan has a separate supply of power to begin with and will not directly affect electrical EFB metric readings of the platform, as depicted in Figure 4.5.

UNSTABLE POWER DELIVERY For this scenario, henceforth called UnderVolt, we have reduced the voltage supply to a level below the required amount for the platform, but still keeping the device functional. This was a reduction from 5.0 Volts to 4.7 Volts. We have also made sure that the voltage supply is at a sufficient level and it will not result in glitches during the execution of tasks.

As it was the arrangement for data collection under normal circumstances, reflecting the normal behaviour, we have considered the exact same cases with different batches of images, different numbers of processing rounds and different cores. Accordingly, our experiments resulted in eight cases for each scenario, representing Normal, NoFan and UnderVolt situations. It must be mentioned that having equal number of cases and basically equal number of data fields for all scenarios is advantageous as it will result in a balanced data set for classifiers. Having a balanced data set will eliminate the need for imbalance countering techniques, e.g., undersampling and oversampling.

4.4 ALTERNATIVE IDENTIFICATION APPROACH

So far we have elaborated our implementations for different use-cases, taking advantage of the Classic ML approach. Taking what we have

achieved into account, let us focus on the Advanced DL workflow and its elements, which was applied to the image analysis platform use-case.

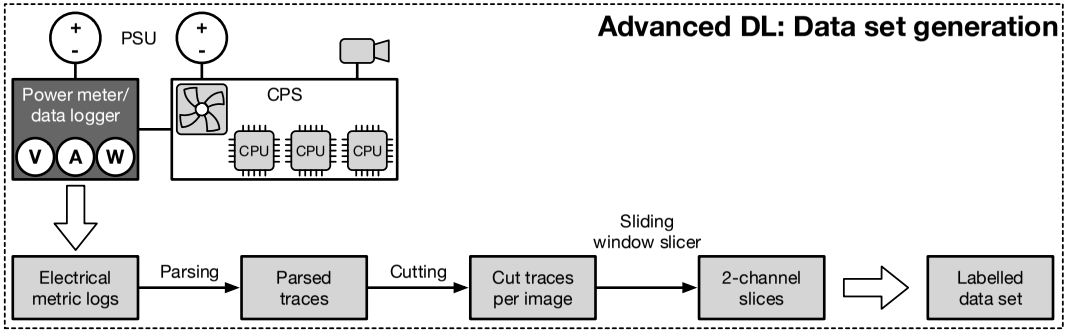
Our Advanced DL flows for data set generation and anomaly classification are depicted in Figures 4.11a and 4.11b, respectively. For both flows, whether the learning leading to the labelled data set, or the classification, the amount of data preparation is minimal. This preparation includes parsing of the raw metric logs, cutting of the parsed traces per image and running a sliding window algorithm to generate two-channel slices of fixed size. These two channels include the time data (timestamps) and the metric data (metric readings). It is necessary to consider the time data as a separate channel, since the metric data collection happens at high frequency. With non-deterministic system behaviour present, high frequency readings result in timestamps that do not exactly match for different experiments. This is an expected effect as industrial CPS are inherently non-deterministic. We have only considered the metric resulting in the highest accuracy for the Classic ML flow as it was seen in [58] and elaborated in Section 5.2, i.e., electrical current. Note that in this approach, there is no need for an intimate understanding of the data to reveal atomic phases within the processing of an image and the trace data related to each image is considered as a whole.

When it comes to time-series data, especially forecasting use-cases, Long Short-Term Memory (LSTM) networks are effective [29]. Although our observations come in the form of time-series data, they are individually isolated. Accordingly we have devised our Advanced DL workflow with CNN as its deep learning model architecture.

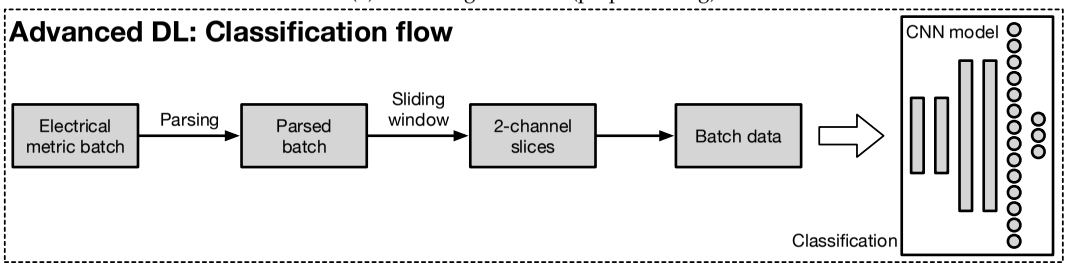
As we will see in Section 5.2, the Classic ML flow has a rather high accuracy [58]. To push the classification accuracy of our Advanced DL flow to similar levels, we have performed a grid search for hyperparameter optimisation. The three groups of considered hyperparameters are data preparation, learning and CNN model parameters, further elaborated in Section 4.4.2.

4.4.1 *Data set*

Our data set is generated from the same raw electrical metric readings used for the power passports workflow with the Classic ML approach, collected via an external power data logger unit, Otii Arc [65], connected to an ODROID-XU4 computing device. These traces are in the form of time-series and every data point has a timestamp and a metric value. The data set for the Classic ML flow has many columns, such as execution time, regression function coefficients and intercept, goodness-of-fit test values and labels [58]. The Advanced DL data set on the other hand is rather simple, only including two separate time and metric



(a) Data set generation (preprocessing)



(b) Anomaly classification

Figure 4.11: The two flows involved with the Advanced DL approach, (a) data set generation flow from large amounts of trace data, leading to a training data set, and (b) an anomaly identification flow, using much smaller trace batches with a previously trained classifier. Note the reduction in number of steps and the overall complexity of this approach compared to the Classic ML workflow.

data channels and corresponding labels. For both workflows, we are considering three labels, i.e., Normal, NoFan and UnderVolt. The two considered anomalies were described in Section 4.3.2.

The methodology can be implemented with any number of labels. Our demonstrator involves these labels corresponding to, normal circumstances for reference executions, faulty cooling fan for the system-on-chip, and unstable power supply, respectively. Both data sets are balanced as we have performed equal number of experiments for all scenarios (labels). For Advanced DL, as a common transformation, the data is being normalised at preprocessing. The normalisation is performed according to the formula

$$x_{\text{normalised}} = \frac{x - \mu}{\sigma},$$

where x and $x_{\text{normalised}}$ represent original the normalised values, while μ and σ denote mean and standard deviation of the population. Training set and test set ratios to the whole data set are 80% and 20% for the Advanced DL trainings, respectively and 70% and 30% for the Classic ML trainings, respectively.

4.4.2 *Convolutional neural network structure and search space*

The goal of our optimisation is to find the most accurate classifier. To arrive at an acceptable neural network model design, we have performed a grid search for the hyperparameter variations listed in Table 4.1. Hyperparameters belonging to three different categories are considered. The *Data preparation* category includes variations in slice sizes and the amount of shifting between slices, which are parameters for the sliding window algorithm during the data set generation process. The *Learning* category covers parameters related to the learning loop of the training algorithm and includes Learning Rate (LR) at the beginning of the training, number of epochs, batch sizes, presence of LR decay and if present, different decay periods. LR decay, if present, has a multiplicative factor of 0.1. The *CNN model* is the third category of parameters. Considered models are CNN models with two, four, or six convolutional layers and one Fully Connected (FC) layer.

Our model training implementation keeps track of training loss and training accuracy at the end of each epoch and saves the most accurate model. The best model is not necessarily the one trained at the last epoch. The most optimised model we arrived at consists of six convolutional layers with sizes 64, 64, 128, 128, 256, 256, a FC layer of size 4096, with all kernel sizes being 5×1, with Rectifier Linear Unit (ReLU) activation for each convolutional layer and the FC layer, and with MaxPool layers after even convolutional layers, as visualised in Figure 4.12.

Table 4.1: Three categories of hyperparameters considered during the grid search to find the most accurate CNN model, alongside their considered variations during the search are provided.

Parameter type	Parameter	Variations
Data preparation	Slice sizes	50, 100
	Slice shifts	10, 20
Learning	LR at start	0.01, 0.001, 0.0001, 0.00005
	Epochs	10, 20, 30, 40, 45, 50, 60
	Batch sizes	10, 20, 50, 100
	LR decay	present (mul. factor 0.1), absent
	Decay periods	8, 10, 20
CNN model	Conv. layers	2, 4, 6
	Conv. layer size	8, 16, 32, 64, 128
	Kernel size	3, 5
	FC layer size	512, 1024, 2048, 4096

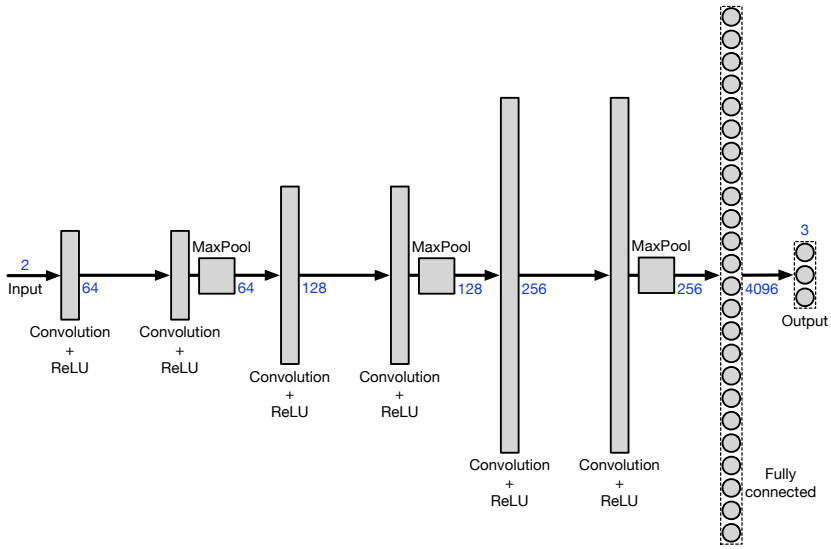


Figure 4.12: The composition of the CNN model with the highest achieved accuracy through our grid search is depicted. The depth, i.e., number of channels per input, different layers and output are given in blue colour.

Our data analysis pipelines have been written in Python 3.8 and we use the Scikit-learn 0.23.2 package for regression and the Classical ML classification, as well as the PyTorch 1.6.0 package for CNN implementations and training. The hardware infrastructure for our experiments is a machine with a 2.20 GHz Intel® Xeon® E5-2650 v4 CPU, 64 GB of RAM and a GeForce RTX 2080 Ti graphics card, with CUDA release 10.0, v10.0.130.

5 CLASSIFICATION RESULTS

The following includes our classification results, i.e., anomaly identification, alongside efforts leading up to model improvements. Discussions on the achieved results, comparisons and the implications deduced from the results are also included.

The contents of this chapter are mainly based on, but not limited to, the previously published conference and/or journal publications of the author. The publications of interest for Chapter 5 are:

- "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
- "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
- "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
- "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Our results mostly focus on performance and physical anomaly prediction accuracies of various classification algorithms, depending on the use-case. We provide these separately for our two industrial use-cases. Accurate classification, as the ultimate goal of the workflow from Figure 3.1, can be achieved with the right choice of metric, execution phase and a suitable classification algorithm.

As described back in Chapter 4, we are collecting EFB metric traces while applying predefined workloads to our platforms to collect and process this data and build training data sets for supervised learning of our classifiers. For our first use-case, i.e., the semiconductor photolithography machine, we take traces related to a wafer's processing operation (*wafer op.*) as our reference, which in turn leads to synthetic anomalous trace generation through the platform's digital twin implementation. For this first use-case, the considered anomalies, defined based on the amount of invoked drift for the total phase execution time,

are *persistent benign*, *persistent harmful*, *transient benign* and *transient harmful*.

With the second use-case, the image analysis platform, all traces, normal and anomalous, come from real processing of different batches of images as workloads, in combination with different platform settings. Here, there are two anomalies present, NoFan and UnderVolt, which are applied directly to the platform. We have also considered three different phases, *image op.*, *neural op.* and *cycle op.*, as well as the combination of the first two.

5.1 SEMICONDUCTOR PHOTOLITHOGRAPHY MACHINE

We have tried Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM) classifiers with the data set representing a single wafer's processing, alongside CPU time, reads count and writes count, as our metrics of choice, as well as the combination of all three. As such, Table 5.1 presents overall prediction accuracies for different classifiers.

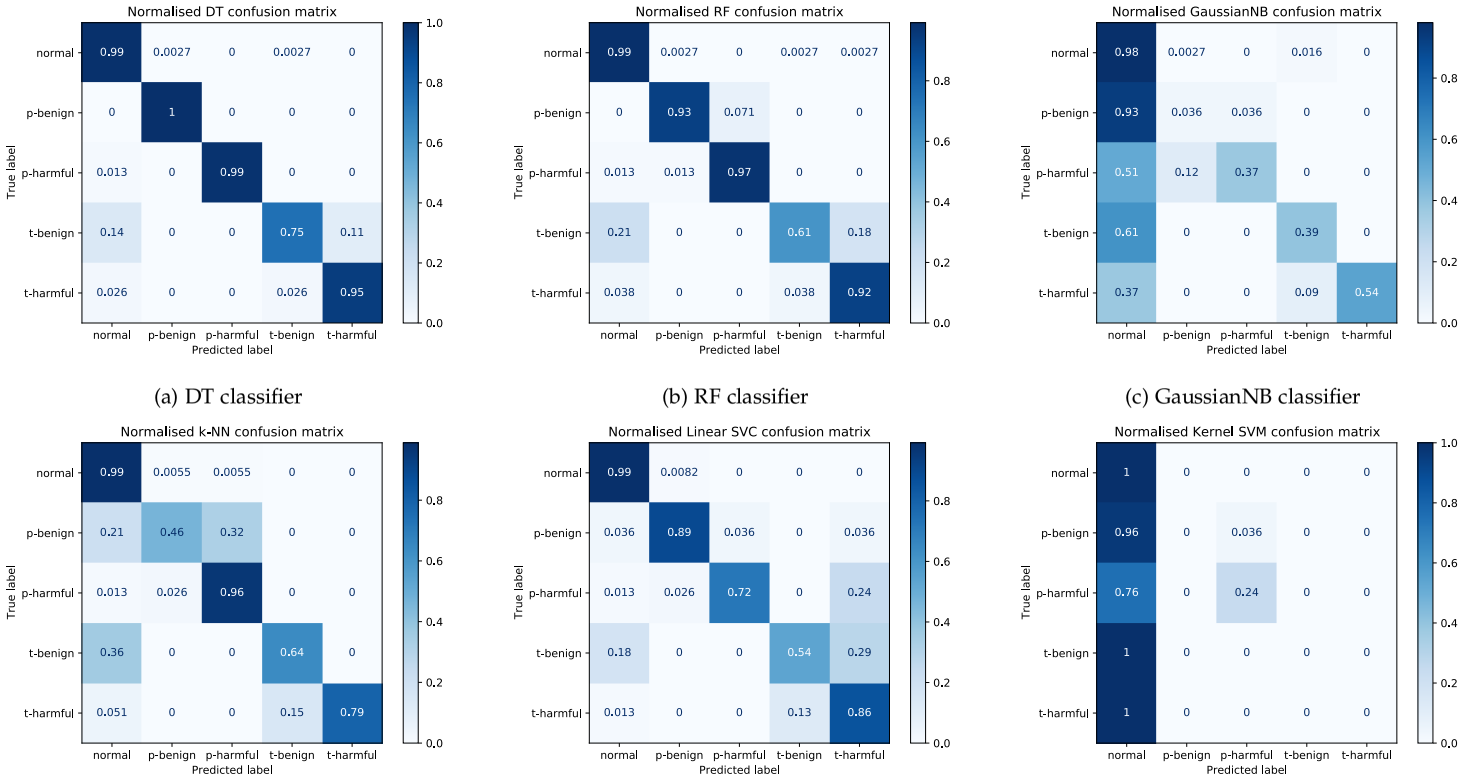
The overall prediction accuracy of DT, RF, GaussianNB, k-NN, LinearSVC and KernelSVM classifiers, while using the data generated from *single wafer processing* phases together with the combination of all three metrics, are 97.56%, 95.83%, 76.38%, 91.66%, 90.97% and 66.49%, respectively. The confusion matrices for all classifiers with this combination are depicted in Figure 5.1, showing the higher prediction performance for DT and RF. Confusion matrices are efficient visualisations to demonstrate the capability of classifiers in prediction of labels from the rest of the data. In these matrices, *predicted labels*, generated by the classifier from feature columns are compared against *true labels* from the data set, on a per case basis. An ideal classifier with 100% prediction accuracy will result in a confusion matrix, showing 1 on all diagonal cells, i.e., every single label's prediction will 100% match the initially designated label from the data set.

As a result of randomness in our synthetic delay injection and scenario generation, the data set has considerable imbalance in favour of normal cases, leading to different accuracies for different label predictions. With an increased number of scenarios representing less frequent labels, relevant accuracies will improve. This is demonstrated in the image analysis use-case.

Considering the results from different classifiers, what stands out is the poor accuracy of KernelSVM. This is partly a result of poor hyperparameter tuning, as we mostly consider default settings out of the box for these classifiers. Any classifier will achieve its best with the right set of hyperparameters, which also depend on the data set and the problem. The major reason though behind different per label

Table 5.1: Overall prediction accuracy percentage of Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM) classifiers based on the data sets generated from different individual metrics, as well as their combination. The highest accuracies belong to DT and RF, when all three metrics, i.e., CPU time, reads count and writes count, are considered.

Metric	**Phase**	**DT**	**RF**	**GaussianNB**	**k-NN**	**LinearSVC**	**KernelSVM**
CPU time	wafer op.	96.00	95.65	79.51	92.18	90.45	66.49
reads count	wafer op.	92.01	95.48	32.98	93.57	90.79	75.69
writes count	wafer op.	94.96	95.65	76.90	92.18	86.45	66.49
all three	wafer op.	97.56	95.83	76.38	91.66	90.97	66.49



(a) DT classifier

(b) RF classifier

(c) GaussianNB classifier

(d) k-NN classifier

(e) LinearSVC classifier

(f) KernelSVM classifier

Figure 5.1: Depicted are normalised confusion matrices for (a) DT, (b) RF, (c) GaussianNB, (d) k-NN, (e) LinearSVC and (f) KernelSVM classifiers, considering the anomaly categories, Normal, Persistent Benign (p-benign), Persistent Harmful (p-harmful), Transient Benign (t-benign) and Transient Harmful (t-harmful) as labels, with the data set based on the combination of three metrics, i.e., CPU time, reads count and writes count, with the wafer processing phase. Best accuracies belong to DT and RF classifiers.

prediction accuracies, which is most apparent for the Kernel SVM, is fuelled by the data set's imbalance. Since there are more cases present for the *normal* label present, the KernelSVM classifier is mistakenly predicting many cases belonging to all other labels as *normal*. Focusing on DT and RF, given that the predictions are rather accurate despite the imbalance, we did not pursue improvement techniques for the data set any further.

5.2 IMAGE ANALYSIS PLATFORM

For this use-case, we have also tried DT, RF, GaussianNB, k-NN, LinearSVC and KernelSVM classifiers with different subsets of our data set. We observe that the choice of phase and metric as sources of data has a considerable effect on the prediction accuracy of the classification. We have tried classifications using data from all three metrics at the same time, as well as every metric individually. For these trials, we have considered either atomic image op., atomic neural op., combo cycle op., or the combination of the data from both atomic image and atomic neural operations. Table 5.2 presents these choices alongside their resulting overall prediction accuracies.

The overall prediction accuracy of DT, RF, GaussianNB, k-NN, LinearSVC and KernelSVM classifiers, while using the data generated from *combo cycle op.* phases together with the *electrical current* metric, are 99.15%, 99.23%, 90.09%, 98.22%, 86.45% and 89.92%, respectively. Here, DT and RF show the highest accuracies out of all combinations. The confusion matrices for all classifiers with this combination are depicted in Figure 5.2, again showing the higher prediction performance for DT and RF classifiers. High prediction accuracy is also observable across the board for all labels in all classifiers, as a result of having a balanced data set.

5.3 CLASSIC ML VS ADVANCED DL

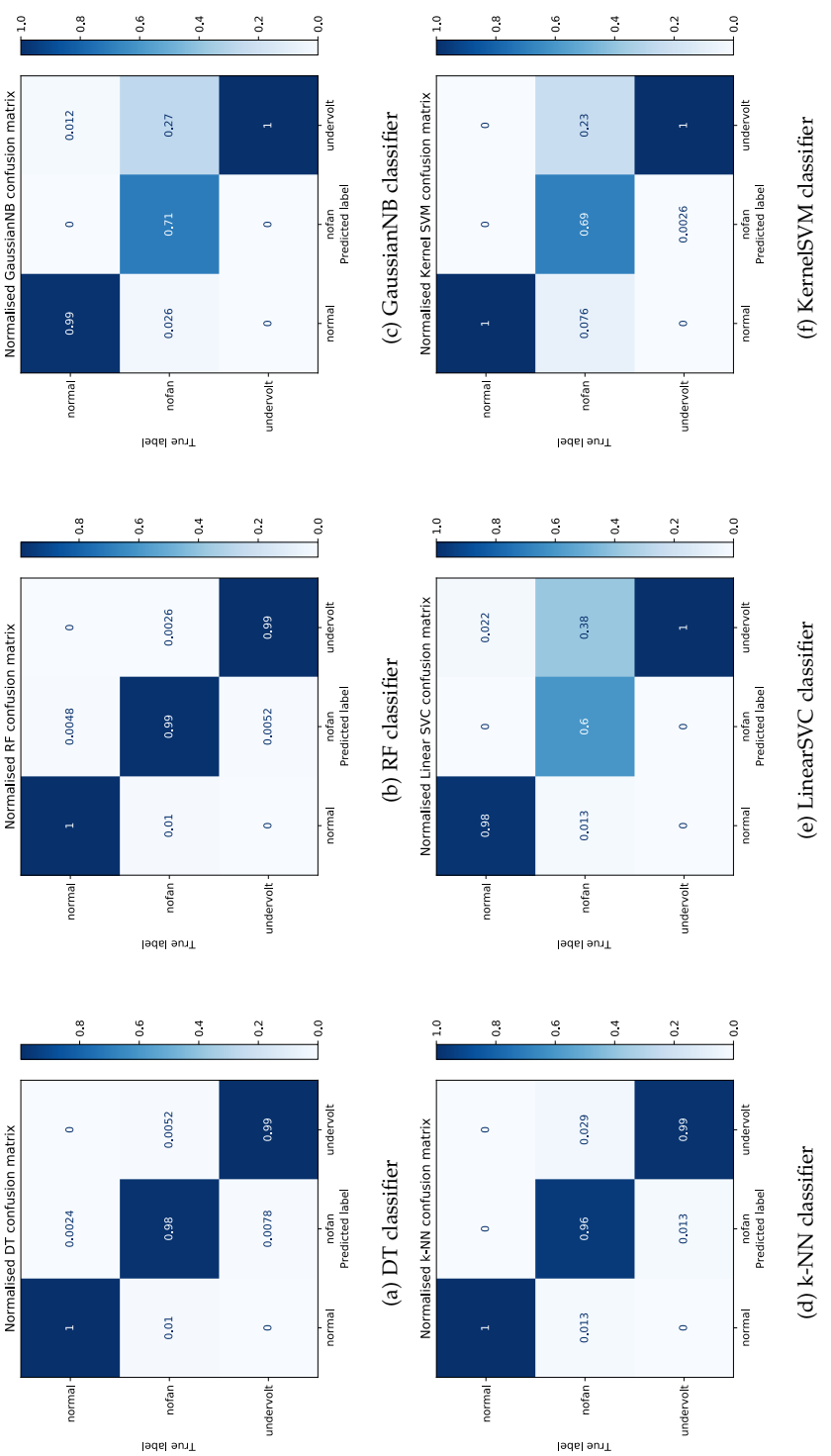
Considering the hyperparameters listed in Table 4.1, Figure 5.3 displays an overview of our grid search for paths achieving higher accuracies. Here, different paths are composed of different sets of choices for available hyperparameters. For instance, we can see that in the absence of LR decay, i.e.,

$$\text{Decay step} = 0,$$

changing other hyperparameters can only take us so far, with accuracies staying below 93%. We can also observe that the improvements on the lower end of the accuracy spectrum are larger compared to the upper end, meaning that as the model improves, new additional

Table 5.2: Overall prediction accuracy percentage of Decision Tree (DT), Random Forest (RF), Gaussian Naïve Bayes (GaussianNB), k-Nearest Neighbours (k-NN), Linear Support Vector Classification (LinearSVC) and Kernel Support Vector Machine (KernelSVM) classifiers, based on the data sets generated from different combinations of metrics and phases, are listed. The highest accuracies belong to RF and DT with the combination of current metric and the cycle op. phase.

Metric	Phase	DT	RF	GaussianNB	k-NN	LinearSVC	KernelSVM
current	image op.	89.83	92.80	67.40	69.34	38.86	39.28
current	neural op.	98.89	98.81	90.09	98.13	91.95	90.51
current	cycle op.	99.15	99.23	90.09	98.22	86.45	89.92
current	image op. + neural op.	94.23	96.06	57.05	83.94	61.79	38.03
power	image op.	81.79	83.23	60.71	62.06	36.15	37.25
power	neural op.	96.86	96.61	84.25	93.98	70.70	70.78
power	cycle op.	97.03	97.54	87.29	93.22	62.99	70.95
power	image op. + neural op.	89.70	91.48	60.10	80.26	59.16	36.00
energy	image op.	74.25	77.13	66.04	59.69	69.85	38.01
energy	neural op.	96.69	97.37	82.21	94.83	87.89	89.33
energy	cycle op.	97.29	97.71	83.06	95.25	88.56	89.50
energy	image op. + neural op.	86.99	88.47	68.91	77.72	78.90	37.06
all three	image op.	83.05	84.89	43.91	62.55	45.41	35.18
all three	neural op.	97.85	98.22	64.92	96.27	61.14	51.36
all three	cycle op.	97.96	98.50	66.05	96.07	68.68	51.42
all three	image op. + neural op.	89.86	91.52	49.39	79.51	57.56	35.39



(a) DT classifier (b) RF classifier (c) GaussianNB classifier

(d) k-NN classifier (e) LinearSVC classifier (f) KernelSVM classifier

Figure 5.2: Depicted are normalised confusion matrices for (a) DT, (b) RF, (c) GaussianNB, (d) k-NN, (e) LinearSVC and (f) KernelSVM classifiers, considering the anomaly categories, Normal, NoFan and UnderVolt as labels, with the data set put together for the electrical current metric and for the cycle op. phase.

improvements are harder to achieve. We have not plotted the paths with poor results to keep the plot clean and legible. It is observable that we realised a decay step of 10 to be the most optimal, which is why it is part of so many paths. The top results for the most part include a decay step of 10.

5.3.1 *Quantitative comparison*

To be able to quantitatively compare the two workflows, we consider *elapsed time* values for different operations, collected with the `time.perf_counter()` call, providing a high-accuracy monotonic clock. Timing results are given in Table 5.3. We are considering the elapsed times for *preprocessing*, i.e., flows in Figures 4.5 and 4.11a, *model training* and *model validation*. Model training has to be considered for our industrial use-cases, since upon the introduction of a new anomaly, i.e., a new label, retraining will be necessary.

Table 5.3: Elapsed times in seconds during different stages of the Classic ML and the Advanced DL workflows are compared. We can observe how the Classic ML workflow is rather fast when it comes to training and validation. On the other hand, the much faster preprocessing and the availability of GPU acceleration for the Advanced DL workflow stands out.

Workflow	Preprocessing	Training	Validation
Classic ML-DT (CPU)	204 648 (∼57h)	0.02	0.001
Classic ML-RF (CPU)	204 648 (∼57h)	0.32	0.021
Advanced DL (CPU)	2576	239 976 (∼67h)	125
Advanced DL (GPU)	2576	18 535 (∼5h)	25.5

We have timed the Classic ML model training and model validation for both DT and RF classifiers, while with regards to the Advanced DL, we have considered CPU and GPU implementations. An important limitation to point out about the Classic ML preprocessing from Figure 4.5 is the dependence of data set generation on the calculation of mean passports. Although this preprocessing is highly parallelised, the aforementioned dependency means that the top reference flow from the figure has to succeed first.

Beyond what we can accurately measure, two rather time-consuming titles are missing from Table 5.3, which are:

- Feature engineering effort: The human effort behind the Classic ML workflow design involving the extensive study and analysis of the operational specifics of the system under scrutiny.

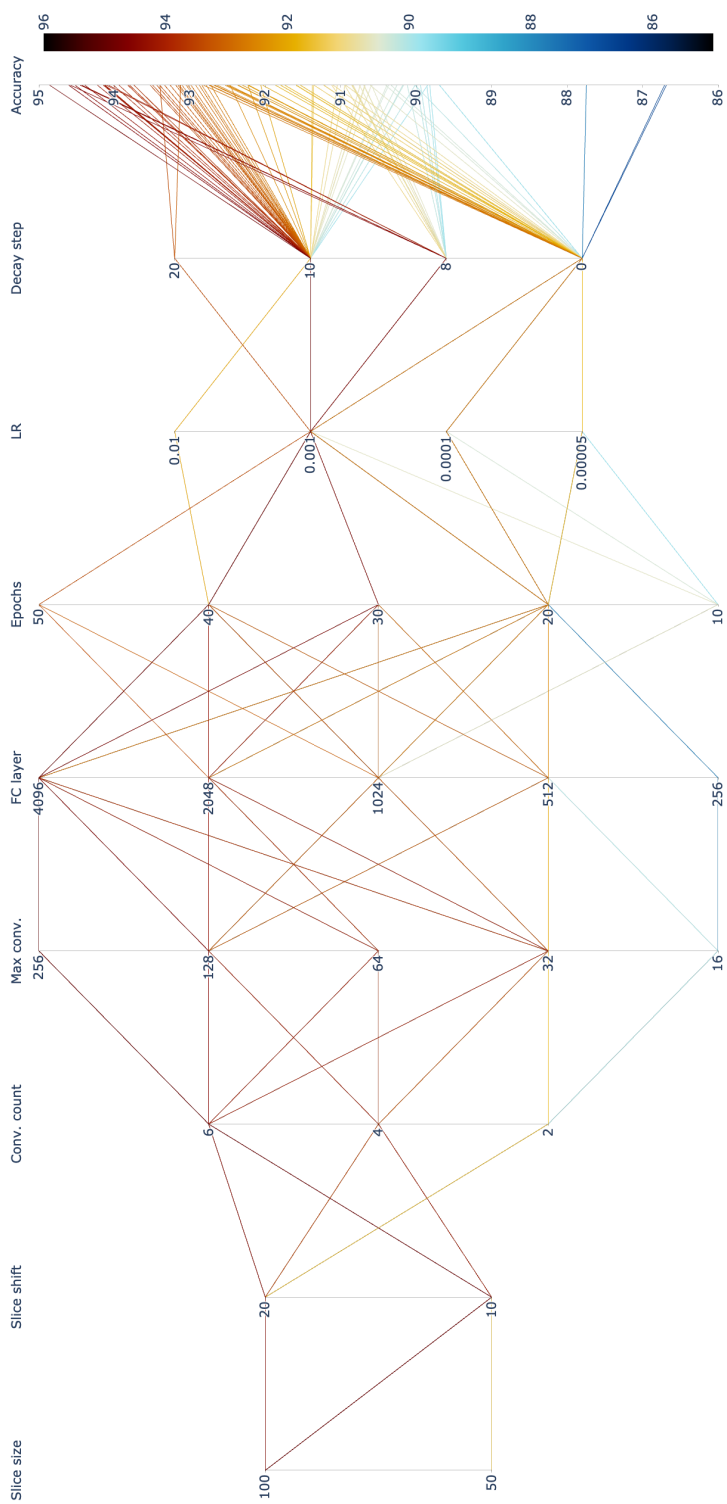


Figure 5.3: The set of highest accuracies resulting from our grid search over different hyperparameter choices are visualised in a parallel coordinates plot. In the above plot, *Slice size* stands for the input slice size from our sliding window implementation, *Slice shift* stands for the amount of shifting over data between consecutive slices, *Conv. count* stands for convolutional layer count, *Max conv.* stands for the width of the broadest convolutional layer, *FC layer* stands for the fully connected layer size, *Epochs* stands for the number of training epochs, *LR* stands for the applied learning rate, *Decay step* stands for the epoch intervals before a decay application on the learning rate with o denoting the absence of decay, and *Accuracy* stands for the achieved model prediction accuracy.

- CNN model optimisation effort: The human effort behind the semi-automated search for effective hyperparameter combination discovery, or design of a fully automatic search, which is necessary for the Advanced DL workflow.

5.3.2 *Qualitative comparison*

It is evident from our observations during the CNN training that there is a limit to the Advanced DL workflow's achievable accuracy. This is considering the fact that minimal amount of preprocessing has been applied for this particular workflow on purpose. We also see that this achievable accuracy is an effective one, up to 94.85%, depending on hyperparameters. The accuracies for our Classic ML workflow using DT and RF classifiers are 99.15% and 99.23%, respectively. However, the high accuracy provided by the Classic ML workflow comes at a cost and arguably, a high one. The amount of analysis, design effort, experimentation and in short, *feature engineering effort* required for the Classic ML workflow is rather vast. Accordingly, there is much need for domain specific knowledge and understanding of the internals of the system under scrutiny. The workflow designer has to know beforehand, or explore, to understand which phases best reflect the overall behaviour of the system for the specific set of anomalies.

One of the capabilities missing in our Advanced DL workflow is the possibility to detect unknown behaviour, i.e., unseen anomalies. Though the CNN model itself can be retrained upon the addition of a new anomaly, the workflow does not include steps facilitating new anomaly discoveries. The reference methodology from Figure 3.1 provisions this possibility, for we can use goodness-of-fit tests and detect unseen levels of deviation from a passport. Following this detection, further analysis will result in a new class of anomaly, which can be added and considered for feature engineering in future data sets. This addition of unknown anomalies is achievable in the Classic ML workflow. However, it does require the designer to go through the whole process again, as the new anomaly may or may not be easily detectable using the same phase data.

We would like to emphasise the fact that our Advanced DL workflow is a truly black box approach, requiring no insight into the data or the system internals. In this fashion, the Advanced DL flow cuts through the data processing complexities of the Classical ML flow. Though optimising hyperparameters is a time-consuming process, it does not depend on the internals of the system and is reusable in the future for more anomalies. We just have to retrain the network. On top of that, neural network frameworks are highly optimised for GPU acceleration, requiring minimal changes to the implementation code.

Stability and maturity of frameworks, in the sense that how much code transformation is enforced from one version to the next is another aspect. In our experience, the change is rapid and substantial with deep learning frameworks, as the field is constantly changing and evolving. This could very well be a factor in a business environment striving for long-term deployment.

Last but not least, with Classic ML workflow, we are able to explain why the classification has resulted in a certain label. Models such as decision trees can be traversed and every processing block in the Classic ML flow can be backtracked to initial trace values, directly connecting the outcome to the input. For instance, the data resulting in a specific regression, or the sample data compared to a specific passport, is known to us. The analytical capability provided by this backtracking is quite powerful and will allow us to improve the method. It will allow us, for example, to fine-tune our data set and detect few incorrectly classified corner cases. We will explain this further in Section 5.4.1.

5.4 DISCUSSION

On the one hand, with the Classic ML workflow, our methodology's outcome is based on feature engineering and traditional classification algorithms. While the methodology itself is applicable to any industrial CPS, implementations are use-case specific. Accordingly, it is important to fine-tune the workflow based on the platform at hand and its characteristics. Such a fine-tuning is necessary to maximise classification accuracy and bring out the full potential of the method. Fine-tuning in this context is the process of selecting the best combination of EFB metrics, execution phases and classification algorithms. While EFB metrics and execution phases of choice are highly use-case dependent, we can see that DT and RF classifiers are the best performing ones for the Classic ML workflow. In the case of DT classifier, a simple but powerful analysis is to visualise the DT graph. The graph can guide us to choose the right execution phase, for the more compact and contained the graph, the easier it is for the classifier to distinguish between labels. For our use-cases, we see that the choice of three combined metrics (CPU time, reads count and writes count) and the wafer processing phase, as well as the choice of electrical current as the metric and the cycle operation phase, lead to rather compact DT graphs. As can be deduced, there is no pattern with regards to the number of metrics to be considered. The combination of these steps are the most time-consuming task when putting together a Classic ML workflow. These graphs are shown in Figures 5.4 and 5.5 for the semiconductor photolithography machine and image analysis platform use-cases, respectively.

On the other hand, with the Advanced DL workflow, there is a sharp decrease in such tasks. As the workflow bypasses the need for

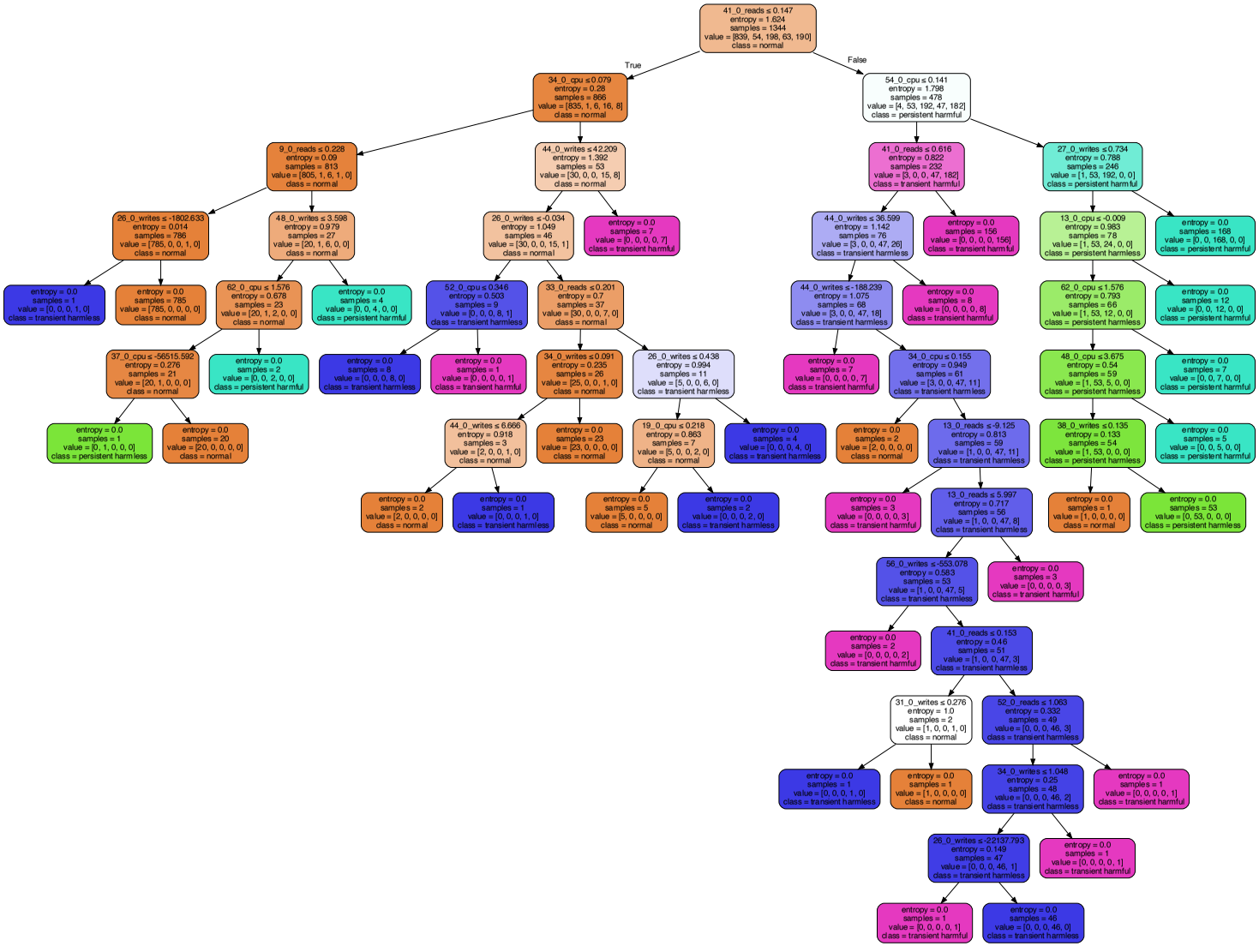


Figure 5.4: Showcases the compactness of the Decision Tree (DT) graph for the semiconductor photolithography machine use-case, using the Classic ML workflow and considering all three metrics, during the wafer op. phase.

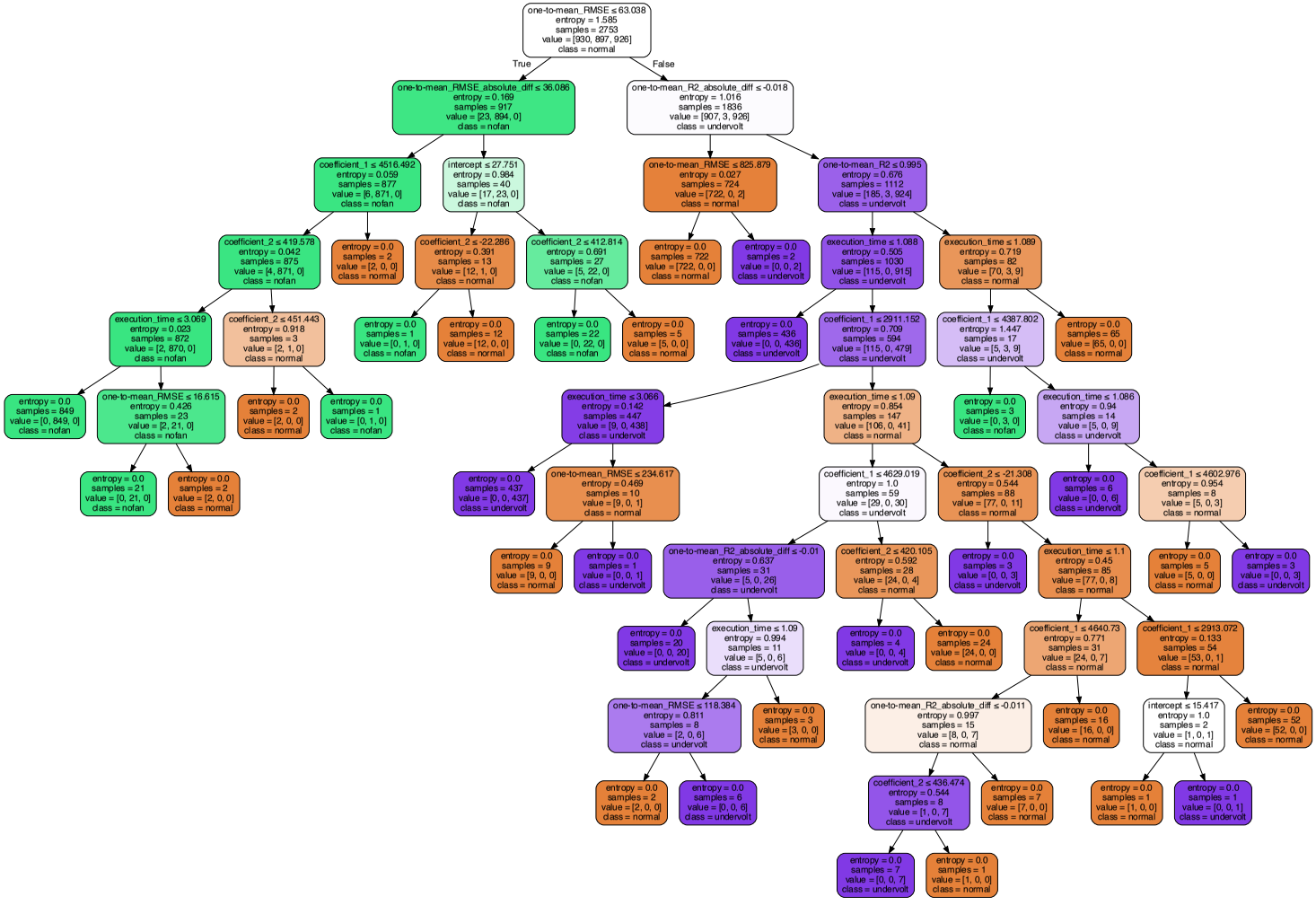


Figure 5.5: Showcases the compactness of the Decision Tree (DT) graph for the image analysis platform use-case, using the Classic ML workflow and considering current as the metric, during the cycle op. phase.

signatures and passports, an analysis towards the selection of the best performing phase data will not be required. The DL algorithm will take care of that for us. Even with the effort attached to finding the best CNN model, it is more of a computational effort than engineering. There are rather mature software frameworks available that are capable of parallelised training of different model variations. While we have not taken advantage of it, the process of hyperparameter tuning could also be automated.

One might argue that anomaly specific monitoring is also capable of detection, especially for metrics external to the system, e.g., voltage supply. While this is a perfectly valid proposition to detect voltage supply instabilities, such detection lacks versatility, as it is single anomaly specific and assumes prior knowledge of the anomaly type. In comparison, our methodology can detect and differentiate between multiple anomalies. This versatility is a direct result of data-centricity and comparisons based on behavioural signatures. Our methodology also foresees detection of unknown anomalies through described comparison tools, namely, goodness-of-fit values.

Yet another interesting observation is the required amount of data. It is known that deep learning algorithms perform beyond the traditional ML algorithms do as the amount of data grows and vice versa [10]. We mentioned that in our semiconductor photolithography machine use-case, the trace data is representative of 66 processes. We also showed in Table 5.1 the different choices of metrics and the resulting impact on classifier accuracies, which arguably was not so big. However, what is important to mention is that pruning the data set based on single metric choices, dramatically reduces its size. For instance, not every process is a producer of data (invoking write events) or a consumer of data (invoking read events). As a result, the number of feature columns we end up with per metric are 29, 8, 22 and 59, for CPU time, reads count, writes count and all three combined, respectively. Considering that feature columns are also per process designations, their count indicates the number of processes providing the data for the selected metric. Although for the reads count metric we end up with just 8 feature columns, but the classifier accuracy does not diminish that much.

Accordingly, there are two points to mention here. First, it is interesting to see that the choice of right metric, e.g., reads count, could dramatically reduce the size of the relevant data set, without imposing a big impact on the prediction accuracy. This could be considered as a measure that makes our data collection and retention procedures more efficient. In this case, it is also not entirely surprising, as data generated by the producers, eventually end up with the consumers. The behavioural changes from anomalies are either directly imposed on consumers, or indirectly reflected on them, as changes in the behaviour

of producers will eventually have an impact on consumers too. The second point is that these results should be interpreted with caution, as our experiments for the first use-case are based on a mock-up, i.e., the digital twin. The anomalies are also synthetically generated. such limitations in our experiments prevent us from drawing generalised conclusions for this use-case. Further studies in this direction can be considered as future work.

Thinking about this from an information position perspective, being able to differentiate between behavioural patterns based on minimum amount of information, i.e., from a poor information position, is good news. However, the very choosing of the correct source metric that is definitive in differentiation, but also leads to a compact data set, in itself, is the result of a white box analytical process. We should not consider such a dilemma as a hindrance though, for we have DL algorithms such as CNN, to pick the right features for us and with the Advanced DL workflow we have demonstrated how this would work.

5.4.1 *Explainable output*

While treating the system under scrutiny as a black box can be advantageous and a requirement in certain use-cases, having a black box solution, i.e., methodology, might not be so favourable. It is important to be able to figure out which changes at every different step of the methodology will lead to a certain result. Since our methodology, as mentioned in Sections 2.5 and 3.1, is based on the combination of extensive feature engineering and traditional classification algorithms, we can reap the benefits of explainable output. More precisely, we can backtrack our choices at different data manipulating steps and highlight the accuracy-increasing ones. While such a capability can help with optimising the workflow itself, e.g., by choosing better metrics and phases as the sources of data, it could also help in narrowing down the root causes of anomalies. For instance, the specific PIDs responsible for deviations in a phase with multiple active PIDs can be detected, as signatures and passports are generated per PID. As an example, if we look at the graph from Figure 5.4 and traverse the tree from its root as,

1. True, False, False, and
2. False, True, False,

both paths lead to *transient harmful* anomaly classification. For the enumeration 1, we see that software signatures from processes 44, 34 and 41 in combination with metrics writes count, CPU time and reads count, respectively, are definitive for the classifier. This knowledge provides a considerable reduction for the scope of desired analyses. Now we know which signatures and in turn, which parts of the raw

traces for a phase have resulted in a *transient harmful* anomaly. For the enumeration 2, we observe that the decision is based on the software signatures from processes 41 and 54, in combination with the metric reads count for process 41, twice, and the metric CPU time for process 54. Note that there are more leafs with this label present.

Besides root cause detection, other design-improving intelligence is available too. For instance, the methodology can show if a specific batch of inputs to the system results in unexpected anomalous behaviour, revealing design deficiencies in handling specific categories of inputs.

5.4.2 *Limitations of the study*

We have had a few limitations when dealing with our use-cases. Some of these were imposed by the industrial platform under scrutiny, such as the absence of real anomalous scenarios in the case of the semiconductor photolithography machine, leading us towards synthesising our own anomalous traces using the real trace and a mock-up platform, i.e., our implemented digital twin. Records of real anomalous executions belong to end users and are strictly confidential. Even the manufacturer, ASML, does not have access to such data. Others were self-imposed with the goal of having better control over experimentation and development of the methodology. For instance, in the case of the image analysis platform, we have opted for a single-threaded and single process application. For this use-case, the choice is still in line with real-world conditions as such set-ups are the preferred choice in low-power use-cases with sequential workflows, belonging to the lower end of the CPS complexity spectrum.

5.4.3 *Implications for industrial deployment*

As both our Classic ML and Advanced DL workflows involve supervised training of classifiers and at the same time, the Classic ML workflow requires generation and presence of behavioural passports as reference constructs, the most common question from the industry is that

> *how often such reference behavioural passport constructs, i.e., the baseline, have to be re-established?*

In other words, especially with regards to our Classic ML workflow, the validity of the behavioural passport construct is one of the key requirements for effective and confident anomaly detection and identification.

As we know, software is not a static entity and continuous development of new or improved functionality is the common practice of

choice. This creates a challenge in the sense that after each update to the software running an industrial CPS, some behavioural drifting from the original reference is expected, depending on the substantiality of the update. Presence of considerable drifting could render anomaly identification ineffective or erroneous, which in turn, necessitates the regeneration of reference behavioural passport constructs. To minimise such overhead, we can consider the new behaviour as unknown, i.e., newly generated post-update passports will be considered as signatures. Using original passports from the previous version as the reference point, the anomaly identification can be employed to check if the new behaviour is still identified as *normal*, as depicted in Figure 5.6.

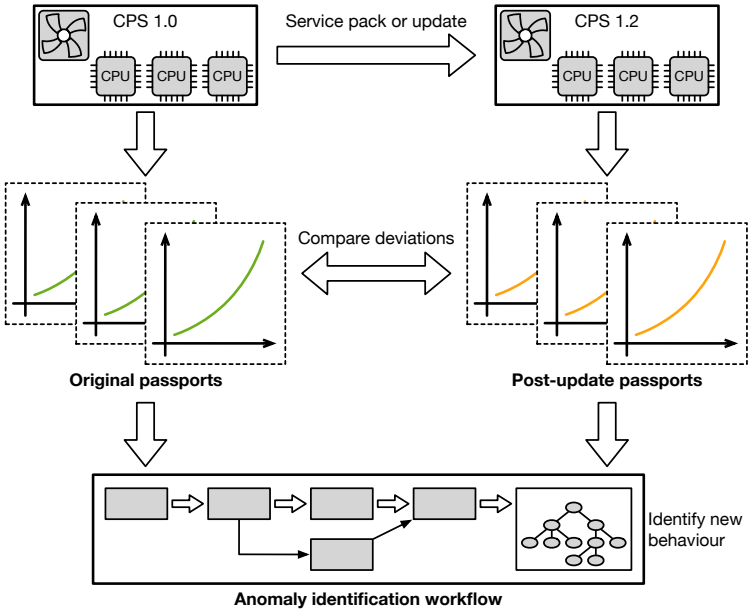


Figure 5.6: The high-level procedure view to check the validity of post-update behavioural passports, before releasing a new service pack or software update is provided. By considering the pre-update passports as reference and the post-update passports as signatures for comparison, the workflow can be used as is to evaluate these signatures.

As explained before in Section 2.4, behavioural passports can be generated per metric, per phase and per process, allowing us to pinpoint which part of the system's behaviour has drifted. This means that we only have to update the drifted passports out of the complete collection. For instance, if the collection of passports considered as reference includes the two metrics of CPU time and memory usage and the new update increases memory usage, but leaves the computational behaviour mostly unchanged, we can simply update passports related to memory usage. If need be, deviation measurements based

on goodness-of-fit tests can also be analysed to have a more granular form of evaluation.

To be able to take advantage of the benefits of our methodology, such a testing process can be part of the overall test and integration strategy for any given industrial CPS. As it is the case for most test and integration tasks in the industry, tests based on our methodology can also be automated.

When dealing with complex industrial CPS in production, yet another scepticism to ponder about is that

> *can we consider a single baseline as the reference for the whole operational timeline of an industrial CPS?*

While the behaviour of the system shows slight natural deviations during distinct executions, even if the conditions and the workload is identical, most often the behaviour at cold start deviates further away from the behaviour after system warm-up. Accordingly, for larger and more complex systems, it is advisable to have multiple sets of behavioural passports, addressing the referential needs per each *execution period*.

Jobs executed by industrial CPS are predominantly in the form of sequences of batches, e.g., the wafer processing workflow given in Section 1.4. Every time a new sequence starts, there will be all kinds of calibrations and adjustments occurring as the initial preparatory period. The first task of the first batch[1], immediately following these calibrations, is to be considered as a cold start task. Thus, the relevant behaviour has to be compared to the cold start reference. From the second task[2] onwards, the relevant behaviour is to be compared to the warmed-up reference. It may very well be the case that the cold start reference's deviation is not so significantly different, in which case, both references are to be considered during the generation of mean passports. The decision will depend on the initial analysis of the use-case at hand. Figure 5.7 showcases different expected executional periods that a complex industrial CPS goes through.

As an example, in a semiconductor photolithography machine, lots, i.e., batches of wafers, are queued for processing. Each lot has a reticle, i.e., the integrated circuit design to be applied on dies, assigned to it. Whenever the defined reticle changes, the machine's robotics will replace the currently installed reticle, invoking a natural delay and subsequent calibration effort. Therefore, for the case of a semiconductor photolithography machine, a reticle change can be considered as an indication for a comparison with the cold start reference. Assuming a single wafer's processing behaviour as our executional phase, the

1 Or the first batch entirely, depending on the granularity of behavioural analysis
2 Or the second batch

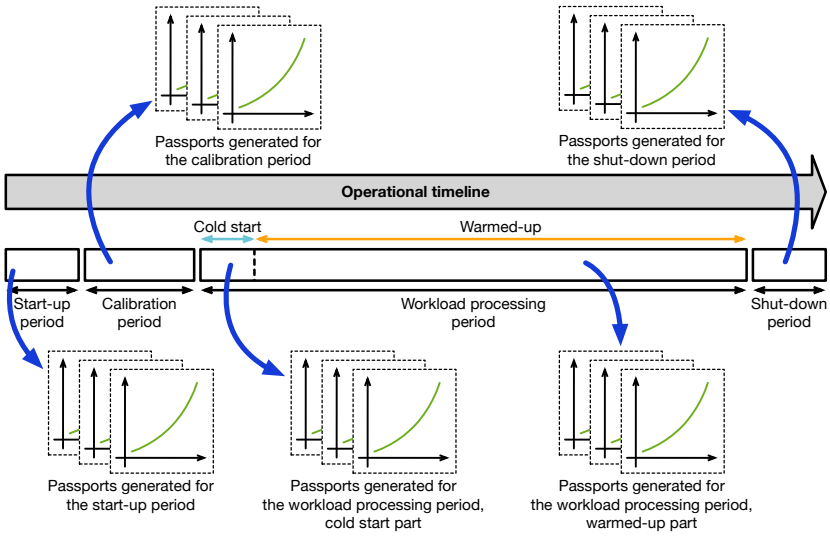


Figure 5.7: Depicted are typical executional periods, i.e., large scale phases, present in large and complex industrial CPS. It would be highly advantageous to detect and identify anomalies in every period as each has their own set of potential anomalies. Since passports for one period are not applicable to the rest, each period needs its own application of our methodology.

behaviour associated with the processing of the very first wafer after the reticle change will be compared to the cold start reference.

One recognisable consideration relevant to this discussion is that the higher the need to distinguish between such execution periods, the more our data requirements will be pushed towards a grey box or a white box information position, as we will need to have a more intimate understanding of the operational timeline. As we mentioned, such a need is fuelled by the complexity level of the system at hand.

6 RELATED WORK

The following includes our collected related work, organised based on major topics relevant to the content of our methodology and techniques of choice.

The contents of this chapter are mainly based on, but not limited to, the previously published conference and/or journal publications of the author. The publications of interest for Chapter 6 are:

- "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems" [50] (P2)
- "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems" [55] (P3)
- "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems" [49] (P4)
- "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB" [58] (P5)
- "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies" [59] (P6)
- "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI" [57] (P7)

Anomaly detection and identification have been on the agenda of the research community for a long time. The relevant body of knowledge is fairly large, with papers focusing on different aspects of anomaly detection and identification. Let us go over some of the more important trends in publications we have considered as our references.

Both Chandola et al. [11] and Ibidunmoye et al. [32] list collections of papers, covering the different strategies utilised by researchers for anomaly detection. In particular, they mention statistical detection, i.e., statistical techniques to detect trend drifts, observation detection, i.e., observation "through direct experimentation", and knowledge-based detection, i.e., detection based on historic patterns and known bottleneck definitions [32]. Compared to these strategies, we could say that

our methodology and workflows combine parts of each in one flow. For instance, we do use statistical techniques with our goodness-of-fit tests to quantify deviations. However this is not done for a trend, but for compartmentalised phases. We also follow the observation detection strategy, as we have test benches and observations are done during experimentations. When it comes to knowledge-based detection, we prefer the term data-centric, as the knowledge is not a priori available. Neither known bottlenecks to rely on, nor models representing the system (for simulation or otherwise), are available beforehand. Baselines will be composed from normal executions and models will be automatically synthesised from traces. More recent publications follow the same course in terms of data reliance, in the sense that they are moving towards solutions that are more data-centric, i.e., give better results in the presence of large amounts of data. Deep learning is gaining more momentum as a result of this, for deep learning models give better results as the amount of available data grows.

The challenging nature of decision support, leading to actuations and better designs for industrial systems, is also attested by the grand challenges presented by Fowler [22]. Most of the body of work given in [32] focus on performance anomalies in distributed systems [28], cloud environments [24] and web applications [14], whereas our methodology is specifically tailored towards industrial CPS. We are of the opinion that our approach helps in simplifying the task for repetitive systems by considering execution phases, regression-based representations and more importantly, white box, grey box, or black box information positions for industrial CPS, where it fits. We have also made use of a digital twin for our first use-case, which is in line with the ever-growing importance of such virtual representations for complex CPS [67].

It is worth repeating that our focus for data collection is on EFB, which is arguably rather similar to the notion of Key Performance Indicators (KPI) used in other works.

6.1 CLASSIFICATION ALGORITHMS

Any identification workflow ends with a classification algorithm and our workflows are no exception. As we have shown through our Classic ML workflows in Sections 4.1 and 4.2 and Advanced DL workflow in Section 4.4, the classification step could be based on traditional Machine Learning (ML) algorithms, or more advanced Deep Learning (DL) algorithms. Focusing on traditional algorithms here, i.e., Classic ML, we have specifically considered DT [69], RF [7], GaussianNB [23], k-NN [16], LinearSVC [4] and KernelSVM [15], as traditional classifiers in our implementations and result reporting. The extensive survey by Chandola et al. [11] covers publications on the topic of anomaly detection and provides a complete list of techniques utilised by different

authors. There are numerous publications taking advantage of neural networks, as well as publications choosing to work with traditional ML techniques, such as, Nearest Neighbour, Support Vector Machines, amongst others. These algorithms are applicable to repetitive data such as ours, just like how we use them.

On the other hand, the complexity of modern CPS has driven researchers towards DL techniques more than ever. In particular, more recent publications [27, 47, 68, 72] demonstrate this tendency, which we will expand in the following section. The recent survey by Chalapathy and Chawla [10], as well as the paper by Ratasich et al. [68], do point out the differences between traditional ML and sophisticated DL techniques by mentioning the black box nature of DL models. One advantage of our Classic ML workflow, which is based on traditional ML, is the ability to traverse the resulting models and backtrack the data pipeline, as mentioned in Section 5.4.1. When dealing with anomalies and looking for potential corrections, root-cause analysis is arguably an integral requirement. Traditional ML in general and our method in particular is rather capable in this aspect.

6.2 INTEREST IN DEEP LEARNING

The popularity of Artificial Intelligence (AI) is growing rapidly and its applications in almost any subfield in computer science is clearly observable. We also have recognised that the interest in solutions based on Deep Learning (DL) is an apparent one and for a good reason. DL is rather capable in cutting through the need for metadata and intimate familiarity of system internals, as we have also taken advantage of DL in our Advanced DL workflows. The prominence of DL is evident from further inclusion of DL models in publications, which is resulting from the desire to produce advanced solutions that reduce the complexity of implementation, by researchers. To further clarify, we are talking about the complexity of implementation for the code handling the training and the initialisation of models. DL models themselves are obviously much more complex compared to traditional ML models.

In the last few years, the complexity of CPS has led to elusive and indiscernible faults. Anomaly detection methods based on traditional machine learning are increasingly substituted with state-of-the-art deep learning techniques [10]. Moreover, CNNs have proven to be well suited for analysis of power signals and other similar time-series data for fault detection and classification [35]. Albasir et al. [2] proposed a CNN-based approach to detect malware activity, utilising the power consumption behaviour of smartphones. Canizo et al. [8] deployed CNN together with recurrent cells to detect anomalies in time-series data from multiple sensors. Deep learning models employing CNNs along with times series data have also shown promise in detection

of physical faults in components. Rotor bar fault detection based on raw stator current signals [33], refrigerant charge fault detection using sensor data from heat pumps [19] and status deduction based on power signals for health systems [45], are a few examples.

Being relevant to our topic, the recent publication from Luo et al. [47] specifically focuses on the use of DL for anomaly detection in Cyber-Physical Systems (CPS) and provides a handy listing of surveys on anomaly detection. Out of the given list, surveys by Giraldo et al. [26], Mitchell et al. [51], Lun et al. [81] and the survey by Luo et al. [47] itself, specifically focus on anomaly detection for CPS. There are also four surveys covering the DL-based solutions [10, 47, 52, 78]. The intersection of these publications is not that large, i.e., survey focusing on DL and anomaly detection and CPS are not too many, although individual publications are numerous. This suggests that there are challenges yet to be solved through DL in the realm of CPS. Interestingly enough, a considerable portion of these surveys focus on anomalies to address security and basically on attack detection. This has been a growing trend in recent years, to see anomalies as consequences of security incidents, which is a valid motivation. Industrial CPS such as photolithography machines on the other hand, are totally isolated and there is more added value in tackling performance anomalies to improve the machine yield, which our work tries to address.

6.3 ELECTRICAL METRICS

Electrical metrics, especially electrical power analysis, have a profound role in cybersecurity research. The famous and now classic paper by Kocher et al. [42] is a great example. What such publications have in common is their view towards electrical metrics in general and electrical power in particular, which is seen as a source of side-channel information. In principle, our view is rather similar and we see such metrics, external to the system, as reflections of its functional behaviour, which is the definition of Extra-Functional Behaviour (EFB).

Power signals can be very effective in detecting anomalies in an embedded system without the explicit requirement of adding extra hardware or software probes. Kim et al. [41] were among the first to highlight that power consumption can be used for anomaly detection, which are otherwise difficult to detect through the static characteristics of devices or the applications running on them. Their prototype works by leveraging power signatures based on the power consumed by the device while running an application. Caviglione et al. [9] detected attacks related to covert channels using the power consumption of the running processes. Covert channels occur when malicious applications exploit different assigned permissions and are able to exchange information. Liu et al. [46] developed a strategy using power side-channel

data to detect anomalous behaviour in control flow execution applied to IoT microcontrollers. Similarly, Xu et al. [80] used power channel to detect attacks on the Distribution Terminal Unit, a critical part of the power grid. Basically, we share a common view with these publications by considering electrical metrics as trend-revealing. The difference is in the application, behavioural fingerprinting as opposed to side-channel attacks.

6.4 POWER OF REGRESSION MODELLING

Considering our behavioural signature and behavioural passport generation technique, regression modelling is an important part of the Classic ML realisation of our methodology. The common use of regression models as a statistical tool for data estimation and inference is well argued in literature by Jain [36] and Chatfield [12]. This is especially true for different performance parameters of application processes as Lee and Brooks suggest [43]. Lee et al. also employ the notion of piecewise polynomial regression [44]. We have conducted a comparable strategy by dividing a timeline into meaningful phases, based on drastic changes in the values of EFB metrics, e.g., CPU utilisation. Though, our division criteria aims at having meaningful and repeatable phases. We are not using all points from an execution, i.e., certain unsuitable parts are being omitted.

Regression modelling has also been taken advantage of by Joseph et al. [38] and Barnes et al., [3] for correlating micro-architectural parameters with processor performance and exploring parallel programme scalability, respectively. Regression modelling has been the choice of Torr and Murray [61], as well as Chen et al. [13], within the domain of image processing.

6.5 MODELLING AND SIMULATION

Modelling and simulation based analysis of distributed industrial systems is a well known notion and its challenges have been on the agenda of the scientific community [17, 22]. Accordingly, our high-level workflow involves elements of the list given by Fowler, i.e., the "design, collect information/data, build, execute, and analyse" problem solving cycle [22]. Under our workflow, modelling and simulation steps are utilised towards building a digital twin. As a starting step in understanding of embedded system behaviour and complex computing systems involving them, efforts in modelling and design-space exploration resulted in efficient offline methodologies and tools, such as the well-known Y-chart approach [40], amongst others [20, 64]. These methodologies and tools are highly effective for CPS as well.

High-level modelling and discrete-event simulation are widely used techniques in the domain of embedded systems. Most existing system-level embedded system modelling and simulation research [17, 20, 25, 40, 64] has focused on:

1. manually building (engineering) system models and
2. typically addressing the design of the embedded systems, not the online analysis of existing, i.e., already engineered, systems.

Instrumenting software components of the system can be an effective way of collecting accurate and real-time information about system behaviour, as shown in [31].

The vast majority of embedded modelling and simulation studies focus on relatively simple use-cases, such a multi-media systems [5, 6, 21, 30, 48, 54, 66, 73, 75]. Our demonstrator, as a production-grade complex industrial CPS, involves challenges beyond the ones mentioned in [22]. We also take advantage of the architectural pattern of the communication subsystem, allowing us to synthesise models in an *automated* fashion, making it suitable for online application. Previous research has also looked at complex systems as a black box [62], or examples such as [70] have considered process mining techniques [1, 76, 79]. These approaches use readily available offline information. In contrast, our work aims at techniques suitable for an online solution, involving modelling and simulation at runtime. Applications of data-driven model generation in other fields, such as hydroinformatics [74] and signal-transduction networks [37] also worth mentioning.

7 CONCLUSION

Arguably, one of the main requirements in achieving robust industrial CPS is having available, reliable and robust software, amongst others. In this thesis, we elaborated and demonstrated our efforts towards achieving robust software operation. We have shown that a well-devised data-centric solution, as we have presented, is capable of anomaly detection and identification for industrial CPS, with high accuracy. Such a data-centric solution is much more flexible compared to self-adaptivity relying on traditional control mechanisms, as it is designed for consumption of monitoring data from a multitude of sources. This type of solution can deal with numerous anomalies at once, it will be expandable with new anomalies and it provides analytical capabilities towards root cause analysis. As mentioned in our motivation from Chapter 1, with the level of complexity at hand for modern industrial CPS, it is not possible to account for every potential anomaly and every corner case upfront. Accordingly, comprehensive mathematical relations between inputs and outputs of the system, cannot be confidently and comprehensively generated. A data-centric approach in general and our method in particular, allows for the addition of new anomalies that are yet to be seen. Resulting from this flexibility and as opposed to *process variable* and *set point*, which are considered by a controller, behavioural signatures and behavioural passports are the data-centric alternatives.

More precisely, we have shown a behaviour classification methodology composed of Extra-Functional Behaviour (EFB) monitoring at runtime, compartmentalisation of execution timeline into repetitive execution phases, generation of representative behavioural signatures and passports, deviation quantification based on goodness-of-fit tests, and traditional classification algorithms.

It must also be mentioned that though we have embarked on this path with tackling performance anomalies resulting from root causes within software in mind, our method is not limited to such anomalies. We have shown that anomalies resulting from physical hardware and electrical subsystems can also be taken on, which is another witness to flexibility of our method. We can generalise that it is, foremost, the choice of monitoring and EFB data collection points that lead to the extent of anomaly identification abilities.

Regarding the Classic ML workflow, our behavioural signature construct is an especially convenient tool, accommodating a variety of metrics as input and representing arbitrary lengths of execution time-

line of a system in a compact fashion. This compactness results in easy to store representations of reference behaviour, i.e., passports, as well as efficient to compare representations of ongoing behaviour, i.e., signatures. Using signatures, behaviour can be represented per execution phase, per metric and per process, if the application consists of multiple processes. We have also shown that the resulting comparison data as a feature set, is most suitable for *decision tree* and *random forest* classifiers, achieving accuracies as high as 99% in certain set-ups. The feature set is complete enough to facilitate classification of anomalies early on. In most anomalous cases it will take a while before visible deviations are present, e.g., under NoFan conditions.

With the Advanced ML workflow and deploying Convolutional Neural Networks (CNN), we have developed an alternative AI workflow, showing high classification accuracy of 94.85%. While achieving the high accuracy of the Classic ML required extensive design, feature engineering effort and costly computations, the Advanced DL also required extensive optimisation effort to come up with an accurate CNN model. We have discussed different qualitative aspects of both workflows, such as dependence on the intimate knowledge of the system and the data, stability of libraries and frameworks, efficient GPU implementation possibilities and root cause analysis through explainable output. There is no clear winner between these workflows. Critical applications and use-cases can benefit from highest accuracies and analytical capabilities provided by the Classic ML workflow, allowing the study of root causes behind anomalies, while ease of extension with different anomalies is best served by the Advanced DL workflow. It is totally use-case dependent.

In view of our proof-of-concept solutions based on given use-cases, we have demonstrated that our methodology is valid throughout the industrial CPS complexity spectrum. We have implemented our methodology for a semiconductor photolithography machine as a large and complex industrial CPS, as well as an image analysis platform as a low-power and less complex industrial CPS. We have also described different monitoring techniques. Techniques using system-level metrics in a white box, invasive and communication-centric approach, and techniques using external metrics in a black box, passive and side-channel-oriented approach. We have presented the role of a digital twin for industrial CPS and elaborated its creation using high-level communication-centric modelling and event-based simulation of traces captured with communication-centric monitoring. Our experiments indicate that the communication-centric analysis of industrial CPS, relying on communication subsystems is effective. Following such a perspective facilitates monitoring of the system at hand, resulting in a reduced collection of data, without an excessive loss in accuracy. The difference between captured and total CPU utilisation in our experi-

ments is around 10 percentage points, while the difference between simulated, i.e., estimated, and captured CPU utilisation is between 0 to 2 percentage points.

7.1 LOOKING BACK AT RESEARCH QUESTIONS

Let us briefly look back at our research questions given in Section 1.6 and review how our methodology and results are addressing them.

RESEARCH QUESTION 1

How can we follow behavioural diversity in industrial CPS through the variations embedded within sensory data, in an efficient manner?

We have addressed the two key points from this question, i.e., *following behavioural diversity* and the *efficiency* of such a following. The choice of a data-centric approach using EFB sourced data is the key in tracking non-deterministic behaviour, taking our observations beyond the functionalities of the system. When it comes to efficiency, the major role is played by executional phases and the communication-centric monitoring and modelling. Execution phases are relevant for both white box and grey box information positions, where only interesting and useful phases are taken into account. For a fully black box information position, techniques such as Change Point Detection (CPD) can replace the notion of execution phases. CPD however, is out of the scope of this thesis.

Communication-centric monitoring and modelling on the other hand, is effective when looking from a white box information position and in presence of communication subsystems. Rest assured, multi-node industrial CPS depend on such middleware for internode communication and their presence is rather common.

RESEARCH QUESTION 2

How can we demystify such embedded variations by only taking a partial, but yet, a descriptive view of the sensory data, to detect, identify and predict anomalous behaviour?

Demystification of embedded variations are done through our workflows with behavioural signatures and behavioural passports, as constructs representing the behaviour within a phase, at their core. These constructs provide the means to quantify deviation and create data sets, training traditional ML classifiers. Note that our alternative workflow using deep learning classifiers does not require signatures or passports.

RESEARCH QUESTION 3

What are the different approaches towards the identification of such anoma-

lous behaviour? What are the implications for production systems implementing such approaches?

When it comes to *different approaches*, we have demonstrated our Classic ML and Advanced DL classifiers, alongside the data pipelines supporting them. We have also discussed the differences between these two approaches in terms of training and classification speed, data preprocessing requirements and qualities, such as the ability to backtrack from an identified anomaly to raw monitoring data. Initially unforeseen implications of our solution, based on the interests of the industry has also been considered and discussed in Section 5.4.3.

7.2 FUTURE EXTENSIONS

Considering that the contents of this thesis involve topics from diverse subfields, the presented research can be extended in numerous directions. The first potential direction is an obvious one. Upon detection and identification of anomalous behaviour, corrective actions on the system to steer its behaviour back to normal or a manageable state, are highly sought after. Although, such actions may not be available for all anomalies, which brings up the importance of automated root cause analysis. As it is foreseen in our methodology from Figure 3.2, the digital twin plays a decisive role with regards to corrections. From the same figure, actuation policies addressing the anomalous behaviour at hand have to be validated through their application on the digital twin. This could perhaps involve considering a diverse set of event scheduling policies within the simulation. It is also foreseeable that there may very well be multiple actuation policies on the offer for a given anomaly. The advantage of a digital twin in such a scenario is that it allows for parallelisation of the validation effort by means of multiple instances.

Following the root cause analysis direction, we did touch upon the ability to backtrack from an identified anomaly (output) to source traces triggering such an identification (input) for our Classic ML workflow in Section 5.4.1. The real advantage though, is achieved if such an analysis could be performed in an automated fashion. That is no trivial task, for different identification results are based on different combinations of available features. We can observe this in a decision tree for instance. Clearly, the paths leading to different labels are not the same and there are even multiple paths present, leading to the same label.

Another direction to consider is to treat time series traces as signals. Such an approach will enable various possibilities from the signal processing discipline, e.g., CPD, which is likely to reduce our dependence on metadata information from the system internals. The less metadata we need, the closer to a grey box information position we will be,

which is a desirable effect for white box positions. Aside from automated phase boundary detection, finding the right phase granularity could also be facilitated. Currently, our methodology relies on expert knowledge and experimentation for the choice of the best phase.

Lastly, we make use of different flavours of classifiers in our workflows, both from traditional machine learning and more advanced deep learning domains. Since the behaviour of a system is an evolving characteristic, ultimately, there will be a point at which it would change beyond the tolerances of the reference behavioural passports. This change, will also trigger consequences for anomalous behavioural signatures and accordingly, for trained classification models. The lazy and perhaps relatively acceptable way of addressing such a setback is to run our workflows from scratch. However, we expect that there can be partial retraining strategies that are quicker to pull off and that are less resource-intensive. Negotiating such a challenge will require further research. An overview of the described potential extensions to the work presented in this thesis is depicted in Figure 7.1.

7.3 FINAL THOUGHTS

Ultimately, one might ask, what would be the gist of anomaly identifying solutions for industrial cyber-physical systems? How does it benefit the society from a grand perspective? We know that from an economic perspective, such solutions improve the yield of production systems and manufacturing machinery. This in turn, reduces costs and thus the price, which is advantageous for customers. However, is it really beneficial for a society to increase the accessibility of products and by implication, promote consumption culture? Perhaps not, or at least it depends on the product and its uses. At the very least, it is complicated. There are so many digital products and appliances in the market that do not serve a real need. Others, do have numerous benefits, but also come packed with non-useful and sometimes addictive features.

On the other hand, cyber-physical systems have a big presence in safety-critical and infrastructure applications [47]. Although our method is fine-tuned for repetitive systems, a characteristic that may not be readily present in safety-critical systems, but at the end of the day, the domain of tasks and behaviour related to these tasks is limited for all purpose-built CPS. We can indeed argue that with clever considerations, the concept of execution phases are applicable to these systems, at a more diverse scale. Bottom line, we could say that what we are doing does provide a clear positive impact for the society and solutions incentivised by economical requirements could culminate into improvements for what really matters.

Keep in mind that most of the research fuelling scientific advances in this field, or in any field for that matter, are supported by public

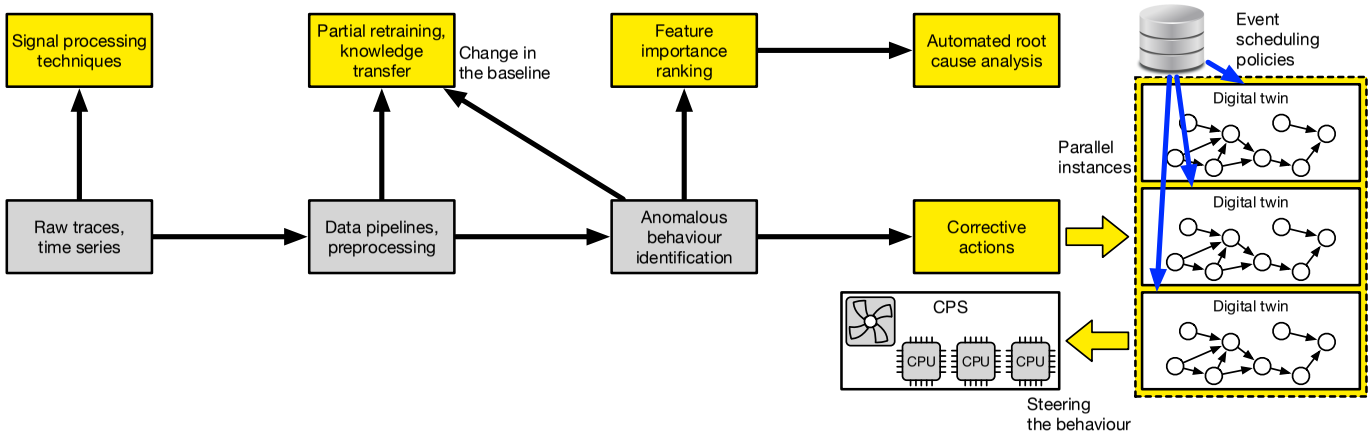


Figure 7.1: Potential future directions to the research work presented in this thesis are depicted, with extensions highlighted in yellow.

funding, meaning that taxpayers are spending money to be able to spend more money on products. Take these questions and thoughts with a grain of salt though, as these are multifaceted topics that require all sorts of different considerations. In short, in this author's humble opinion, it would be far more valuable to focus on the fact that these solutions improve safety-critical systems and as we all know, no virtuous recipe comes without at least a slight hint of mischief.

BIBLIOGRAPHY

[1] R. Agrawal, D. Gunopulos, and F. Leymann. "Mining process models from workflow logs". In: *Advances in Database Technology — EDBT'98*. Ed. by H. J. Schek, G. Alonso, F. Saltor, and I. Ramos. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 467–483. ISBN: 978-3-540-69709-1. DOI: 10.1007/BFb0101003.

[2] A. Albasir, R. Manzano, and K. Naik. "Deep Learning Based Approach for Classifying Power Signals and Detecting Anomalous Behavior of Wireless Devices". In: *2019 IEEE World Congress on Services (SERVICES)*. Vol. 2642-939X. 2019, pp. 92–98. DOI: 10.1109/SERVICES.2019.00030.

[3] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, and M. Schulz. "A Regression-Based Approach to Scalability Prediction". In: *Proceedings of the 22Nd Annual International Conference on Supercomputing*. ICS '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 368–377. ISBN: 9781605581583. DOI: 10.1145/1375527.1375580.

[4] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. "Support Vector Clustering". In: *Journal of Machine Learning Research* 2 (Mar. 2002), pp. 125–137. ISSN: 1532-4435.

[5] E. Bondarev, M. Chaudron, and P. H. N. de With. "CARAT: A Toolkit for Design and Performance Analysis of Component-based Embedded Systems". In: *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '07. Nice, France, 2007, pp. 1024–1029. ISBN: 978-3-9810801-2-4. DOI: 10.1109/DATE.2007.364428.

[6] E. Bondarev, P. de With, M. Chaudron, and J. Muskens. "Modelling of input-parameter dependency for performance predictions of component-based embedded systems". In: *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*. EUROMICRO '05. 2005, pp. 36–43. ISBN: 0-7695-2431-1. DOI: 10.1109/EUROMICRO.2005.40.

[7] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[8] M. Canizo, I. Triguero, A. Conde, and E. Onieva. "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study". In: *Neurocomputing* 363 (2019), pp. 246–260. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2019.07.034.

[9] L. Caviglione, M. Gaggero, J. Lalande, W. Mazurczyk, and M. Urbański. "Seeing the Unseen: Revealing Mobile Malware Hidden Communications via Energy Consumption and Artificial Intelligence". In: *IEEE Transactions on Information Forensics and Security* 11.4 (2016), pp. 799–810. DOI: 10.1109/TIFS.2015.2510825.

[10] R. Chalapathy and S. Chawla. *Deep Learning for Anomaly Detection: A Survey*. 2019. arXiv: 1901.03407 [cs.LG].

[11] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[12] C. Chatfield. "Model Uncertainty, Data Mining and Statistical Inference". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 158.3 (1995), pp. 419–466. DOI: 10.2307/2983440.

[13] D. Chen, X. Shao, B. Hu, and Q. Su. "Simultaneous Wavelength Selection and Outlier Detection in Multivariate Regression of Near-Infrared Spectra". In: *Analytical Sciences* 21.2 (2005), pp. 161–166. DOI: 10.2116/analsci.21.161.

[14] L. Cherkasova, K. Ozonat, M. Ningfang, J. Symons, and E. Smirni. "Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change". In: *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. 2008, pp. 452–461. DOI: 10.1109/DSN.2008.4630116.

[15] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[16] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[17] P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli. "Modeling Cyber-Physical Systems". In: *Proceedings of the IEEE* 100.1 (2012), pp. 13–28. ISSN: 0018-9219. DOI: 10.1109/JPROC.2011.2160929.

[18] D. Dvorak. "NASA Study on Flight Software Complexity". In: *AIAA Infotech@Aerospace Conference*. 2009. DOI: 10.2514/6.2009-1882.

[19] Y. H. Eom, J. W. Yoo, S. B. Hong, and M. S. Kim. "Refrigerant charge fault detection method of air source heat pump system using convolutional neural network for energy saving". In: *Energy* 187 (2019), p. 115877. ISSN: 0360-5442. DOI: 10.1016/j.energy.2019.115877.

[20] C. Erbas, A. D. Pimentel, M. Thompson, and S. Polstra. "A Framework for System-Level Modeling and Simulation of Embedded Systems Architectures". In: *EURASIP Journal on Embedded Systems* 2007.1 (2007). ISSN: 1687-3963. DOI: 10.1155/2007/82123.

[21] A. Filieri, H. Hoffmann, and M. Maggio. "Automated Design of Self-Adaptive Software with Control-Theoretical Formal Guarantees". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: Association for Computing Machinery, 2014, pp. 299–310. ISBN: 9781450327565. DOI: 10.1145/2568225.2568272.

[22] J. W. Fowler and O. Rose. "Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems". In: *SIMULATION* 80.9 (2004), pp. 469–476. DOI: 10.1177/0037549704044324.

[23] N. Friedman, D. Geiger, and M. Goldszmidt. "Bayesian Network Classifiers". In: *Machine Learning* 29.2 (1997), pp. 131–163. ISSN: 1573-0565. DOI: 10.1023/A:1007465528199.

[24] S. Fu. "Performance Metric Selection for Autonomic Anomaly Detection on Cloud Computing Systems". In: 2011, pp. 1–5. DOI: 10.1109/GLOCOM.2011.6134532.

[25] A. Gerstlauer, C. Haubelt, A. D. Pimentel, T. P. Stefanov, D. D. Gajski, and J. Teich. "Electronic System-level Synthesis Methodologies". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.10 (2009), pp. 1517–1530. ISSN: 1937-4151. DOI: 10.1109/TCAD.2009.2026356.

[26] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell. "A Survey of Physics-Based Attack Detection in Cyber-Physical Systems". In: *ACM Comput. Surv.* 51.4 (July 2018). ISSN: 0360-0300. DOI: 10.1145/3203245.

[27] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. "Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks". In: 2017, pp. 140–145. DOI: 10.1109/HASE.2017.36.

[28] D. Gunter, B. L. Tierney, A. Brown, M. Swany, J. Bresnahan, and J. M. Schopf. "Log summarization and anomaly detection for troubleshooting distributed systems". In: 2007, pp. 226–234. DOI: 10.1109/GRID.2007.4354137.

[29] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[30] H. Hoffmann. "CoAdapt: Predictable Behavior for Accuracy-Aware Applications Running on Power-Aware Systems". In: *2014 26th Euromicro Conference on Real-Time Systems*. 2014, pp. 223–232. ISBN: 978-1-4799-5798-9. DOI: 10.1109/ECRTS.2014.32.

[31] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. E. Miller, and A. Agarwal. "Application Heartbeats: A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments". In: *Proceedings of the 7th International Conference on Autonomic Computing*. ICAC '10. Washington, DC, USA: Association for Computing Machinery, 2010, pp. 79–88. ISBN: 9781450300742. DOI: 10.1145/1809049.1809065.

[32] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth. "Performance Anomaly Detection and Bottleneck Identification". In: *ACM Comput. Surv.* 48.1 (July 2015). ISSN: 0360-0300. DOI: 10.1145/2791120.

[33] T. Ince. "Real-time broken rotor bar fault detection and classification by shallow 1D convolutional neural networks". In: *Electrical Engineering* 101.2 (2019), pp. 599–608. ISSN: 1432-0487. DOI: 10.1007/s00202-019-00808-7.

[34] information is beautiful. *Codebases - Millions of lines of code*. 2015. URL: www.informationisbeautiful.net/visualizations/million-lines-of-code/ (visited on 06/29/2021).

[35] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller. "Deep Learning for Time Series Classification: A Review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1.

[36] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1990. ISBN: 978-0-471-50336-1.

[37] K. A. Janes and M. B. Yaffe. "Data-driven modelling of signal-transduction networks". In: *Nature Reviews Molecular Cell Biology* 7.11 (2006), pp. 820–828. ISSN: 1471-0080. DOI: 10.1038/nrm2041.

[38] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. "Construction and use of linear regression models for processor performance analysis". In: *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* 2006, pp. 99–108. DOI: 10.1109/HPCA.2006.1598116.

[39] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. "Model-integrated development of embedded software". In: *Proceedings of the IEEE* 91.1 (2003), pp. 145–164. DOI: 10.1109/JPROC.2002.805824.

[40] B. Kienhuis, E. F. Deprettere, P. van der Wolf, and K. A. Vissers. "A Methodology to Design Programmable Embedded Systems - The Y-Chart Approach". In: *Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation - SAMOS*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 18–37. ISBN: 978-3-540-45874-6. DOI: 10.1007/3-540-45874-3_2.

[41] H. Kim, J. Smith, and K. G. Shin. "Detecting Energy-Greedy Anomalies and Mobile Malware Variants". In: *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*. MobiSys '08. Breckenridge, CO, USA: Association for Computing Machinery, 2008, pp. 239–252. ISBN: 9781605581392. DOI: 10.1145/1378600.1378627.

[42] P. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by M. Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_25.

[43] B. C. Lee and D. M. Brooks. "Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction". In: *SIGOPS Oper. Syst. Rev.* 40.5 (Oct. 2006), pp. 185–194. ISSN: 0163-5980. DOI: 10.1145/1168917.1168881.

[44] B. C. Lee, D. M. Brooks, B. R. de Supinski, M. Schulz, K. Singh, and S. A. McKee. "Methods of Inference and Learning for Performance Modeling of Parallel Applications". In: *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. PPoPP '07. San Jose, California, USA: Association for Computing Machinery, 2007, pp. 249–258. ISBN: 978-1-59593-602-8. DOI: 10.1145/1229428.1229479.

[45] H. Lee, Y. Kwon, and K. Kim. "Power Signal Classification with Combinational Spectrogram-based CNN for Embedded System Health Management". In: *2018 18th International Conference on Control, Automation and Systems (ICCAS)*. 2018, pp. 1102–1106.

[46] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu. "On Code Execution Tracking via Power Side-Channel". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 1019–1031. ISBN: 9781450341394. DOI: 10.1145/2976749.2978299.

[47] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao. "Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities". In: *ACM Comput. Surv.* 54.5 (May 2021). ISSN: 0360-0300. DOI: 10.1145/3453155.

[48] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva. "Power Optimization in Embedded Systems via Feedback Control of Resource Allocation". In: *IEEE Transactions on Control Systems Technology* 21.1 (2013), pp. 239–246. ISSN: 1558-0865. DOI: 10.1109/TCST.2011.2177499.

[49] H. Meyer, U. Odyurt, A. D. Pimentel, E. Paradas, and I. Gonzalez Alonso. "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. SAC '20. Brno, Czech Republic: Association for Computing Machinery, 2020, pp. 210–217. ISBN: 9781450368667. DOI: 10.1145/3341105.3373851.

[50] H. Meyer, U. Odyurt, S. Polstra, E. Paradas, I. Gonzalez Alonso, and A. D. Pimentel. "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems". In: *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. 2018, pp. 979–986. ISBN: 978-1-7281-1141-4. DOI: 10.1109/BDCloud.2018.00143.

[51] R. Mitchell and I. R. Chen. "A Survey of Intrusion Detection Techniques for Cyber-Physical Systems". In: *ACM Comput. Surv.* 46.4 (Mar. 2014). ISSN: 0360-0300. DOI: 10.1145/2542049.

[52] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. "Deep Learning for IoT Big Data and Streaming Analytics: A Survey". In: *IEEE Communications Surveys Tutorials* 20.4 (2018), pp. 2923–2960. DOI: 10.1109/COMST.2018.2844341.

[53] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. "Cyber-physical systems in manufacturing". In: *CIRP Annals* 65.2 (2016), pp. 621–641. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2016.06.005.

[54] A. Muttreja, A. Raghunathan, S. Ravi, and N. K. Jha. "Automated Energy/Performance Macromodeling of Embedded Software". In: *Proceedings of the 41st Annual Design Automation Conference*. DAC '04. 2004, pp. 99–102. ISBN: 1-51183-828-8. DOI: 10.1145/996566.996599.

[55] U. Odyurt, H. Meyer, A. D. Pimentel, E. Paradas, and I. Gonzalez Alonso. "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems". In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by D. N. Pnevmatikatos, M. Pelcat, and M. Jung. Cham: Springer International Publishing, 2019, pp. 255–268. ISBN: 978-3-030-27562-4. DOI: 10.1007/978-3-030-27562-4_18.

[56] U. Odyurt, H. Meyer, S. Polstra, E. Paradas, I. Gonzalez Alonso, and A. D. Pimentel. "Work-in-Progress: Communication-Centric Analysis of Complex Embedded Computing Systems". In: *2018 International Conference on Embedded Software (EMSOFT)*. 2018,

pp. 1–3. ISBN: 978-1-5386-5560-3. DOI: 10.1109/EMSOFT.2018.8537189.

[57] U. Odyurt, A. D. Pimentel, and Ignacio Gonzalez Alonso. "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI". In: (2021). (currently under review by the journal "IET Cyber-Physical Systems: Theory & Applications").

[58] U. Odyurt, J. Roeder, A. D. Pimentel, I. Gonzalez Alonso, and C. de Laat. "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB". In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. 2021, pp. 152–157. ISBN: 978-1-7281-6207-2. DOI: 10.1109/ICPS49255.2021.9468262.

[59] U. Odyurt, D. Sapra, and A. D. Pimentel. "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies". In: *Advances and Trends in Artificial Intelligence. From Theory to Practice*. Ed. by H. Fujita, A. Selamat, J. CW. Lin, and M. Ali. Cham: Springer International Publishing, 2021, pp. 474–484. ISBN: 978-3-030-79463-7. DOI: 10.1007/978-3-030-79463-7_40.

[60] E. Ostertagová. "Modelling using Polynomial Regression". In: *Procedia Engineering* 48 (2012). Modelling of Mechanical and Mechatronics Systems, pp. 500–506. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2012.09.545.

[61] D. W. Murray P. H. S. Torr. "Outlier detection and motion segmentation". In: *Sensor Fusion VI*. Ed. by P. S. Schenker. Vol. 2059. International Society for Optics and Photonics. SPIE, 1993, pp. 432–443. DOI: 10.1117/12.150246.

[62] V. V. Parappurath, J. P. M. Voeten, and K. C. Kotterink. "Calibration Error Bound Estimation in Performance Modeling". In: *Proceedings of the 2013 Euromicro Conference on Digital System Design*. DSD '13. 2013, pp. 97–102. ISBN: 978-1-4799-2978-8. DOI: 10.1109/DSD.2013.18.

[63] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[64] A. D. Pimentel, C. Erbas, and S. Polstra. "A systematic approach to exploring embedded system architectures at multiple abstraction levels". In: *IEEE Transactions on Computers* 55.2 (2006), pp. 99–112. ISSN: 1557-9956. DOI: 10.1109/TC.2006.16.

[65] Qoitech AB. *Otii Arc - Otii by QOITECH*. 2020. URL: www.qoitech.com/otii/ (visited on 06/29/2021).

[66] W. Quan and A. D. Pimentel. "A Scenario-Based Run-Time Task Mapping Algorithm for MPSoCs". In: *Proceedings of the 50th Annual Design Automation Conference*. DAC '13. Austin, Texas: Association for Computing Machinery, 2013. ISBN: 9781450320719. DOI: 10.1145/2463209.2488895.

[67] A. Rasheed, O. San, and T. Kvamsdal. "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective". In: *IEEE Access* 8 (2020), pp. 21980–22012. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2970143.

[68] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci. "A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems". In: *IEEE Access* 7 (2019), pp. 13260–13283. DOI: 10.1109/ACCESS.2019.2891969.

[69] L. Rokach and O. Maimon. "Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Ed. by O. Maimon and L. Rokach. Boston, MA: Springer US, 2005, pp. 165–192. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_9.

[70] A. Rozinat, I. S. M. De Jong, C. W. Günther, and W. M. P. van der Aalst. "Process Mining Applied to the Test Process of Wafer Scanners in ASML". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39.4 (2009), pp. 474–479. ISSN: 1558-2442. DOI: 10.1109/TSMCC.2009.2014169.

[71] G. H. Ruffo. *Tesla Cars Have A Memory Problem That May Cost You A Lot To Repair*. 2019. URL: insideevs.com/news/376037/tesla-mcu-emmc-memory-issue/ (visited on 06/29/2021).

[72] P. Schneider and K. Böttinger. "High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 9781450359924. DOI: 10.1145/3264888.3264890.

[73] M. Shafique, L. Bauer, and J. Henkel. "enBudget: A Run-time Adaptive Predictive Energy-budgeting Scheme for Energy-aware Motion Estimation in H.264/MPEG-4 AVC Video Encoder". In: *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '10. 2010, pp. 1725–1730. ISBN: 978-3-9810801-6-2. DOI: 10.1109/DATE.2010.5457093.

[74] D. Solomatine, L. M. See, and R. J. Abrahart. "Data-Driven Modelling: Concepts, Approaches and Experiences". In: *Practical Hydroinformatics: Computational Intelligence and Technological Developments in Water Applications*. Ed. by Robert J. Abrahart, Linda M. See, and Dimitri P. Solomatine. Berlin, Heidelberg: Springer

Berlin Heidelberg, 2008, pp. 17–30. ISBN: 978-3-540-79881-1. DOI: 10.1007/978-3-540-79881-1_2.

[75] S. Stuijk, M. Geilen, B. Theelen, and T. Basten. "Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications". In: *2011 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*. 2011, pp. 404–411. ISBN: 978-1-4577-0801-5. DOI: 10.1109/SAMOS.2011.6045491.

[76] W. van der Aalst, T. Weijters, and L. Maruster. "Workflow mining: discovering process models from event logs". In: *IEEE Transactions on Knowledge and Data Engineering* 16.9 (Sept. 2004), pp. 1128–1142. ISSN: 1558-2191. DOI: 10.1109/TKDE.2004.47.

[77] A. Varga and R. Hornig. "An Overview of the OMNeT++ Simulation Environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. ISBN: 9789639799202.

[78] E. MSP Veith, L. Fischer, M. Tröschel, and A. Nieße. "Analyzing Cyber-Physical Systems from the Perspective of Artificial Intelligence". In: *Proceedings of the 2019 International Conference on Artificial Intelligence, Robotics and Control*. AIRC '19. Cairo, Egypt: Association for Computing Machinery, 2019, pp. 85–95. ISBN: 9781450376716. DOI: 10.1145/3388218.3388222.

[79] A. J. M. M. Weijters and W. M. P. van der Aalst. "Process mining: discovering workflow models from event-based data". In: *Proceedings of the 13th Belgium-Dutch Conference on Artificial Intelligence (BNAIC 2001)*. Ed. by B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren. BNVKI, 2001, pp. 283–290.

[80] A. Xu, Y. Jiang, Y. Cao, G. Zhang, X. Ji, and W. Xu. "ADDP: Anomaly Detection for DTU Based on Power Consumption Side-Channel". In: *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*. 2019, pp. 2659–2663. DOI: 10.1109/EI247390.2019.9062014.

[81] Y. Zacchia Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto. "State of the art of cyber-physical systems security: An automatic control perspective". In: *Journal of Systems and Software* 149 (2019), pp. 174–216. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.12.006.

PUBLICATIONS

[1] H. Meyer, **U. Odyurt**, A. D. Pimentel, E. Paradas, and I. Gonzalez Alonso. "An Analytics-Based Method for Performance Anomaly Classification in Cyber-Physical Systems". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. SAC '20. Brno, Czech Republic: Association for Computing Machinery, 2020, pp. 210–217. ISBN: 9781450368667. DOI: 10.1145/3341105.3373851.

[2] H. Meyer, **U. Odyurt**, S. Polstra, E. Paradas, I. Gonzalez Alonso, and A. D. Pimentel. "On the Effectiveness of Communication-Centric Modelling of Complex Embedded Systems". In: *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. 2018, pp. 979–986. ISBN: 978-1-7281-1141-4. DOI: 10.1109/BDCloud.2018.00143.

[3] **U. Odyurt**, H. Meyer, A. D. Pimentel, E. Paradas, and I. Gonzalez Alonso. "Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems". In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by D. N. Pnevmatikatos, M. Pelcat, and M. Jung. Cham: Springer International Publishing, 2019, pp. 255–268. ISBN: 978-3-030-27562-4. DOI: 10.1007/978-3-030-27562-4_18.

[4] **U. Odyurt**, H. Meyer, S. Polstra, E. Paradas, I. Gonzalez Alonso, and A. D. Pimentel. "Work-in-Progress: Communication-Centric Analysis of Complex Embedded Computing Systems". In: *2018 International Conference on Embedded Software (EMSOFT)*. 2018, pp. 1–3. ISBN: 978-1-5386-5560-3. DOI: 10.1109/EMSOFT.2018.8537189.

[5] **U. Odyurt**, A. D. Pimentel, and Ignacio Gonzalez Alonso. "Improving the Robustness of Industrial Cyber-Physical Systems Using Behavioural Signatures, Behavioural Passports and AI". In: (2021). (currently under review by the journal "IET Cyber-Physical Systems: Theory & Applications").

[6] **U. Odyurt**, J. Roeder, A. D. Pimentel, I. Gonzalez Alonso, and C. de Laat. "Power Passports for Fault Tolerance: Anomaly Detection in Industrial CPS Using Electrical EFB". In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. **IEEE IES Students and Young Professionals Best Paper Award**

and **Best Presentation in Session Award**. 2021, pp. 152–157. ISBN: 978-1-7281-6207-2. DOI: 10.1109/ICPS49255.2021.9468262.

[7] **U. Odyurt**, D. Sapra, and A. D. Pimentel. "The Choice of AI Matters: Alternative Machine Learning Approaches for CPS Anomalies". In: *Advances and Trends in Artificial Intelligence. From Theory to Practice*. Ed. by H. Fujita, A. Selamat, J. CW. Lin, and M. Ali. Cham: Springer International Publishing, 2021, pp. 474–484. ISBN: 978-3-030-79463-7. DOI: 10.1007/978-3-030-79463-7_40.

SUMMARY

In this thesis, we take a strictly data-centric approach to tackle the challenge of anomaly detection and identification in industrial Cyber-Physical Systems (CPS). This work is specifically considering anomalies as deviations from the normal behaviour of a CPS, affecting its throughput, or disrupting its stable state, e.g., issues with timeliness. CPS in general, are complex systems interacting with the physical realm, integrating multiple computing nodes with heterogeneous architectures that are networked and distributed. The natural progression of CPS can be seen as a steady computerisation trend. Accordingly, these systems have evolved into software-intensive designs with a plethora of available hardware sensors and software probes, turning them into data-rich ecosystems. To that extent, data-rich ecosystems can best be interacted with, through data-centric methods.

Though the described evolution is valid for all CPS, we further focus on a specific breed, industrial CPS. These systems do share the same characteristics as other CPS, but more importantly, they are purpose-built to address industrial and manufacturing tasks relevant to their intended use-case. As a result, they demonstrate highly repetitive operational patterns. We analyse such patterns by considering repeated units of execution, i.e., *execution phases* as we call them, to define the aimed compartmentalisation of the executional timeline of the system under scrutiny. The sensory data collected within the boundaries of a phase in time, is further processed, resulting in the generation of unique *behavioural signatures*. The sensory data comes from continuous data collections, or purposefully planted software probes, recording metric readings that reflect *Extra-Functional Behaviour (EFB)* of the system. Examples are metrics revealing system's performance behaviour, e.g., CPU time or memory consumption, or metrics revealing system's power/energy consumption behaviour, e.g., electrical current. Accordingly, behavioural signatures are generated per phase, per metric and if applicable, per process. Behavioural signatures generated from reference executions, a.k.a., golden executions, of the system that are known to be anomaly-free, are considered as *behavioural passports*.

Our technique of choice when it comes to generating efficient and capable representations for behavioural signatures and passports is regression modelling. Regression modelling results in an efficient output, as it reduces any number of collected data points to their closest mathematical function by means of interpolation. Regression models are also capable constructs for the purpose of deviation quantification, when combined with goodness-of-fit statistical tests. These tests are

used to quantify the amount of deviation between a given behavioural signature and its corresponding behavioural passport.

We are taking advantage of Artificial Intelligence (AI) in our solution through the application of classifiers, identifying different anomaly types. Deviation results from comparisons and other values, e.g., coefficients from regression functions, are considered as the feature set. Ultimately, we train different classifiers, e.g., decision tree and random forest, with the resulting data set, in a supervised fashion.

We have demonstrated the effectiveness of our data-centric methodology with two proofs-of-concept from the industry, to represent the two ends of the industrial CPS complexity spectrum, with one being a large semiconductor photolithography machine, while the other is an image analysis platform. Each use-case comes with its own characteristics and limitations, confirming the flexibility of our methodology and the relevance of its integral steps in the approach towards the initial analysis and data transformations. We have shown the overall high accuracy of our anomaly identification methodology. By considering the right choice of phase and metric combination, we were able to achieve anomaly identification accuracies above 99%.

Considering our second use-case, the image analysis platform, we compare the pros and cons of two different approaches when composing our AI solution. We demonstrate how our *Classic ML* workflow, based on traditional Machine Learning (ML) classifiers, differs from our *Advanced DL* workflow, based on more sophisticated Deep Learning (DL) with Convolutional Neural Networks (CNN) models. Though both workflows prove to result in highly accurate classification of anomalies, Classic ML is superior in this regard, with 99.23% accuracy against 94.85% from Advanced DL. This comes at a cost, as Classic ML requires total insight and expertise regarding the system under scrutiny and heavy amounts of feature engineering, while Advanced DL treats the data as a black box, minimising the amount of preprocessing. At the same time, we show that finding the best performing CNN model design for our Advanced DL workflow is not trivial. We present a quantitative comparison of both workflows in terms of elapsed times for training, validation and preprocessing, alongside discussions on qualitative aspects.

SAMENVATTING

In dit proefschrift hanteren we een strikt datacentrische benadering om de uitdaging van anomaliedetectie en identificatie in industriële Cyber-Fysieke Systemen (CFS) aan te pakken. Dit werk beschouwt anomalieën specifiek als afwijkingen van het normale gedrag van een CFS, die de doorvoer beïnvloeden of de stabiele toestand verstoren, bijvoorbeeld problemen met tijdigheid. CFS in het algemeen zijn complexe systemen die interageren met het fysieke domein, waarbij meerdere computerknooppunten worden geïntegreerd met heterogene architecturen die zijn genetwerkt en gedistribueerd. Het natuurlijke verloop van CFS kan worden gezien als een gestage automatiseringstrend. Zodoende zijn deze systemen geëvolueerd naar software-intensieve ontwerpen met een overvloed aan beschikbare hardwaresensoren en softwaresondes, waardoor ze in gegevensrijke ecosystemen zijn veranderd. In die mate kan met datarijke ecosystemen het beste worden gecommuniceerd via datacentrische methoden.

Hoewel de beschreven evolutie geldt voor alle CFS, richten we ons verder op een specifiek soort, de industriële CFS. Deze systemen hebben dezelfde kenmerken als andere CFS, maar wat belangrijker is, ze zijn speciaal gebouwd om industriële taken en productietaken aan te pakken die relevant zijn voor het beoogde gebruik. Als gevolg hiervan vertonen ze zeer repetitieve operationele patronen. We analyseren dergelijke patronen door herhaalde uitvoeringseenheden te beschouwen, d.w.z. zogenoemde uitvoeringsfasen, om de beoogde compartimentering van de uitvoeringstijdlijn van het onderzochte systeem te definiëren. De sensorische data die binnen de grenzen van een fase in de tijd worden verzameld, worden verder verwerkt, wat resulteert in het genereren van unieke gedragssignaturen. De sensorische gegevens zijn afkomstig van continue gegevensverzamelingen, of doelbewust geplante softwaresondes, die metrische aflezingen registreren die het Extra-Functioneel Gedrag (EFG) van het systeem weerspiegelen. Voorbeelden zijn meetwaarden die het prestatiegedrag van het systeem onthullen, bijv. CPU tijd of geheugenverbruik, of meetwaarden die het stroom/energieverbruik van het systeem onthullen, bijv. elektrische stroom. Zodoende worden gedragssignaturen gegenereerd per fase, per metriek en indien van toepassing per proces. Gedragshandtekeningen die zijn gegenereerd op basis van referentie executies, ook wel gouden executies genoemd, van het systeem waarvan bekend is dat het afwijkingsvrij is, worden beschouwd als gedragspaspoorten.

Onze gekozen techniek als het gaat om het genereren van efficiënte en bekwame representaties voor gedragshandtekeningen en paspoorten is

regressiemodellering. Regressiemodellering resulteert in een efficiënte output, aangezien het een willekeurig aantal verzamelde datapunten reduceert tot hun dichtstbijzijnde wiskundige functie door middel van interpolatie. Regressiemodellen zijn ook bekwame constructies voor het kwantificeren van afwijkingen, in combinatie met goodness-of-fit statistische tests. Deze tests worden gebruikt om de hoeveelheid afwijking tussen een bepaalde gedragssignatuur en het bijbehorende gedragspaspoort te kwantificeren.

We maken gebruik van Kunstmatige Intelligentie (KI) in onze oplossing door classificaties toe te passen, waarmee verschillende typen anomalie worden geïdentificeerd. Afwijkingsresultaten van vergelijkingen en andere waarden, bijv. coëfficiënten van regressiefuncties, worden beschouwd als de feature set. Uiteindelijk trainen we verschillende classifiers, zoals een decision tree en een random forest, met de resulterende data set, met supervised learning.

We hebben de effectiviteit van onze datacentrische methodologie aangetoond met twee soorten industriële proof-of-concept, om de twee uiteinden van het industriële CFS-complexiteitsspectrum te vertegenwoordigen, waarbij de ene een grote halfgeleider fotolithografiemachine is en de andere een platform voor beeldanalyse. Elke use-case heeft zijn eigen kenmerken en beperkingen, wat de flexibiliteit van onze methodologie en de relevantie van de integrale stappen in de benadering van de initiële analyse en datatransformaties bevestigt. We hebben de algemene hoge nauwkeurigheid van onze anomalie identificatiemethode aangetoond. Door de juiste keuze van fase en metrische combinatie te overwegen, waren we in staat om anomalie identificatienauwkeurigheden van meer dan 99% te bereiken.

Gezien onze tweede use-case, het beeldanalyseplatform, vergelijken we de voor en nadelen van twee verschillende benaderingen bij het samenstellen van onze KI oplossing. We laten zien hoe onze Classic ML workflow, gebaseerd op traditionele Machine Learning (ML) classifiers, verschilt van onze Advanced DL-workflow, gebaseerd op meer geavanceerde Deep Learning (DL) met Convolutional Neural Network (CNN) modellen. Hoewel beide workflows blijken te resulteren in zeer nauwkeurige classificatie van afwijkingen, is Classic ML superieur in dit opzicht, met een nauwkeurigheid van 99,23% tegen 94,85% van Advanced DL. Dit brengt kosten met zich mee, aangezien Classic ML volledig inzicht en expertise vereist met betrekking tot het systeem dat onder de loep wordt genomen en daarbij grote hoeveelheden feature engineering vereist, terwijl Advanced DL de gegevens behandelt als een black box, waardoor de hoeveelheid preprocessing wordt geminimaliseerd. Tegelijkertijd laten we zien dat het vinden van het best presterende CNN modelontwerp voor onze Advanced DL workflow niet triviaal is. We presenteren een kwantitatieve vergelijking van beide

workflows in termen van verstreken tijd voor training, validatie en preprocessing, naast discussies over kwalitatieve aspecten.