Probabilistic Methods for Predictive Maintenance and Beyond: Graph and Human-in-the-Loop Machine Learning

Alexander Nikitin
Probabilistic Methods for Predictive Maintenance and Beyond: Graph and Human-in-the-Loop Machine Learning

Alexander Nikitin

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall U7 PWC (Otakaari 1, Espoo) of the school on 15 February 2024 at 15:00.
Supervising professor
Professor Samuel Kaski, Aalto University, Finland and The University of Manchester, United Kingdom

Preliminary examiners
Professor Zaharah A. Bukhsh, Eindhoven University of Technology (TU/e), Netherlands
Professor Frank Emmert-Streib, Tampere University, Finland

Opponent
Professor Stefan Feuerriegel, Ludwig Maximilian University of Munich, Germany
Abstract

Probabilistic methods are key tools for machine learning problems. Even so, there remain many applications where they cannot be applied due to their limitations. These limitations may include the lack of methods for a particular data format (e.g., manifolds, texts, or graphs), data unavailability, or the inability to work collaboratively with human experts.

Inspired by problems in predictive maintenance (PdM), this thesis introduces a set of machine learning solutions that are more generally applicable. It begins with applied tasks in cable networks, data centers, and other telecom applications and indicates the crucial limitations of current approaches: the absence of (i) probabilistic methods for spatio-temporal graph problems, (ii) practical human-in-the-loop methods that learn from detailed domain experts’ feedback, and (iii) systems for synthetic temporal data creation that enable secure sharing of sensitive data between parties. Moreover, even if such methods become available, it is important to describe how those methods can be used in an end-to-end system for predictive maintenance covering both the modeling and operations sides. This thesis analyses and resolves these issues.

The first issue, the lack of probabilistic methods for graph and spatio-temporal graph data, was resolved by connecting graph kernels with stochastic partial differential equations (SPDEs). This method results in a variety of kernels suitable for machine learning problems on graphs, including Matérn, stochastic heat, and stochastic wave graph kernels.

The second issue, the lack of human-in-the-loop methods with domain experts’ explicit feedback, was resolved by developing a decision rule elicitation mechanism and its domain adaptation properties. The method is grounded in human decision-making mechanisms and has been tested in several user studies. It leads to a simple yet effective method for working with domain experts. Next, synthetic data generation was resolved by introducing an open-source software framework called TSGM. This framework effectively generates synthetic time series data and provides a toolkit for evaluation. This work also examined the various approaches to the generation and evaluation of synthetic data.

Finally, the methods proposed in the thesis resulted in successful real-world implementations tested on several large-scale cases with our industrial partner Elisa Oyj. Furthermore, those implementations led to five submitted patents, one of which has already been granted.

This thesis discusses the aforementioned results, places them into a broader perspective, and provides possible avenues for future research.

Keywords  Machine Learning; Deep Learning; Gaussian Processes; Stochastic Differential Equations; Generative Models; Predictive Maintenance; Human-in-the-Loop; Open Source;

ISSN (printed) 1799-4934 ISSN (pdf) 1799-4942
Location of publisher Helsinki Location of printing Helsinki Year 2024
The thesis describes my Ph.D. work at Aalto University. In this thesis, I attempt to resolve some of the most topical problems in machine learning, that prevent the development of effective practical systems such as predictive maintenance systems. This thesis led me to explore different topics and strive to find a balance between operationally effective and theoretically justified methods. By balancing those two extremes, I hope this work is both practical and theoretically interesting.

In addition to this, I sought an answer to the question “How should machine learning research be conducted?” To answer this question, I investigated different ways of developing machine learning methods, such as searching for inspiration in cognitive psychology, creating heuristics and empirically verifying them, developing theoretically motivated principled approaches, and creating practical tools (e.g., open-source frameworks). Even though I still cannot answer the question precisely, this journey led me to a much better understanding of how to choose the right approach to research for particular problems, and I hope to share it through this thesis.

My work was supported by Elisa Oyj, and the Academy of Finland (Flagship programme: Finnish Center for Artificial Intelligence FCAI). In addition, I am extremely grateful to Aalto Science-IT, which provided a fantastic infrastructure for research and experimentation.

I wish to acknowledge my supervisor Prof. Samuel Kaski for providing guidance and advice, which shaped my view of the research process. Our joint work helped me ask deeper questions, think more broadly, and focus on the most significant problems.

I thank my collaborators, colleagues, and friends — Arno Solin, ST John, Diego Mesquita, Nidhi Singh, Letizia Iannucci, Ivan Yaroslavtsev, Ivan Yashchuk, Aleksei Tiulpin, Ayush Bharti, and many others — for our wonderful conversations and curiosity about machine learning and probabilistic methods that brought us together.
Preface

Last, but not least, I thank my family for their infinite kindness, support, and love.

Espoo, January 4, 2023, Alexander Nikitin
Preface 1

Contents 3

List of publications 5

Author’s contribution 7

Abbreviations 9

Symbols 11

1. Introduction 13
   1.1 Applied predictive maintenance problems in telecommunications .................... 14
   1.2 Research questions ...................................................................................... 16
   1.3 Outline ........................................................................................................ 17

2. Graph Gaussian Processes via the SPDE Framework 19
   2.1 Introduction to graphs .................................................................................. 19
   2.2 Graph Neural Networks .............................................................................. 20
   2.3 Graph Gaussian Processes .......................................................................... 21
   2.4 Contribution: SPDE framework for graph GPs ........................................... 23
      2.4.1 Background and overview of the framework ....................................... 23
      2.4.2 Experimental evaluation ..................................................................... 26
      2.4.3 Extensions ............................................................................................ 26
      2.4.4 Relation to PdM ................................................................................... 26
   2.5 Revisiting Research Question 1. ................................................................. 26

3. Human-in-the-loop machine learning with decision rule elicitation 29
   3.1 Motivation for HITL research ................................................................... 29
   3.2 Contribution: Decision Rule Elicitation ..................................................... 31
      3.2.1 Analogy: How can we teach a human? ................................................. 31
3.2.2 Problem setting and method .......................... 31
3.3 Revisiting Research Question 2 .......................... 33

4. Time series generative modeling ......................... 35
  4.1 Synthetic time series generation ........................ 35
  4.2 Contribution. TSGM: generative time series modeling
    framework ............................................. 36
  4.3 Revisiting Research Question 3 ........................ 40

5. Human-in-the-loop predictive maintenance system .... 41
  5.1 Problem overview ..................................... 41
  5.2 Contribution. Human-in-the-loop predictive maintenance
    of workstations ....................................... 42
  5.3 Revisiting Research Question 4 ........................ 45

6. Conclusion and discussion ............................... 47

References ................................................. 49

Publications ............................................... 57
This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s contribution

Publication I: “Non-separable Spatio-temporal Graph Kernels via SPDEs”

I came up with the idea, designed and executed experiments, and was the primary author of the paper. All authors contributed to the text, presentation, and discussions.

Publication II: “Decision Rule Elicitation for Domain Adaptation”

I came up with the idea, designed and executed experiments, and was the primary author of the paper. SK contributed to the idea development and presentation, and the text.

Publication III: “Human-in-the-Loop Large-Scale Predictive Maintenance of Workstations”

I came up with the idea, designed and executed experiments, and was the primary author of the paper. SK contributed to the text and presentation.


I came up with the idea, designed and executed experiments, and was the primary author of the paper. LI contributed to some of the methods in TSGM, and visualizations. All authors contributed to the text and presentation.
Language check

The language of my dissertation has been checked by Aalto Writing Clinic Staff. I have personally examined and accepted/rejected the results of the language check one by one. This has not affected the scientific content of my dissertation.
Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Approximate Bayesian computation</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial intelligence</td>
</tr>
<tr>
<td>BO</td>
<td>Bayesian optimization</td>
</tr>
<tr>
<td>DL</td>
<td>Deep learning</td>
</tr>
<tr>
<td>DRE</td>
<td>Decision rule elicitation</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative adversarial network</td>
</tr>
<tr>
<td>GNN</td>
<td>Graph neural network</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>HITL</td>
<td>Human-in-the-loop</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long short-term memory</td>
</tr>
<tr>
<td>ML</td>
<td>Machine learning</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum likelihood estimation</td>
</tr>
<tr>
<td>NN</td>
<td>Neural network</td>
</tr>
<tr>
<td>PdM</td>
<td>Predictive maintenance</td>
</tr>
<tr>
<td>PML</td>
<td>Probabilistic machine learning</td>
</tr>
<tr>
<td>SDE</td>
<td>Stochastic differential equation</td>
</tr>
<tr>
<td>SPDE</td>
<td>Stochastic partial differential equation</td>
</tr>
<tr>
<td>VAE</td>
<td>Variational autoencoder</td>
</tr>
</tbody>
</table>
## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{R}$</td>
<td>Set of real numbers</td>
</tr>
<tr>
<td>$G$</td>
<td>A graph</td>
</tr>
<tr>
<td>$V$</td>
<td>A set of graph vertices</td>
</tr>
<tr>
<td>$E$</td>
<td>A set of graph edges</td>
</tr>
<tr>
<td>$W$</td>
<td>Weight matrix of a graph</td>
</tr>
<tr>
<td>$A$</td>
<td>Adjacency matrix of a graph</td>
</tr>
<tr>
<td>$D$</td>
<td>Degree matrix of a graph</td>
</tr>
<tr>
<td>$\mathcal{N}(\mu, \Sigma)$</td>
<td>Multivariate normal distribution with mean $\mu$ and covariance $\Sigma$</td>
</tr>
<tr>
<td>$w$</td>
<td>White noise</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>Laplace operator</td>
</tr>
<tr>
<td>$L$</td>
<td>Graph Laplacian</td>
</tr>
<tr>
<td>$\dot{u}(t)$</td>
<td>Temporal derivative of a function $u$</td>
</tr>
<tr>
<td>$\ddot{u}(t)$</td>
<td>Second temporal derivative of a function $u$</td>
</tr>
<tr>
<td>$W_t$</td>
<td>Wiener process</td>
</tr>
<tr>
<td>$\text{GP}(\mu, K)$</td>
<td>Gaussian process with mean $\mu$ and kernel $K$</td>
</tr>
<tr>
<td>$E$</td>
<td>Expectation</td>
</tr>
<tr>
<td>$\text{Cov}$</td>
<td>Covariance</td>
</tr>
</tbody>
</table>
1. Introduction

Telecom applications have become increasingly prominent in modern society due to the growth in the number of personal and work devices. This growth has driven the development of novel technologies that connect those devices, such as wireless and cable networks. These technologies are imperative for communication purposes and fundamental to major technological breakthroughs, such as self-driving vehicles and the Internet of Things. It is vital to guarantee the robust and reliable operation of the underlying telecom equipment. Luckily, these technologies generate a large amount of data that can potentially be used by machine learning (ML) algorithms to predict the future behavior of telecom systems. In particular, this thesis considers the problem of Predictive Maintenance (PdM). In a nutshell, PdM aims to arrange maintenance operations in order to ensure that users experience no interruptions or failures due to system malfunction.

Further, this chapter discusses the background of research in ML PdM and demonstrates the connection between the most topical PdM problems in telecommunications and machine learning research fields.

**Background to PdM.** The predictive maintenance literature covers many application areas and modeling approaches. In terms of applications, predictive maintenance has been applied to the energy sector (Venkateswari and Sreejith, 2019), aircraft engines (Austin et al., 2003), medical technologies (Shamayleh et al., 2020), wind turbines (Canizo et al., 2017), the Internet of Things (Casado-Vara et al., 2019), nuclear infrastructure (Gohel et al., 2020), mobile work equipment (Yang et al., 2022), and many other fields. In terms of methodology, the whole spectrum of machine learning approaches has been used in PdM, ranging from support-vector machine (Santos et al., 2015) and random forest (Chen et al., 2018) to deep learning approaches such as autoencoders (Ma et al., 2018), convolutional neural networks (Yoo and Baek, 2018), and recurrent neural networks (Wang et al., 2020). The present thesis aims to develop and research a broader toolset of ML methods applicable to PdM. For instance, it applies probabilistic methods to graph and spatio-temporal graph problems (Chap-
Overview. This thesis describes the work performed by the author in developing ML methods for predictive maintenance. The project, which was completed in collaboration with Elisa Oyj (a Finnish telecommunication company), strived to (i) identify and fill gaps in the research in order to develop effective PdM methods, and (ii) demonstrate how, using these novel methods, practical PdM systems can be implemented. The work has also resulted in several patent applications and granted patents, which are not covered in this thesis.

1.1 Applied predictive maintenance problems in telecommunications

Telecommunications companies build scalable networks and manage large amounts of equipment. Naturally, some parts of their equipment might fail and cause end-user interruptions. These problems can be avoided by implementing proactive actions, for example, using PdM. One approach to developing PdM models based on probabilistic methods is to estimate the probability of a failure or the time until a failure; these values help prioritize equipment with a higher failure risk. To demonstrate the potential problems where probabilistic machine learning (PML) can be applied to telecom problems, we provide several specific examples of such problems occurring in telecommunication operators.

Example 1. Predictive maintenance of cable networks. Cable networks are crucial for telecom operations, as they provide widely-used services, such as cable television and cable internet access (Large and Farmer, 2004). Cable networks consist of connected equipment, such as switches, routers, and Cable Modem Termination Systems (CMTSs) (Large and Farmer, 2004). Each element of the network can break down or cause interruptions to end users when, for example, either attenuation or signal power exceeds certain limit conditions or when a modem falls offline. Thus this problem can be formulated in ML parlance as a spatio-temporal predictive task on graphs, where each node represents a piece of equipment, and the target for the prediction is a value associated with time to failure.

Example 2. Workstation predictive maintenance. Telecommunications companies manage a large number of workstations and servers. These include employees’ computers (including laptops) and servers. To help domain experts identify the most problematic workstations, one possible solution is to use workstation predictive maintenance. Such a PdM system should be capable of effective collaboration with domain experts since domain experts are the primary decision-makers and source of knowledge...
regarding maintenance procedures. To solve this problem with machine learning, the model must work with temporal data and collaborate with domain experts — that is, be explainable and controllable.

**Example 3. Wireless network management.** Wireless technology unites several protocols, such as 4G, 5G, and IEEE 802 (also known as WiFi), for communication without using a physical medium. This example will only consider WiFi networks, though similar issues can occur when using other standards.

WiFi networks can be critically affected by external factors or internal problems which cause the deterioration of the end-user experience. For instance, such deterioration can be induced by interference with other devices or the effects of weather on infrastructure. Connection quality can be improved using a two-step approach. The first step entails predicting or identifying devices or households at the greatest risk of experiencing those issues. The second step involves recommending specific actions to domain experts to avoid connection problems. This problem can be re-formulated as a temporal graph task, similar to Example 1.

**Example 4. Effective data center management and predictive maintenance.** Data centers generate large amounts of data that can be used for more effective management and problem prevention. For instance, Google was able to utilize such data to reduce the energy used for cooling one of their data centers by 40% (Evans and Gao, 2016). Other applications of ML methods for mitigating data center problems include security, control policy, automatic resource allocation (Bodík et al., 2009), and log-based predictive maintenance (Decker et al., 2020). In data centers, the problems are also related to predictive maintenance. For example, we considered the problem of predicting future issues in the networks of devices in our partner’s data center. For each timestamp, a list of measurements was extracted from the servers, and the network topology was fixed and known. This problem is further complicated by two topologies in data centers (virtual and physical). Similarly to Examples 1 and 3, this problem involves predicting temporal signals on graphs (either degradation, or technical characteristics of the nodes in the network).

**A remark about domain experts.** In all the problems listed above, domain experts are present. Domain experts are specialists in the given field who make technical decisions and adjustments to the system. In the case of PdM, they decide on a maintenance strategy, implement some of the actions manually, and automate the maintenance process whenever it is possible. PdM systems based on machine learning should recognize the presence of domain experts. Indeed, if domain experts and Artificial Intelligence (AI) work separately, the actions taken by the system and the experts might become unaligned, or the systems may become challenging to manage.

**Benefits of PdM systems.** The benefits of such systems are tremendous:
they boost organizations’ productivity, reduce workstation downtime, and help avoid critical high-cost problems. Furthermore, since the problem can be generalized to other domains, the methods developed for these tasks can be used in other contexts. For instance, predicting future equipment breakdowns is similar to forecasting problems in health informatics where future health issues are the predictive target.

**Challenges.** It is challenging to develop PdM systems for the tasks listed above for several reasons. First, many of the problems examined in this thesis contain information about network topology, requiring algorithms that effectively work on graphs. Second, domain experts are busy and must be effectively included in decision-making. Their inclusion should be two-fold: it is necessary for them to be able both to affect the model’s predictions and understand the model’s decisions (this property is known as *interpretability* in the ML literature). Third, PdM systems must be resilient to rapid changes in the environment. One such setting is known as a domain shift — when the generative distribution of data switches to a new domain. In this case, the performance of the ML system begins to degrade, and in a PdM setting, it is vital to react to such events. Fourth, in the context of this work, the telecom company not only managed its infrastructure but also provided such services for other businesses. Therefore, it was necessary to develop a scalable solution. Last, when collaborating with the telecom operators with sensitive data, it is necessary to create synthetic substitutes for reproducibility and experimentation because, due to various regulations, it might be impossible to share real data.

### 1.2 Research questions

The practical issues described above can be translated into machine learning research questions as follows:

**Research Question 1.** *Many real-world systems are described with spatial or spatio-temporal signals on graphs. Can a PML method be developed for spatial and spatio-temporal ML problems on graphs that achieves competitive predictive performance while enabling uncertainty quantification and the inclusion of prior knowledge? Is it possible to adopt Gaussian processes for such problems?*

**Research Question 2.** *In order to include domain experts in the decision-making loop, it can be useful to elicit their detailed feedback (instead of using only the correct labels as feedback). Can domain experts be included in the machine learning pipeline to steer the system’s decision-making such that their detailed feedback improves the performance of the ML*

---

1E.g., with respect to recurrent graph neural networks performance.  
2GPs are instrumental in uncertainty-aware prognosis applications, as it was demonstrated in Biggio et al. (2021).
model, specifically predictive performance or the ability to cope with domain adaptation? Moreover, if the answer is positive, it is important to describe the method constructively.

**Research Question 3.** Collaboration with an industry organization often requires organizing secure and effective data sharing. In particular, the applications listed above require temporal data. A natural way to overcome such issues is to generate synthetic time series data. Moreover, it is important to develop a software tool that allows for synthetic time series generation and evaluation. Can synthetic time series data be generated and used for experimentation such that real data can be replaced with synthetic? What metrics can be used to assess the utility of generated data? The answer to this question should include developing an effective software tool for generating and evaluating synthetic data for quality and security.

**Research Question 4.** It is imperative for practice and research to describe how all the components of a predictive maintenance system fit together for production implementation. Building such a system requires the resolution of numerous applied machine learning problems. Therefore, we pose the following research question. Can a production predictive maintenance system based on machine learning with humans-in-the-loop be implemented and scaled to be successful at identifying issues in a real-life scenario? If the answer is positive, it should include a description of such a system and its evaluation.

### 1.3 Outline

Chapter 2 is devoted to discussing graph machine learning with an emphasis on graph Gaussian processes (GPs) and describes our SPDE framework for spatial and spatio-temporal problems on graphs (results of Publication I). Next, Chapter 3 discusses human-in-the-loop approaches and introduces the decision rule elicitation (DRE) ideas. Chapter 4 covers synthetic data generation, which often requires collaboration between industry and research organizations. This chapter presents the software framework developed for synthetic temporal data generation. Chapter 5 discusses the implementation of workstation predictive maintenance based on probabilistic methods. The last chapter (Chapter 6) summarizes the contributions of this thesis and discusses the results obtained.

**Structure.** Chapters 2-5 are devoted to answering specific research questions. This allows readers interested in a particular question to read about it and skip over results that are irrelevant to them. In each of those chapters, **methods, related work, and answers to research questions** are provided. The last chapter summarizes the results of the previous chapters and discusses the broader significance of the presented research.
2. Graph Gaussian Processes via the SPDE Framework

This chapter is devoted to the thesis results on developing PML methods for spatial and spatio-temporal tasks on graphs (Publication I).

2.1 Introduction to graphs

A graph $G = (V,E)$ is a pair of sets: a set of vertices $V$ and a set of edges $E \subseteq V \times V$. Edges might contain additional attributes such as weight or capacity. Since graphs are a natural structure used to represent relational data, they appear in many ML applications, including drug discovery (Jiang et al., 2021), wireless network resource allocation (Eisen and Ribeiro, 2020), and epidemiology (Panagopoulos et al., 2021). The graph formalism extends to more complex objects, such as dynamic graphs (nodes or vertices dynamically change over time) and hypergraphs (where edges can connect an arbitrary number of vertices). Although a number of ML methods have been developed for such structures (e.g., (Pareja et al., 2020) and (Chitra and Raphael, 2019)), this thesis focuses on classical graphs.

Several problem settings have been explored for evaluating graph ML methods. One example of predictive tasks on graphs is semi-supervised node classification. In this setting, some of the graph nodes are labeled, and an algorithm is used to predict the labels for the rest of the nodes. The benchmarks for these problems include the Cora (Sen et al., 2008), CiteSeer, and PubMed citation networks (Sen et al., 2008; Namata et al., 2012) with thousands of nodes. In this thesis, graph problems that do not involve time are termed spatial tasks on graphs. Another graph-based problem type is spatio-temporal tasks on graphs. In this setting, an ML model is employed to predict the values of a function on either all or a subset of nodes over time (interpolation or extrapolation tasks). More formally, the model determines an approximation of a function $f : V \times \mathbb{R} \rightarrow \mathbb{R}$. Such problems can arise, for instance, in network predictive maintenance, where graphs represent the network topology, or in epidemiology, where the problem is to predict the expansion of an epidemic over graphs (see...
Figure 2.1. Schematic depiction of spatio-temporal problems on static graphs. Each node contains temporal features and a target signal, and the graph remains unchanged. The goal is either to interpolate or extrapolate the temporal signal.

Fig. 2.1 for an example).

**Spectral graph methods.** The earliest approaches to ML problems on graphs were based on spectral methods that use the spectrum of the graph Laplacian for modeling. Let us define the graph Laplacian first.

**Definition 2.1.1.** For a graph \( G = (V, E) \) with the weighted adjacency matrix \( W \), the graph Laplacian \( L \) is defined as

\[
L = D_W - W,
\]

where \( D_W = \text{diag}(w_i) \), and \( w_i = \sum_{j:(i,j)\in E} w_{ij} \).

A special case of this definition is the Laplacian for unweighted graphs: \( L = D - A \), where \( D \) is the degree matrix and \( A \) is the adjacency matrix. The analysis of the Laplacian spectrum results in a number of predictive methods, for instance, spectral clustering (Ng et al., 2001) and graph partitioning (Hendrickson and Leland, 1995).

### 2.2 Graph Neural Networks

The successful application of NNs in many domains, including images and texts, has motivated the development of Graph Neural Networks (GNNs). GNNs have achieved state-of-the-art results in semi-supervised graph classification for citation networks and other benchmarks. Notable examples of GNNs include graph convolutional networks (Kipf and Welling, 2017), graph attention networks (Veličković et al., 2017), and personalized propagation of neural predictions (Klicpera et al., 2019). Combining approaches for semi-supervised (or spatial) problems on graphs and recurrent architectures for temporal tasks has led to the development of architectures for spatio-temporal problems on graphs. Examples of such architectures include diffusion convolutional recurrent neural networks (Li et al., 2018),
spatio-temporal graph convolutional networks (Yu et al., 2017), and graph convolution embedded LSTMs (Zhao et al., 2019).

One core idea in building GNNs is a message-passing framework that unifies many GNN architectures. This framework repeats several iterations of message-passing steps, which can be expressed as

$$m_{t+1}^v = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}),$$  \hspace{1cm} (2.2)$$

$$h_{t+1}^v = U_t(h_v^t, m_{t+1}^v).$$  \hspace{1cm} (2.3)$$

Here, $M_t$ represents message functions, $U_t$ are vertex update functions, $h$ are hidden states, $m$ are messages, and $N(v)$ is a set of neighbors of a node $v$. Next, the readout operation calculates a graph embedding

$$\hat{y} = R(\{h_v^T \mid v \in G\}),$$  \hspace{1cm} (2.4)$$

where $R$ is a readout function (Gilmer et al., 2017). Message passing generalizes graph convolutions, and can be used to develop other architectures.

**GNNs from spatio-temporal problems.** Temporal dynamics can be considered on graphs using three strategies: a temporal signal on a dynamic graph, a static signal on a dynamic graph, and a temporal signal on a static graph. We will focus on the latter because the applied problems mentioned earlier (see Chapter 1), which hereinafter are termed spatio-temporal problems on graphs, can be approached using such a strategy. An example of such a problem is the prediction of an epidemic distribution over a graph, where nodes represent cities and edges represent the connections between the cities. For such problems, several neural network approaches have been developed, for example (see, e.g., (Li et al., 2018; Yu et al., 2017; Zhao et al., 2019; Seo et al., 2018; Yu et al., 2018)).

### 2.3 Graph Gaussian Processes

This thesis explores another paradigm applied to graph machine learning problems — graph Gaussian processes.

Gaussian processes (GPs) are a probabilistic framework applicable to many machine learning problems. This approach considers a dataset as a collection of random variables where any finite combination has a joint multivariate normal distribution. Since the distribution is modeled as Gaussian, the posterior’s mean and covariance can be estimated in a closed form. One caveat is the scalability of the method; therefore, several approximation approaches were suggested, e.g., sparse and variational Gaussian processes (Titsias, 2009), stochastic variational inference (Hensman et al., 2013), and low-rank approximations such as random Fourier features (Rahimi and Recht, 2007). GPs are parameterized using kernels (sometimes called a covariance function), which specify a prior distribution over
the space of functions. In continuous domains, popular choices of kernels include radial basis function (RBF, Equation 2.6), periodic (Equation 2.7), or Matérn (Equation 2.8) (Rasmussen and Williams, 2006):

\[ k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}, k(x,x') = \text{Cov}(x,x'), \]  
\[ k(r) = \sigma^2 \exp \left( -\frac{r^2}{2\ell^2} \right), \]  
\[ k(r) = \sigma^2 \exp \left( -\frac{2}{\ell^2} \sin^2 \left( \frac{\pi r}{p} \right) \right), \]  
\[ k_{\text{Matérn}}(r) = 2^{1-\nu} \nu^\nu K_\nu(\gamma)/(\Gamma(\nu)), \quad \gamma = \sqrt{2vr}/\ell, \]  

where \( r = \|x - x'\| \) is the Euclidean distance between \( x \) and \( x' \), \( p \) is the period of the periodic kernel, and \( \ell \) and \( \nu \) are parameters that control length scale and smoothness respectively.

GPs can be extended to other domains by constructing kernels operating on those domains. For example, Beck and Cohn (2017) proposed text processing GPs using string kernels, where the kernel can be constructed by estimating the number of common sub-sequences between two texts (Cancedda et al., 2003). In another example, Girard et al. (2002) applied GPs to multi-step time series forecasting and observed proper uncertainty estimation properties for non-linear time series problems with noisy inputs.

GPs have also been applied to graphs as well. For example, Kondor and Lafferty (2002) introduced random walk and heat kernels to graphs by considering power series in the graph Laplacian (for \( \lambda \in [0,1] \)):

\[ k(i,j) = \left[ \sum_k \frac{\lambda^k}{k!} L^k \right]_{ij} = [e^{\lambda L}]_{ij}. \]
In addition, Ng et al. (2018) proposed a graph Gaussian process (GGP) model which averages 1-hop neighborhood function values to obtain likelihood parameters:

$$h_n = \frac{f(x_n) + \sum_{l \in \text{Ne}(n)} f(x_l)}{1 + |\text{Ne}(n)|},$$

where $h_n$ is the likelihood parameter, $\text{Ne}(n)$ is the 1-hop neighbourhood of the vertex, $f(x)$ is a GP distributed latent function, and $x$ are the multi-dimensional features of the vertices. To name a few other works, Opolka and Liò (2020) applied simple graph convolutions, and Opolka et al. (2022) used spectral graph wavelets to construct graph kernels.

The next section introduces the developed framework that can be used to derive spatial and spatio-temporal kernels on graphs by connecting them to SPDEs.

### 2.4 Contribution: SPDE framework for graph GPs

#### 2.4.1 Background and overview of the framework

Stochastic partial differential equations (SPDEs) are closely related to GPs: the solutions to SPDEs represent Gaussian random fields (GRF). This connection helps to derive useful kernels; in particular, the Matérn kernel can be derived from the SPDE (Whittle, 1963):

$$(2v + 2)^{-\frac{1}{2}} f(x) = w(x),$$

where $f : \mathbb{R}^d \to \mathbb{R}$, $v$ and $\kappa$ are the parameters (they were introduced in the Matérn kernel description above), and $w(x)$ is spatial white noise ($\mathbb{E}[w(x)] = 0, \text{Cov}[w(x), w(x')] = \delta(x - x')$).

The connection between spatio-temporal GPs and SPDEs was explored in Da Prato and Zabczyk (2014); Solin (2016); Sarkka and Hartikainen (2012). For instance, a spatio-temporal GP model can be written as

$$f(x, t) \sim \text{GP} \left(0, K(x, t; x', t')\right),
\quad y = \mathcal{H}_t f(r, t) + \epsilon(t),$$

where $x \in \mathbb{R}^d$ is a $d$-dimensional vector, $t \in \mathbb{R}$ is time, $K$ is a spatio-temporal kernel, $y_i$ are modeling targets, $\mathcal{H}_t$ are measurement functionals, and $\epsilon(t)$ are Gaussian noise vectors of time. The GP model can be transformed into SPDE (Solin, 2016):  

$$\frac{\partial f(x, t)}{\partial t} = \mathcal{F} f(x, t) + \Sigma w(x, t),
\quad y = \mathcal{H}_t f(x, t) + \epsilon(t),$$

23
where $\mathcal{F}$ is an operator (e.g., a differential operator).

Publication I proposes to expand these ideas to spatial and spatio-temporal problems on graphs (Nikitin et al., 2022). Our framework consists of the following steps:

1. Define an SPDE using prior knowledge about the target data distribution,

2. Convert the continuous SPDE to its graph counterpart,

3. Solve the graph counterpart,

4. Derive the corresponding mean and covariance functions for GPs on graphs.

The SPDE framework for graph kernels is shown schematically in Figure 2.3: To illustrate, we fit the model to some process (which could, for example, be the temperature or epidemic distribution) over the U.S. states. This process can be described with the heat equation (this is prior knowledge). Then, the continuous domain is discretized and represented as a graph. Each node of the graph represents a state, and connections represent adjacencies over the states (note that for a given domain, several discretizations exist, e.g., in this case, a graph of the flights between the states can be used). Then, the discretized SPDE, a system of SDEs, can be solved, resulting in a non-separable kernel. Last, the kernel is used in GPs that are fitted to the data using, for instance, MLE-II.

Now, we describe how SPDEs on graphs can be formulated and solved. First, the framework can be applied to spatial signals on graphs by discretizing the corresponding spatial SPDEs. For example, let us consider the discretization of Equation 2.10 into

$$\left(\frac{2v}{k^2} I + L\right)^{\frac{1}{2}} v = w. \quad (2.11)$$

By solving this equation, we derive Matérn graph kernels (similar to (Borovitskiy et al., 2021) for graphs and (Borovitskiy et al., 2020) for manifolds). For the self-adjoint Laplacian, the resulting kernel is:

$$K = \left(\frac{2v}{k^2} I + L\right)^{-v}. \quad (2.12)$$

This framework is applicable far beyond the derivation of the Matérn kernel. To demonstrate its effectiveness, we apply the framework to the stochastic heat

$$\frac{d u}{d t} = -c \tilde{L} u + \sigma \tilde{W}_t, \quad (2.13)$$

and stochastic wave equations

$$\frac{d^2 u}{d t^2} = -c^2 \tilde{L} u + \sigma \tilde{W}_t. \quad (2.14)$$
Here, $u(t)$ is a spatio-temporal function on a graph, $\tilde{L}$ is the fractional graph Laplacian $\tilde{L} = \left( \frac{2\nu}{\pi^2} I + L \right)^{\frac{1}{2}}$, and $\tilde{W}_t$ is the formal derivative of the Wiener process $W_t$.

The discretization of SPDEs on graphs, where the Laplace operator is replaced with the fractional graph Laplacian, results in systems of SDEs. This system can be solved in closed form and results in kernels for spatio-temporal processes on graphs. These results are presented in Propositions 2 and 3 in Publication I:

### Stochastic Heat Equation Kernel (SHEK):

\[
 u(t) \sim \text{GP}(\mu(t), \text{Cov}[u(s), u(t)]), \\
 \mu(t) = e^{-cL_t} u(0), \\
 \text{Cov}[u(t), u(s)] = \frac{\sigma^2}{c} e^{-c\tilde{L}_t - c\tilde{L}_s} \left( e^{c\tilde{L}_s} \tilde{L}_s \right)^{-1} - I \left( \tilde{L} + \tilde{L}^\top \right)^{-1}. \tag{2.15}
\]

### Stochastic Wave Equation Kernel (SWEK):

\[
 u(t) \sim \text{GP}(\mu, \text{Cov}[u(s), u(t)]), \\
 \mu(t) = \frac{1}{c} \tilde{L}^{-\frac{1}{2}} \sin(\Theta t) \hat{u}(0) + \cos(\Theta t) u(0), \tag{2.16}
\]
Graph Gaussian Processes via the SPDE Framework

\[ \text{Cov}[\mathbf{u}(s), \mathbf{u}(t)] = \sigma^2 \Theta^{-2} \left( \cos(\Theta(t-s)) \min(t,s) - \frac{1}{2} \cos(\Theta \max(t,s)) \sin(\Theta \min(t,s)) \right), \]

where \( \Theta = c \sqrt{L} \) and \( P \) is defined using the diagonalization of the fractional Laplacian matrix:

\[ \widetilde{L} = P^{-1} \tilde{L}_d P. \quad (2.17) \]

In Publication I, the generalization of SHEK to matrix scaled noise are presented, solving the following system of SDEs:

\[ \frac{d\mathbf{u}}{dt} = -c \tilde{L} \mathbf{u} + \mathbf{W}_t. \quad (2.18) \]

2.4.2 Experimental evaluation

We evaluated the methods described above using synthetic data generated from physical processes (heat and wave distribution over a surface) and epidemiologic data on the distribution of COVID-19 in the US and chickenpox in Hungary. In our experiments, the derived kernels performed better than other approaches to spatio-temporal graph GP modeling (for instance, product separable kernels) and spatio-temporal graph neural networks (in particular, Diffusion Convolutional Recurrent Neural Network (Li et al., 2018)).

2.4.3 Extensions

The framework can be applied to other SPDEs, resulting in more novel kernels for spatial and spatio-temporal graph problems. Moreover, graph SPDEs can be numerically solved using approximate solvers (e.g., algebraic multi-grid), which could enable faster sampling from GPs.

2.4.4 Relation to PdM

Developing probabilistic methods for spatio-temporal signals on graphs is an important problem in PdM. In particular, such problems arise in the applications of PML to PdM of equipment connected in a network, such as cable or wireless network cases presented in Chapter 1. Here, we choose Gaussian processes, a principled probabilistic approach that can be applied to many PdM problems. This chapter’s results expand kernels to spatio-temporal graphs, making GPs, in particular, applicable to the practical network cases mentioned above.

2.5 Revisiting Research Question 1.

— Can a PML method be developed for spatial and spatio-temporal ML problems on graphs that achieve competitive predictive performance? Is it
possible to adopt Gaussian processes for such problems?

**Answer.** The answer to this question is in the affirmative, and Publication I proposes an approach to developing kernels for spatial and spatio-temporal problems on graphs. The resulting kernels can be used in Gaussian processes, and are competitive to graph neural networks.

**Details.** Publication I proposes an SPDE-based framework for deriving graph kernels for spatial and spatio-temporal problems on graphs. To demonstrate the effectiveness of the proposed framework, several kernels were derived, namely Matérn, stochastic heat, and wave equation kernels. Experiments have shown the applicability of this approach to a wide range of ML tasks; for example, Matérn kernels are applicable to citation networks problems, and spatio-temporal kernels derived from stochastic heat and wave equations are practical, for example, for epidemic distribution tasks on graphs and physical modeling. Moreover, this approach estimates uncertainties of the prediction, which can help in the decision-making process in critical tasks such as predictive maintenance and, in some cases, graph GPs offer advantages over GNNs. Additionally, the approach utilizes expert human knowledge because the SPDE can be chosen based on prior knowledge about the modeled process.
3. Human-in-the-loop machine learning with decision rule elicitation

This chapter describes how domain experts can be effectively included in the machine learning loop by introducing the decision rule elicitation method (Publication II).

3.1 Motivation for HITL research

In the standard approach to ML, humans (e.g., users or domain experts) are only included in the modeling process to a limited extent. However, this causes a lack of efficiency and effectiveness in model interaction and decision-making. Indeed, two major concerns are that (i) the goals of humans and AI become unaligned, and (ii) humans and AI cannot exchange information. The latter means that the results of the models are not interpretable by humans and, conversely, feedback from humans cannot inform the model. This chapter contributes to resolving these issues by proposing a method for HITL ML called decision rule elicitation (DRE), which combines a data-driven approach with decision rules elicited from human experts.

**Background.** The volume of recently published research in human-in-the-loop ML underscores the importance of discussing humans in the ML context. For example, Holzinger (2016) argues that applying pure ML algorithms to healthcare is unrealistic, at least in the near future, because of their limited extrapolation capabilities. To avoid this problem, Lage et al. (2018) use human experts to optimize their models for interpretability.

Human-in-the-loop ML research focuses on effective collaboration between humans and AI systems. Human feedback is most commonly used in the form of labeled data, for instance, in active learning (Settles, 2009). This approach is, however, severely limited, and interactive ML studies more effective means of interaction between experts and models (Amer-shi et al., 2014). Some approaches introduce human interaction to the reinforcement learning setting (Christiano et al., 2017) and use partially observable Markov decision processes (POMDP) for modeling human-
Human-in-the-loop machine learning with decision rule elicitation (Lam and Sastry, 2014). In turn, HITL methods have been employed in self-driving cars (Wu et al., 2021), medical image analysis (Budd et al., 2021), predictive maintenance systems 5, and drug design (Sundin et al., 2022). Our approach to human-in-the-loop systems allows the transfer of more knowledge from domain experts to models. Moreover, it renders the decision-making process more interpretable. It does not replace other methods but can be used in combination with them.

**Problem Setting.** The goal of the current research problem is to find a method that collaborates more effectively with humans (in the case of predictive maintenance, those humans are domain experts in maintenance). The proposed method should be interactive and elicit useful knowledge from domain experts. The problem is not restricted in any other way. Thus, the method should be applicable to regression, classification, and other ML tasks.

In the PdM context, the utility of such a method could be substantial. Models in this domain are often utilized by domain experts — it is they who take the final decision and perform the maintenance operations, so both generalization ability and interactivity are significant in this application. Such a method could also be useful in a lifelong learning setting (Chen and Liu, 2018) and help with domain shifts due to seasonal or intrinsic system factors.
3.2 Contribution: Decision Rule Elicitation

3.2.1 Analogy: How can we teach a human?

This analogy is only intended to explain the motivation for our method and help the reader to intuitively understand it. Let us think about teaching a child to separate photos of cats and dogs. For this problem, sophisticated explanations are ineffective, and the simplest way to teach the child is to demonstrate several examples and let them practice. Suppose we have shown several training examples with the correct answers, and now we are demonstrating training examples one by one and asking the child whether it is a cat or a dog. We observe that the child hones intuition through higher exposure. Occasionally, more complicated examples occur, and the child makes a mistake. What would be the most effective method to help the child learn from this example? By analogy with machine learning, one way is to reveal the correct answer for each example. However, this method could potentially be improved because it contains no information on the aspects of the photo that are useful for making a decision. Moreover, it does not contain any information on how the decision was made using this information (e.g., that the decision was taken by observing the shape of the paws and ears). Surprisingly, even though it is suboptimal to provide only the correct answer without additional information, this approach is prevalent in machine learning. A more effective method for a teacher is to explain why the prediction was wrong during the training process by providing an explicit explanation. Consequently, the child can use such explanations later when they observe a novel confusing sample. This idea is the core for constructing the proposed framework. The teacher explains those cases where the model was wrong. It allows the model to build data-driven “intuition” as well as explainable rules for the most confusing samples.

3.2.2 Problem setting and method

Overview. While working with industry experts, we have noticed that they often can explain their decision-making with simple, heuristic yet generalizable decision rules. These rules can thus be utilized to build an interactive model with the following algorithm: (i) build a data-driven model for future breaks prediction, (ii) run the model in the production, (iii) collect wrong predictions and send them to domain experts, (iv) request domain experts to explain their decision-making process for incorrectly predicted samples via decision rules, and (v) use the elicited decision rules in combination with the data-driven model to improve the system. Intuitively, this algorithm can use decision-making rules from domain
experts, provided they are sufficiently generalizable, to correct the blind spots in a data-driven model. Those heuristic rules are applied later to new data points and provide interpretable explanations. The ability of domain experts to operate with heuristic rules is supported in psychological literature, where experts in various fields explain their thinking process with such heuristic rules (Gigerenzer and Todd, 1999; Green and Mehr, 1997).

**Description of DRE.** Here, the method is described more formally. Let us consider a supervised ML problem with a dataset \((X, y)\), where \(X \in \mathbb{R}^{n \times d}\), \(n\) is the number of samples, \(d\) is the number of features. For the sake of brevity, we do not define the domain of \(y\), but it could be any target, such as regression or classification. \(M(x, \theta_M)\) is an ML model, parameterized by \(\theta_M\), \(x \in X\). Parameters \(\theta_M\) are learned from the data. The feedback model is built by combining the elicited decision rules \(f_i\) as follows:

\[
C_{fb}(x) = \zeta \left( \sum_{i=1}^{F} f_i(x) \right) \sim(X_{test}^i, x, \theta_f^i), \quad (3.1)
\]

where \(\zeta\) is a smoothing function, and \(F\) is the total number of the elicited rules.

The weight \(\alpha(x)\) between the data-driven and the elicited knowledge components helps to continuously average between those components. At the limit, it switches the resulting model to one of the components. It is especially beneficial when the algorithm operates under domain shifts because human knowledge is often robust to domain shifts, and a model trained on historical data is not. The weights for each of the rules can be learned from the data or set from prior knowledge, for example, from the experience of domain experts.

**Experimental results.** The study evaluated DRE with synthetic data, sentiment analysis tasks, and by running a predictive maintenance system based on this algorithm. Additionally, user studies were conducted to verify that the methods were applicable in those settings with real users. Both better domain adaptation and better performance were observed. Moreover, we found it practical to use DRE for the production implementation of predictive maintenance systems; see Chapter 5 for a comprehensive study. The approach can be beneficial in many other PdM applications where domain adaptation is of critical importance, such as in Nejjar et al. (2023).
3.3 Revisiting Research Question 2

— Can domain experts be included in the machine learning pipeline to steer the system’s decision-making such that their detailed feedback improves the performance of the ML model, specifically predictive performance or the ability to cope with domain adaptation? Moreover, if the answer is positive, it is important to describe the method constructively.

**Answer.** Yes, Publication II proposes a method that includes domain experts in the loop and improves both predictive performance and domain adaptation properties (Nikitin and Kaski, 2021). Moreover, the method can be combined with arbitrary data-driven approaches, making it applicable to a broad set of ML scenarios.

**Details.** Publication II demonstrates that decision rule elicitation is a convenient and practical method for organizing AI interaction with domain experts. It delivers improved performance in domain adaptation and allows domain experts to steer the decision-making process of ML models. This approach is easily scalable and can be adjusted on the fly, which supports lifelong learning and robustness under domain shifts. The approach was experimentally evaluated using the modeling of domain expert knowledge employing shallow decision trees with restricted access to historical information and user studies. Moreover, Publication III studies the application of this method to a large-scale predictive maintenance case and demonstrates its practical applicability.
4. Time series generative modeling

4.1 Synthetic time series generation

This chapter discusses synthetic time series generation — a crucial research direction for industrial ML problems (Publication IV). Synthetic data are generated rather than collected, which enables the sharing of realistic datasets without violating privacy. The primary objectives of synthetic data generation are sharing synthetic counterparts instead of sensitive real data and the augmentation of real data. Each of these aims is considered in more detail below.

**Objective 1. Data Sharing.** Today, data are subject to many regulations. These regulations include the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and the California Consumer Privacy Act (CCPA). To be able to collaborate, research and industrial organizations have, therefore, to establish data-sharing processes that do not reveal private information. One possible approach to this problem is sharing synthetic data instead of historical data.

**Objective 2. Augmentation of real data.** Modern ML models, in particular DL models, require extensive datasets for their training. Thus, augmenting these datasets with synthetic data can enhance the predictive performance and robustness of DL systems (Shorten and Khoshgoftaar, 2019).

As the majority of predictive maintenance tasks considered here require the temporal modeling of a system, the present research is interested in the generation of synthetic time series; henceforth, the focus will be on this problem.

**Background.** A natural probabilistic approach to the generation of synthetic time series data is *generative models*. Generative models, in contrast to discriminative ones, learn the probability density of a conditional distribution of observable data \( p(x|c) \). Here, \( c \) can be either the
target or additional covariates (e.g., image generation using texts (Reed et al., 2016)). Several approaches have been developed to train generative models, including diffusion models, energy-based models, and generative adversarial networks (Murphy, 2023). Many previous studies have discussed generative modeling and synthetic data generation in the context of time series. Examples of such works include autoregressive modeling (Dunne, 1992), generative adversarial networks (Esteban et al., 2017; Mogren, 2016), Gaussian processes (Roberts et al., 2013), and recurrent neural networks (Graves, 2013). Yoon et al. (2019) proposed timeGAN — a method for generative time series modeling, where the model is trained by optimizing jointly supervised and adversarial losses through a joint embedding space. Other architectures developed specifically for time series generative modeling include Esteban et al. (2017) and Xu et al. (2020). Application-wise, generative time series methods have been employed in several domains, including medical (Esteban et al., 2017), audio (Donahue et al., 2019), and financial data (Takahashi et al., 2019) contexts.

Some of the frameworks provide limited functions related to time series generation. For instance, DeepEcho\(^1\) (an extension of a framework for tabular synthetic data generation (Patki et al., 2016), which was proposed in Zhang et al. (2022)) implements probabilistic autoregression and basic GAN for time series generation or Maat et al. (2017) provides autoregression models and GPs. The software framework developed in this work enables broader functionality: (i) it generates time series data conditioned on scalar or temporal variables, (ii) it provides a broad set of evaluation metrics, (iii) it supports both time series generation and augmentation scenarios, and (iv) it includes not only data-driven generators (such as GANs, VAEs, TimeGAN, and Gaussian processes) but also simulator-based approaches that can be trained using likelihood-free inference. Moreover, TSGM model optimizers can be easily switched to differentially private versions, which facilitates effortless privacy research and experimentation.

### 4.2 Contribution. TSGM: generative time series modeling framework.

Publication IV developed TSGM,\(^2\) an easy-to-use framework for time-series generative modeling (Nikitin et al., 2023). This framework allows a user to generate and evaluate the quality of synthetic data in just a few lines of code. Moreover, this framework extends beyond generative models and provides several simulator-based approaches. A versatile set of metrics for evaluation is introduced and implemented in the framework. In this framework, we differentiate time series modeling approaches into the

---

\(^1\) [https://github.com/sdv-dev/DeepEcho](https://github.com/sdv-dev/DeepEcho)

\(^2\) [https://github.com/AlexanderVNikitin/tsgm](https://github.com/AlexanderVNikitin/tsgm)
following categories: dataless simulator-based, data-driven, and simulator-based. Next, each of them is briefly described.

**Dataless simulator-based generators.** This type of generator offers a way to produce synthetic time series data when exemplary data are unavailable, but a program can simulate the underlying dynamic. For example, such generators include stochastic processes or equipment simulators. Since historical data are unavailable, this type of generator cannot employ a training procedure, and thus it is rare in the ML literature. However, one example of a dataless simulator-based generator is a simulator based on survival analysis used in Publication III or the simulator for spiking neural networks developed in Goodman and Brette (2009). This generator type is significant for applied problems since it enables experimentation when data collection is challenging or impossible.

**Data-driven generators.** This type of generator makes no prior assumptions about the underlying dynamics and uses generative models to learn such dynamics solely from the data. Examples of this type include GANs, Variational Autoencoders (VAEs), or neural processes (NPs). The framework implements many data-driven methods and utilities for monitoring and visualizing them.

**Simulator-based generators.** The last type of generator combines ideas from the previous two — both data and a simulator process are available. Here, the simulator is parameterized, and the data are available, and the goal of a training procedure is to estimate those parameters from the data. One example of this type of generator is approximate Bayesian computation (ABC) for inferring simulator parameters. Moreover, the framework can be easily combined with Optuna (Akiba et al., 2019), a tool for hyperparameter optimization, to find optimal simulator parameters (this integration is exemplified in TSGM tutorials).

**Evaluation metrics.** Publication IV systematizes and implements several approaches to the evaluation of synthetic time series data. For instance, it proposes to compare the generated data by **similarity**, **predictive consistency**, **privacy**, **downstream effectiveness**, **fairness**, **diversity** metrics, and compare the datasets qualitatively. **Similarity** metric measures how similar the generated data are to historical data samples, for instance, by calculating the distance in the space of summary statistics. **Predictive consistency** metric measures whether downstream models from a fixed set give consistent results on synthetic and real data. **Privacy** metric determines whether generated data are private. For instance, the framework measures the precision of membership inference attacks (Shokri et al., 2017). **Downstream effectiveness** measures the effectiveness of the generated data for a given downstream model. There are several specific approaches to assessing downstream effectiveness. For example, it can measure performance gain if training data are augmented with synthetic data, or alternatively, if an algorithm
is trained on synthetic data, the performance gain can be evaluated on real data. **Fairness** metrics aim to measure the effect of synthetic data on perpetuating or creating harmful biases in downstream ML applications. **Diversity** compares synthetic time series and the reference samples using, e.g., the Kolmogorov-Smirnov two-sample test. Finally, TSGM provides a set of utilities for synthetic time series visualization — users can conveniently compare the visualized time series **qualitatively**. These evaluation metrics help to measure the utility of synthetic data both for data-sharing tasks and for augmenting real data.

The framework is concise and provides high-level abstractions, which help practitioners to easily adapt it to custom problems (see Listing 4.1 for a brief example).

**Figure 4.1.** t-SNE embeddings of synthetic data generated by different methods. Such visualizations can be easily generated using TSGM. From Publication IV.

Experimental evaluation. The framework was evaluated on various open datasets. The methods implemented in the framework achieve high performance both quantitatively and qualitatively. As an example of qualitative evaluation, Fig. 4.1 presents a t-SNE visualization of periodic data and synthetic data for three implemented methods: TimeGAN, RGAN, and VAE. The generated data proved to be similar to the original data, especially with VAEs. A brief example of quantitative evaluation is provided in Table 4.1; see Publication IV for complete details. A minimal example of a time series generation code using recurrent GAN via TSGM is demonstrated in Listing 4.1.
Table 4.1. Sines dataset experiments. 100 samples of sine data were generated using a training dataset of 5000 periodic functions. Here distance is \( L_2 \) distance in the space of summary statistics, downstream gain is the MSE gain on the autoregression task, achieved by augmenting training data with synthetic data; consistency is the fraction of pairs of downstream models with consistent performance on synthetic and real data; and privacy is one minus precision of a membership inference attack method. From Publication IV.

<table>
<thead>
<tr>
<th>Method</th>
<th>Distance</th>
<th>DG(\text{MSE}(1\text{e-2}))</th>
<th>Consistency</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>(2.52 \pm 0.03)</td>
<td>(0.00 \pm 0.00)</td>
<td>(0.67 \pm 0.00)</td>
<td>(0.49 \pm 0.00)</td>
</tr>
<tr>
<td>GAN</td>
<td>(7.01 \pm 0.31)</td>
<td>(-0.03 \pm 0.03)</td>
<td>(0.83 \pm 0.17)</td>
<td>(0.87 \pm 0.00)</td>
</tr>
<tr>
<td>TimeGAN</td>
<td>(2.84 \pm 0.05)</td>
<td>(0.00 \pm 0.00)</td>
<td>(0.33 \pm 0.33)</td>
<td>(0.41 \pm 0.00)</td>
</tr>
</tbody>
</table>

Listing 4.1. A minimal example of time series generation via TSGM using a conditional GAN model.

```python
import tsgm

# ... Define hyperparameters and read data ...

arch = tsgm.models.architectures.zoo['cgan_base_c4_l1'](seq_len=seq_len, feat_dim=feature_dim, latent_dim=latent_dim, output_dim=0)

gan = tsgm.models.cgan.GAN(
    discriminator=arch.discriminator,
    generator=arch.generator,
    latent_dim=latent_dim
)

gan.compile(
    d_optimizer=keras.optimizers.Adam(learning_rate=3e-4),
    g_optimizer=keras.optimizers.Adam(learning_rate=3e-4),
    loss_fn=keras.losses.BinaryCrossentropy(from_logits=True),
)

result = gan.generate(10)
```

Performance. TSGM is compatible with hardware accelerators. The experimental evaluation includes experiments with various hardware accelerators and compares the performance of synthetic time series generation using CPU, GPU, multiple GPU, and TPU training.

Production use. As a part of the experimental evaluation, feedback was collected from the industrial partner on the amount of time the proposed framework saves compared to implementing the target methods from other sources. The time savings were estimated in months.

The method presented above, and the experimental results are sufficient to answer the research question #3.
4.3 Revisiting Research Question 3

— Can synthetic time series data be generated and used for experimentation such that real data can be replaced with synthetic? What metrics can be used to assess the utility of generated data? The answer to this question should include developing an effective software tool for generating and evaluating synthetic data for quality and security.

**Answer.** With the framework developed in Publication IV, synthetic time series data can be generated and used for experimentation such that real data can be replaced with synthetic data. Publication IV developed a multilateral set of metrics for evaluating synthetic time series that was found to be effective in offline experiments and production use.

**Details.** Publication IV enabled effective time series generation with TSGM. Performance was assessed by experiments using multiple time series datasets. Those experiments demonstrated that the proposed framework was able to generate time series data for various downstream tasks, including but not limited to predictive maintenance. According to feedback collected on the production use of the framework, it saves a significant amount of time for machine learning practitioners.

Moreover, Publication IV proposes and implements diverse metrics to evaluate synthetic time series data, including similarity, consistency, privacy, fairness, and downstream performance. These metrics can assist and guide researchers toward creating private and effective synthetic time series data for augmentation and synthetic data sharing.
5. Human-in-the-loop predictive maintenance system

This chapter discusses the practical implementation of a human-in-the-loop predictive maintenance system for workstations that was implemented within the partner’s organization’s infrastructure (Publication III). Here, the objective was to make the system based on our research scalable, accessible to domain experts, and effective at spotting real workstation malfunctions. This chapter also describes the technical aspects of creating such a system, including the infrastructure and deployment process.

5.1 Problem overview

Workstations (laptops, computers, and servers) can experience breakdowns and interruptions that consequently may cause problems in the downstream tasks of their users. That situation can be avoided by employing PdM systems to resolve the arising issues in advance. The benefits of such systems include improved safety, better cost-efficiency, and higher reliability of both workstation service and downstream applications. One approach to developing such a system is predicting future failures and recommending action to maintenance personnel. In technical parlance, the system must collect temporal data from a set of workstations and predict which of the workstations are likely to fail in the observable future. In practice, the temporal data are extremely fickle and contain missing values, heterogeneous data types, and noise. In addition, in the context of the present research, it should be noted that it was necessary for the system to be applicable to multiple domains. This is because the PdM system was required to work with numerous customers with diverse workstation types and modes of workstation use.

Below, an overview of the field and the solution is presented. More technical details can be found in Publication III.

Background Some studies have focused on the incident management of workstations. For instance, Potharaju et al. (2013) and Zhou et al. (2017) aimed to improve response time by extracting knowledge from the incident
description. By contrast, the primary goal of the current research is to predict future failures.

Methodologically, recent research in machine learning for PdM recently has been focused on DL (Fink et al., 2020), for example, autoencoders convolutional neural networks (Yang et al., 2019) and recurrent neural networks (Guo et al., 2017; Shen et al., 2021). Earlier works also explored other methods for failure prediction, for instance, SVM (Fulp et al., 2008) or logistic regression (Liao et al., 2006). Other relevant works are listed in Chapter 1, in the section on the background of PdM. Even though DL approaches were sound in our cases, they were not sufficiently interpretable and transparent to be deployed in a real-life environment. Moreover, previous works have not explored the role of domain experts and their interaction with a PdM system.

5.2 Contribution. Human-in-the-loop predictive maintenance of workstations.

In Publication III, an end-to-end solution to this problem is described (Nikitin and Kaski, 2022). On a high level, the system performs the following steps:

1. Training. Beforehand, on historical data:
   1.1. Optimization of data preparation hyperparameters. In particular, the system finds the optimal values for time intervals that are used for aggregating data and target variables.
   1.2. Preparation of an ML dataset from collected historical data. The data are aggregated into a classification ML dataset.
   1.3. A model is trained to predict the probability of a problem happening in a given time horizon.

2. Inference and interaction. Iteratively, during operation:
   2.1. Running the ML model on the most recent data snapshots and providing results to domain experts. The results are transferred to a domain expert as a ranked list of computers by the probability of a future break in a given time period.
   2.2. Eliciting expert feedback as decision rules. The experts provide their feedback for those samples that the model incorrectly classified.
   2.3. Storing the elicited decision rules are stored in a database and using them for continuous improvement of the model.

General notes. Publication III frames the PdM problem as a binary classification problem — the algorithm aims to predict whether a particular workstation will fail in a given time period in the future. Initially,
several experiments were performed with a regression problem (predicting remaining useful life), but the classification problem formulation led to slightly better results.

**Proxy for a target.** In PdM applications, data are often imbalanced. Indeed, the number of samples where a system functions properly is higher than the number of samples that describe malfunctioning. In this case, instead of using a reported malfunction as a target, the number of warnings and errors experienced by a person at a particular moment was used as a proxy. Samples with more warnings than a selected threshold were marked as problematic, and others as non-problematic. Threshold selection was performed by running offline experiments with various thresholds and adjusting the selected threshold to business needs.

**Training.** The training process consists of three steps. The first step involves finding the hyperparameters of the data preparation process and the ML algorithm. These hyperparameters include how much temporal data to use before the prediction point and the prediction horizon. In order to find those hyperparameters, the system employs Bayesian optimization (BO). Then, in the second step, the system preprocesses raw data with the selected hyperparameter values. It aggregates various measurements related to the workstations (e.g., memory usage or CPU load) over the selected by the previous step periods and uses aggregated statistics as features (e.g., percentiles). Last, the ML model is trained on the preprocessed data. The best results were achieved by ensembles such as gradient boosting (Friedman, 2001) and extremely randomized trees (Geurts et al., 2006).

**Inference.** During the inference process, the model serves predictions to domain experts and elicits their decision rules as it was described in Chapter 3 and more detail Publication II. It applies the model to the most recent snapshot and provides a ranked list of workstations with additional information on selected decision rules. The domain experts then analyze these data and return feedback in the form of decision rules for incorrect predictions. This work expands DRE to rules that return probability.

**Communication with domain experts.** Communication with domain experts follows the DRE approach, described in Publication II. Domain experts analyze the model errors, and suggest feedback in the form of decision rules for those samples. Then, those decision rules are used in combination with the data-driven ML component.

**Production deployment.** Publication III describes the service’s production implementation. As an overview, a schematic visualization of the developed services is presented in Fig. 5.1. The architecture is modular and scalable; each logical component is implemented as separate isolated services (e.g., services for data collection, training, and inference), which enables horizontal scaling. Additionally, the quality of this end-to-end system was evaluated in the field. In particular, it should be noted that
Human-in-the-loop predictive maintenance system

Figure 5.1. Visualization of the deployment process. Data collector collects the data from a third-party repository; it stores the collected data in an S3-compatible storage. Those data are used by the training service, which performs the process for training the models described above, and by the inference service, which delivers the predictions in the form of email notifications and to the web interface. Then, by interacting with the web interface, the domain experts perform maintenance actions and submit the decision rules as feedback to the training server. All services are deployed in the form of Docker containers which are built by Jenkins. The image is from Publication III.

the system worked for one year without interruptions, which shows the reliability of such an approach.

Experimental evaluation. The model and service developed in this thesis were experimentally evaluated both in the field and by using historical data. Many real issues with customers’ workstations were identified by the system, and about 50% of daily issues were detected in our experiments. Furthermore, this approach improved predictive performance with an increase in the number of elicited decision rules. The approach proved to be reliable in field tests and worked for over a year, delivering its predictions to domain experts.
5.3 Revisiting Research Question 4

— Can a production predictive maintenance system based on machine learning with humans-in-the-loop be implemented and scaled to be successful at identifying issues in a real-life scenario? If the answer is positive, it should include a description of such a system and its evaluation.

The answer to this question is affirmative, and Publication III shows a possible implementation of such a workstation PdM system. Moreover, this work demonstrates the effectiveness of this approach to PdM problems in the field.

The predictive maintenance system described in this chapter is effective at locating real issues and also performs well in offline experiments. It is scalable, at least to dozens of clients with thousands of workstations, and continuously interacts with domain experts. Moreover, it can indicate issues in synthetic experiments, historical data, and, most importantly, production. Continuous interaction is based on the decision-rule elicitation method presented in Chapter 3, and allows domain experts to adjust and steer the developed system, and effectively include their feedback.
6. Conclusion and discussion

This thesis studies the development of probabilistic methods useful for the most pressing problems in predictive maintenance and beyond. To navigate the research landscape, at the beginning of the thesis, we formulated four research questions connected to practical cases from our industrial partner. Those questions concern (i) creating probabilistic methods for spatial and spatio-temporal graph problems, (ii) developing a method for the effective interaction between domain experts and machine learning models, (iii) enabling synthetic data generation, and (iv) developing an end-to-end system that can help to resolve the issues proactively.

First, the thesis addressed solving spatial and spatio-temporal graph problems with GPs (Publication I). This was achieved by introducing graph SPDEs, solving them, and deriving the kernels from the SPDE solutions. This method allows researchers to tackle graph machine learning tasks using probabilistic methods such as Gaussian processes and incorporates inductive bias based on the choice of an SPDE. Several specific kernels were proposed based on this idea, and, more importantly, we introduced a framework for designing novel kernels.

Then, the thesis moved on to address the inclusion of domain experts in the decision-making loop and proposed a model-agnostic method for interacting with experts using a method dubbed decision rule elicitation (Publication II). The thesis demonstrated that this method improves ML models’ predictive performance, interpretability, and robustness with small overheads for domain experts. Moreover, this method proved beneficial in benchmark experiments and field tests.

Next, the thesis addressed the underlying issue of PdM and other tasks involving sharing data between organizations — the creation of synthetic data counterparts (Publication IV). A framework was developed for synthetic data generation and a suite of evaluation metrics. This tool is a convenient way to create synthetic temporal data and evaluate their quality with only a few lines of code.

Lastly, we described the large-scale implementation of a human-in-the-loop machine learning system for the predictive maintenance of work-
stations (Publication III). The implementation required revisiting the theoretical results from the previous chapters and resulted in the creation of the system that proved to be beneficial in identifying problems in production. This system utilized a broad range of techniques, including decision rule elicitation for the interaction between experts and AI, and it can be considered a go-to example for developing an interactive ML system for predictive maintenance. Moreover, this solution to a predictive maintenance problem can be extended to a broader range of forecasting tasks beyond equipment management. One example of such a problem could be health management, where our approach is applicable (though specific models should be adjusted to fit the case better).

**Impact.** In the broader picture, this work contributes to ML research by creating novel methods for graph tasks, the inclusion of humans in the decision-making loop, and synthetic data generation. It also contributes to PdM research and practice by showcasing an end-to-end PdM system based on human-in-the-loop machine learning. Moreover, this thesis provides an overview of applied tasks and demonstrates how to connect those tasks to problems in ML. The methods developed in this research are applicable beyond PdM and might be utilized in a broader range of cases. For example, similar approaches can be adopted in health informatics, where, instead of equipment malfunction, the target is predicting health issues in patients.

**Ethics.** Possible ethical issues were taken into account in this thesis. For instance, rather than creating a fully autonomous system, domain experts were included in the decision-making loop to develop controllable and interpretable ML methods.

Furthermore, the work on synthetic time series generation highlights the importance of metrics to evaluate the fairness and privacy of the generated data and underlines the possible risks associated with using such data.

**Future works.** The methods presented in this thesis can be extended and improved in future research works. The decision rule elicitation method is a convenient way to organize collaboration between domain experts and AI and can be expanded to natural language human feedback. Moreover, the elicited rules can be studied for alignment and causal properties. The results on kernels via SPDEs show how probabilistic methods can be developed for temporal signals on graphs and can be extended to other popular SPDEs. Moreover, it is interesting to apply this method to more applications. The synthetic time series generation work in the current research proposes several evaluation methods, and some of them should be studied more deeply, for instance, the fairness and privacy metrics. In addition, it would be interesting to extend the end-to-end system for PdM presented in this thesis to other domains, such as healthcare.
References


References


References


Probabilistic methods are key tools for machine learning problems. Even so, there remain many applications where they cannot be applied due to their limitations. These limitations may include the lack of methods for a particular data format (e.g., manifolds, texts, or graphs), data unavailability, or the inability to work collaboratively with human experts.

Inspired by problems in predictive maintenance (PdM), this thesis introduces a set of machine learning solutions that are more generally applicable. It begins with applied tasks in cable networks, data centers, and other telecom applications and indicates the crucial limitations of current approaches: the absence of (i) probabilistic methods for spatio-temporal graph problems, (ii) practical human-in-the-loop methods that learn from detailed domain experts' feedback, and (iii) systems for synthetic temporal data creation that enable secure sharing of sensitive data between parties. Moreover, even if such methods become available, it is important to describe how those methods can be used in an end-to-end system for predictive maintenance covering both the modeling and operations sides. This thesis analyses and resolves these issues.