POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

LEARNING IN NON-STATIONARY ENVIRONMENTS: FROM A SPECIFIC APPLICATION TO MORE GENERAL ALGORITHMS

Doctoral Dissertation of:
Giuseppe Canonaco

Supervisor:
Prof. Manuel Roveri

Co-Supervisor:
Fabrizio Podenzani

Tutor:
Prof. Nicola Gatti

The Chair of the Doctoral Program:
Prof. Barbara Pernici

2021 – Cycle XXXIV

Abstract

In recent years, Machine Learning (ML) has gained a lot of attention and popularity because of its ability to model highly complex phenomena given sufficiently large data sets. This thriving field embraces all the algorithms and techniques able to automatically learn a given task using a finite amount of data that can be thought of as experience. These algorithms are usually paired with some assumptions which may or may not be satisfied in real-world practical applications. For instance, some algorithms assume to have an input signal and its associated output (also called supervised information) for each example in their training data set such that they can learn a mapping from the input signal to the output that generalizes on previously unseen examples. This assumption is not always satisfied in practice, where the output signal could be too expensive to be collected. An example of this mismatch between theory and practice can be found in the context of corrosion prediction for pipeline infrastructures. Here the output signal, corresponding to the presence of corrosion in a given point of the pipeline, is hardly available for the infrastructure of interest due to the incredibly huge cost companies have to bear in order to collect it. Another fundamental assumption associated with ML techniques is about stationary data-generating processes, which implies that the phenomenon we are trying to learn does not change as time passes by. Examples of applications where the stationarity assumption about the data-generating process does not hold are in the context of finance, due to market evolution, in the context of water reservoir systems, due to climate change, in the context of corrosion, because of the wear and tear of infrastructures that increases with time, etc. In all the above-mentioned scenarios, ML techniques cannot be directly applied without softening the assumptions about the availability of supervised information or stationarity they are equipped with. Therefore, in the context of this dissertation, inspired by the specific application needs of corrosion prediction in pipeline infrastructures, we will

investigate ML solutions able to weaken the assumptions of available supervised information and stationarity. Even though the lack of supervised information is a thoroughly researched problem thanks to Transfer Learning (TL), it is overlooked in the context of corrosion prediction requesting tailored solutions for this critical application. Softening the assumption of stationary data-generating processes, instead, is much less studied in lots of different ML sub-fields motivating a much more general investigation within the scope of this dissertation.

The contribution of this dissertation is composed of three main parts.

The first part deals with ML-based techniques for corrosion prediction starting from the data set creation up to the development of predictive models that can circumvent the need for supervised information when it is not available for the pipeline infrastructure of interest. Building a predictive model without the availability of supervised information on the facility of interest while being never considered by the related literature on corrosion prediction is of utmost importance because the collection of such information is incredibly expensive for companies managing such infrastructures and constitutes an additional step toward a more appropriate corrosion prevention through ML-based tools and techniques.

The second part deals with Reinforcement Learning (RL) in non-stationary environments developing two methodological solutions: the former is an active-adaptive approach that can weaken the stationarity assumption in the context of the task the RL agent is currently trying to solve, and it achieves better average returns in presence of a concept drift as opposed to its non-active-adaptive counterpart; the latter is a TL approach for RL able to deal with a time-variant distribution underlying the task generating process that is an overlooked setting in the related TL literature and in which the proposed solution obtains performance improvements in terms of average return over its time-invariant equivalent.

The third part deals with Federated Learning (FL) under non-stationarity and pervasive systems introducing a passive-adaptive approach to mitigate the effect of non-stationary data-generating processes in FL settings, and an ad hoc birdsong detection approach for highly constrained devices at the edge of a pervasive system. The passive-adaptive approach is able to experimentally improve the convergence rate of the learning curve after a concept drift, and, in some cases, even in stationary conditions before the concept drift occurs, w.r.t. its non-adaptive counterpart. Instead, the birdsong detection approach achieves performances in line with the state of the art at a lower cost in terms of computational demand and memory footprint. Furthermore, with appropriate approximations, this last solution can be deployed on real Internet-of-Things (IoT) units.

Sommario

Negli ultimi anni, il Machine Learning (ML) ha guadagnato molta attenzione e popolarità grazie alla sua capacità di modellare fenomeni altamente complessi attraverso l'uso di un insieme di dati sufficientemente grande. Questa fiorente disciplina abbraccia tutti gli algoritmi e le tecniche in grado di apprendere automaticamente un determinato compito utilizzando una quantità finita di dati che possono essere pensati come esperienza. Questi algoritmi sono solitamente associati ad alcune ipotesi che possono essere soddisfatte o meno in contesti reali. Ad esempio, alcuni algoritmi presumono di avere un segnale di ingresso e uno d'uscita (chiamato anche informazione supervisionata) per ogni campione presente nel loro insieme di dati d'addestramento in modo tale da poter apprendere una relazione tra il segnale di ingresso e quello d'uscita che generalizzi su campioni mai visti in precedenza. Questa ipotesi non è sempre soddisfatta nella pratica, dove il segnale di uscita potrebbe essere troppo costoso per essere collezionato. Un esempio di questa discrepanza tra teoria e pratica lo si trova nel contesto della predizione della corrosione per le condutture. Qui il segnale in uscita, corrispondente alla presenza di corrosione in un dato punto della condotta, è difficilmente disponibile per l'infrastruttura di interesse a causa dei costi incredibilmente alti che le aziende devono sostenere per raccoglierlo. Un'altra assunzione fondamentale associata alle tecniche di Machine Learning riguarda la stazionarietà dei processi di generazione dei dati, il che implica che il fenomeno che stiamo cercando di apprendere non cambia con il passare del tempo. Esempi di applicazioni in cui l'ipotesi di stazionarietà del processo di generazione dei dati non regge li troviamo nel contesto della finanza, a causa dell'evoluzione del mercato, nei sistemi di riserva idrica, a causa del cambiamento climatico, nella corrosione, a causa dell'usura delle infrastrutture che aumenta con il passare del tempo, ecc. In tutti gli scenari summenzionati, le tecniche di Machine Learning non possono essere applicate direttamente senza ammorbidire le ipotesi sulla

disponibilità di informazione supervisionata o sulla stazionarietà. Pertanto, nell'ambito di questa tesi, ispirata alle specifiche esigenze applicative della predizione della corrosione nelle condutture, indagheremo soluzioni ML in grado di indebolire le ipotesi di disponibilità di informazione supervisionata e stazionarietà. Anche se la mancanza di informazione supervisionata è un problema studiato a fondo grazie al Transfer Learning (TL), essa viene completamente trascurata nel contesto della predizione della corrosione, che richiede, quindi, apposite soluzioni su misura. L'indebolimento dell'assunzione relativa alla stazionarietà dei processi di generazione di dati, invece, è molto meno studiato in diverse aree del Machine Learning, motivando, quindi, un'indagine molto più generale di questa tematica nell'ambito di questa tesi.

Il contributo di questa tesi è composto da tre parti principali.

La prima parte si occupa di tecniche basate sul Machine Learning per la predizione della corrosione partendo dalla creazione di un insieme di dati rappresentativo del fenomeno corrosivo fino allo sviluppo di modelli predittivi in grado di aggirare la necessità di informazione supervisionata quando essa non è disponibile per la conduttura di interesse. Costruire un modello predittivo senza la disponibilità di informazioni supervisionate sulla struttura di interesse, pur non essendo una problematica mai considerata dalla relativa letteratura sulla predizione della corrosione, rappresenta un obbiettivo di massima rilevanza perché la raccolta di tali informazioni è incredibilmente costosa per le aziende che gestiscono tali infrastrutture e costituisce un ulteriore passo verso una prevenzione della corrosione più appropriata attraverso strumenti e tecniche basati sul Machine Learning.

La seconda parte si occupa dell'appendimento per rinforzo (o Reinforcement Learning (RL)) in ambienti non stazionari sviluppando due soluzioni metodologiche: la prima è un approccio attivo-adattivo in grado di indebolire l'assunzione di stazionarietà nel contesto del compito che l'agente RL sta attualmente cercando di portare a termine e che è in grado di ottenere returns medi migliori rispetto alla sua controparte non attiva-adattiva; la seconda è un approccio TL per RL in grado di gestire una distribuzione tempo-variante alla base del processo di generazione dei compiti che l'agente RL deve portare a termine. Tale approccio costituisce una soluzione per una configurazione che non viene mai considerata dalla relativa letteratura transfer, e in cui l'approccio proposto ottiene miglioramenti delle prestazioni in termini di return medio rispetto alla sua controparte tempo invariante.

La terza parte tratta l'apprendimento federato (o federated Learning (FL)) in sistemi pervasivi caratterizzati da non-stazionarietà introducendo un approccio passivo-adattivo per mitigare l'effetto della non-stazionarietà nei processi di generazione dei dati. Infine, nel contesto di un'applicazione ML per sistemi pervasivi, viene presentato un approccio ad hoc di rilevamento del canto degli uccelli per dispositivi fortemente vincolati. L'approccio passivo-adattivo è in grado di migliorare sperimentalmente il tasso di convergenza della curva di apprendimento dopo una non-stazionarietà, e, in alcuni casi, anche in condizioni stazionarie prima che si verifichi la non-stazionarietà stessa, rispetto alla sua controparte non adattiva. Per quanto riguarda l'approccio di rilevamento del canto degli uccelli, esso

raggiunge prestazioni in linea con lo stato dell'arte ad un costo inferiore in termini di necessità computazionali e di memoria, inoltre, con opportune approssimazioni questa soluzione può essere implementata su unità IoT (Internet-of-Things) reali.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AUC Area Under Curve. 110, 112

CNN Convolutional Neural Network. 88, 97, 98, 103

CUSUM CUmulative SUM. 48, 61, 66

CV Cross-Validation. 11, 20, 22, 26, 36, 110

DL Deep Learning. 102, 103, 105, 106, 109, 110, 113

DMA Dynamic Memory Allocation. 113

DWT Discrete Wavelet Transform. 102

ELBO (negative) Evidence Lower BOund. 70, 73, 130

EMA Exponential Moving Average. 90, 92, 93, 97

FL Federated Learning. 3, 6, 85, 87–92, 94, 98, 117, 118

GMM Gaussian Mixture Model. 103

HMMDP Hidden Mode Markov Decision Process. 48

IoT Internet-of-Things. 3, 86, 87, 102, 105–108, 110, 112, 114, 115

IS Importance Sampling. 34, 51, 52, 54–56

IWCV Importance Weighted Cross-Validation. 10, 39–44, 120, 121

kIWCV k-fold Importance Weighted Cross-Validation. 36, 37

KMM Kernel Mean Matching. 35–37, 44

MDP Markov Decision Process. 2–4, 48, 49, 53, 58, 66, 68, 69, 74

ML Machine Learning. 1–3, 10, 11, 13, 23, 29–32, 47, 85–88, 102, 117

NSD-RL Non-Stationarity Detector for Reinforcement Learning. 51, 53–55, 57, 61–63, 65, 66

PG Policy Gradient. 53, 58

PIG Pipeline Inspection Gauge. 14, 15, 26, 30, 32

PIGs Pipeline Inspection Gauges. 10

RKHS Reproducing Kernel Hilbert Space. 35

RL Reinforcement Learning. 1–5, 47–49, 51–55, 57, 58, 62, 66–70, 74, 75, 79, 80, 85, 117, 118

ROC Receiver Operating Characteristic. 110

SL Supervised Learning. 1–4, 10, 11, 23, 32, 67, 118

STFT Short Time Fourier Transform. 102, 105–107

SVM Support Vector Machines. 25, 27, 29, 30, 38–40, 44

TD Temporal Difference. 69, 76, 133–135

TL Transfer Learning. 2–4, 10, 32, 33, 37, 39, 40, 42–44, 48, 67, 68, 85, 86, 103, 105, 118

UL Unsupervised Learning. 1, 3, 4

CHAPTER 1

Introduction

Machine Learning (ML) has recently become more and more effective in modeling highly complex phenomena given sufficiently large and descriptive data sets. This characteristic makes it a suitable tool for a plethora of applications in the most disparate fields such as: computer vision, robotics, healthcare, natural language processing, transportation, industry etc. ML can be described as the set of all algorithms able to automatically learn to perform a certain task using data which can be regarded as experience. The more the experience the better the algorithm will learn the task.

The ML field can be classically split into three different macro-areas: Supervised Learning (SL), Unsupervised Learning (UL) and Reinforcement Learning (RL). SL techniques deal with problems where the supervised information is available and strive to obtain a function $\hat{y} = f(x)$ whose objective is to correctly predict the output y, called supervised information, associated with the input vector x. UL techniques, instead, strive to learn the underlying structure of the data without having access to the supervised information. Finally, RL techniques strive to find the optimal policy to be executed by an agent on the environment in order to reach a certain pre-specified goal. The optimal policy is learned via an optimization process whose objective is to maximize the long-term cumulative reward, where the reward is what the agent receives upon executing an action on the environment itself.

The algorithms and techniques developed within the huge ML field are equipped with some assumptions which are often not satisfied in practical applications. For instance, SL algorithms assume to have enough labeled data so that predictive models can be properly trained. This assumption does not always hold in real-world scenarios where the labeling process could be too costly or time-consuming. An instance of this setting is in the context of corrosion prediction for pipeline infrastructures, where, usually, the supervised information about the presence of corrosion is hardly available for a pipeline of interest due to the intrinsic cost companies have to bear in order to collect it. For these kinds of applications, where the supervised information is very scarce if not missing at all, standard SL learning techniques are not able to provide good predictors for the objective phenomenon. However, if supervised data is available for some related problem, we could leverage Transfer Learning (TL) which coupled with SL will allow us to alleviate the need of supervised information in the target problem object of interest. TL is a transversal ML sub-field dealing with knowledge transfer across different tasks. In order to be successful, a given TL technique needs to answer three main questions dealing with "what", "how", and "when" to transfer the knowledge across different tasks, which translates into deciding what form of knowledge to be transferred, the appropriate way to transfer it, and, most crucially, when to execute the transfer. The last question is supposed to deal with the negative transfer phenomenon that happens whenever source and target tasks are not sufficiently similar to each other. In the SL context, we talk about inductive TL whenever there is some supervised data coming from the target task, instead, we talk about transductive TL whenever there is only unsupervised data coming from the target task. In both the previously mentioned cases there is plenty of supervised data coming from the source tasks. Finally, we talk about unsupervised TL whenever the supervised information is missing both from the source and target tasks. In this context, clustering or dimensionality reduction problems are usually transferred. On the other hand, for what concerns RL, the situation is a bit more complex since we have an agent-environment interaction where the environment is modeled through a Markov Decision Process (MDP) and the agent by a policy. In this context, TL allows a greater sample efficiency, and transfer algorithms can be distinguished in:

- techniques able to deal with source and target tasks that have different state or action spaces
- and techniques working under the assumption that both state and action spaces will stay the same among the source and target tasks.

The applications of TL are disparate and they span different fields such as robotics, games, natural language processing, healthcare, bioinformatics, recommender systems, corrosion, etc.

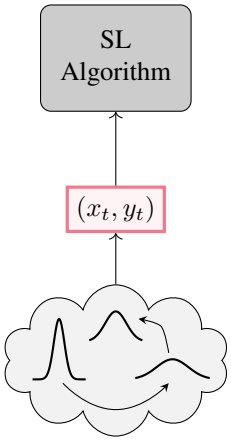
Besides the above-mentioned data-availability requirement, another crucial assumption

tied to ML techniques is about stationary data-generating processes. This means that the phenomenon we are trying to learn does not evolve with time and allows the ML algorithm to converge to a solution for the problem of interest. In the SL context, this translates into the fact that the couples (x, y) always come from the same distribution. For what concern UL, instead of having couples (x, y) we will just have the vector x that still will always come from the same distribution. Finally, in the RL setting, it is the MDP, modeling the environment and the allowed interactions, that will always stay the same, whereas, for what concerns TL, it is the available historical knowledge that does not expose a time-variant structure. However, there are many applications where the above-mentioned assumption about stationarity does not hold, e.g., finance, due to market evolution, robotics, due to faults affecting either sensors or actuators, water reservoir systems, due to the climate change our planet is currently undergoing, corrosion, where the wear and tear of equipment or infrastructures increase with time, etc. In all of these applications ML techniques cannot be directly employed without taking particular care of the time variance at play, that, depending on the particular framework we are using, will affect the distribution generating the couples (x, y) (see Figure 1.1a), the distribution generating the vector x (see Figure 1.1b), the MDP (see Figure 1.1c), or the available historical knowledge (see Figure 1.1d).

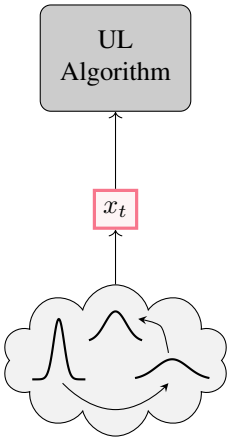
In the context of this dissertation, guided by the specific application needs of corrosion prediction in pipeline infrastructures, we will investigate ML solutions able to weaken the assumptions of available supervised information and stationarity. Despite being a very well researched area thanks to the thriving field of TL, reducing the requirement of supervised information in the context of corrosion prediction is not investigated at all motivating the research of tailored solutions for this critical application. Weakening the assumption of stationary phenomena, instead, is much less studied in lots of ML sub-fields motivating a much more general investigation in the context of this dissertation.

1.1 Original Contributions and Outline

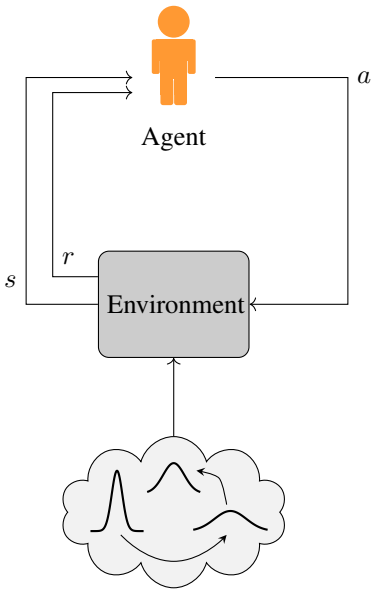
The contribution of this dissertation is threefold. The first one, examined in Part I, deals with learning techniques for corrosion in pipeline infrastructures starting from the data set creation up to devising SL techniques that, with the aid of TL, are able to build a model of the corrosion phenomenon without using supervised information coming from the pipeline of interest. The second one, examined in Part II, deals with RL in non-stationary environments and develops both an active-adaptive approach to cope with changing environments and a TL technique for RL able to take into account an underlying time-variant structure intrinsic to the available historical knowledge. Finally, the third one, examined in Part III, deals with Federated Learning (FL) under non-stationarity and pervasive systems. This last part introduces a passive-adaptive approach to mitigate non-stationarity in FL contexts and a birdsong detection approach able to run on a highly constrained Internet-of-Things (IoT)



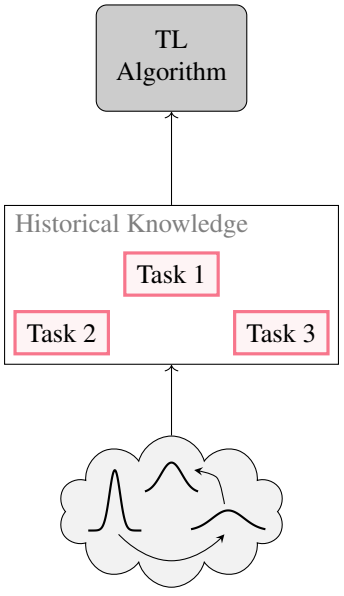
(a) *Non-stationary Supervised Learning: here the data fed to the SL algorithm may be sampled from a different distribution as time progresses.*



(b) *Non-stationary Unsupervised Learning: here the data fed to the UL algorithm may be sampled from a different distribution as time progresses.*



(c) *Non-stationary Reinforcement Learning: here the environment, and hence the MDP, may change as time progresses requiring an adaptation by the RL agent.*



(d) *Non-stationary Transfer Learning: here the historical knowledge that comes from different source tasks and that is leveraged by the TL algorithm may expose a time-variant structure.*

Figure 1.1: *Non-stationary Machine Learning.*

unit.

In Table 1.1, we list all the publications produced in the context of this dissertation.

Table 1.1: *Produced Publications.*

Title	Venue	Reference	Status
Corrosion Prediction in Oil and Gas Pipelines: a Machine Learning Approach	I2MTC 2020	Canonaco et al. [2020b]	Published
A Machine-Learning Approach for the Prediction of Internal Corrosion in Pipeline Infrastructures	I2MTC 2021	Canonaco et al. [2021b]	Published
A transfer-learning approach for corrosion prediction in pipeline infrastructures	Applied Intelligence	Canonaco et al. [2021c]	Published
Model-Free Non-Stationarity Detection and Adaptation in Reinforcement Learning	ECAI 2020	Canonaco et al. [2020a]	Published
Time-Variant Variational Transfer for Value Functions	UAI 2021	Canonaco et al. [2021d]	Published
Adaptive Federated Learning in Presence of Concept Drift	IJCNN 2021	Canonaco et al. [2021a]	Published
Birdsong Detection at the Edge with Deep Learning	Smartcomp 2021	Disabato et al. [2021]	Published

1.1.1 Outline

Part I is composed of three *Chapters*:

Chapter 2 where an introduction to the problem of corrosion prediction in pipeline infrastructures together with a review of the literature are given.

Chapter 3 where we dive deep into building a data set representing the corrosion phenomenon in Oil and Gas pipeline infrastructures followed by devising a corrosion classification model for the produced data set.

Chapter 4 where we shift the attention onto building a corrosion classification model for a given pipeline of interest where the supervised information about corrosion itself is missing, but it is available for one or a set of related pipelines.

Part II is composed of three *Chapters*:

Chapter 5 where an introduction to the problem of non-stationarity in RL is given together with a review of the related literature.

Chapter 6 where we develop two different tools in order to cope with non-stationarity. The first one to detect a change in the environment. The second one that, in view of a detected change, allows the RL agent to promptly react with the aim of bridging the gap in performance induced by the non-stationarity as fast as possible.

Chapter 7 where we devise a variational transfer algorithm for RL able to account for a time-variant distribution underlying the task-generating process which reflects in an intrinsic time-variant structure exposed by the available historical knowledge.

Part III is composed of three *Chapters*:

Chapter 8 where an introduction to both FL and the problem of on-field birdsong detection are given.

Chapter 9 where we describe and develop an adaptive learning rate which is able to passively and adequately update the FL-model's parameters in presence of changes in the data-generating process.

Chapter 10 where we consider the problem of detecting bird songs at the IoT edge and devise a model able to satisfy the highly restrictive constraints on memory and computation imposed by a real IoT unit (namely an STM32 Nucleo H7 board).

Chapter 11 where conclusive remarks on the whole dissertation are given.

Part I

Learning Techniques for Corrosion in Pipeline Infrastructures

CHAPTER 2

Introduction and Related Literature

2.1 Introduction

Internal corrosion in pipeline infrastructures used to transport gas and oil poses various threats to the environment and human beings in terms of both contamination and accidents, which may lead also to explosions in presence of gas leakages [Li et al., 2015]. Therefore, the need to develop mechanisms able to predict the presence of such detrimental phenomenon is of utmost importance.

Unfortunately, this harmful phenomenon is very complex to be modeled or predicted. This is due to the incredible amount of factors it depends on. In carbon-steel pipelines, corrosion may be influenced by CO_2, H_2S, oil or water wetting, steel composition, and internal surface conditions [Nešić, 2007]. Additionally, bacteria activity, fluid dynamics, transport conditions over the entire operating lifespan, as well as geometrical characteristics of the pipeline are certainly important influencing factors [De Masi et al., 2017].

The complexity of the envisaged phenomenon is not the only critical issue for corrosion prediction. In fact, the difficulty in gathering information about all the influencing factors is an additional key concern. Indeed, only the geometrical characteristics of the pipeline and its fluid-dynamic information are usually available, the latter via simulation, and, to make the modeling of this phenomenon even harder, we have an integration incompatibility between these two sources of data [Canonaco et al., 2020b] (this is a crucial

point since a correct integration between them will lead to a reliable representation of the factors influencing the corrosion phenomenon). On the other hand, information about all the other influencing factors is extremely difficult (or often impossible) to be gathered from the field. This is the reason why semi-empirical corrosion prediction models, e.g., De Waard and Lotz [1993], Standard [2005], revealed to be not accurate in real-world corrosion scenarios [De Masi et al., 2017].

In the previously described scenario, ML-based solutions constitute a promising alternative to semi-empirical models. Notice that there have been some solutions proposed in the literature trying to model the corrosion phenomenon via ML-based tools, but usually the data set used to do this are very small or the performance evaluation has not been properly carried out questioning the generality of the obtained results (see Section 2.2). Furthermore, the major critical point of ML solutions available in the literature is that, following a SL approach, they require information about the presence of corrosion in a given pipeline. Such information is gathered directly from the field employing Pipeline Inspection Gauges (PIGs) that are robot inspection systems measuring the presence of corrosion within a pipeline. In other words, the main limitation of such supervised approaches for corrosion prediction is that a model is built only when information about the real presence of corrosion is already available which is not usually the case for companies managing pipeline infrastructures. Therefore, it is crucial to overcome such a limitation by designing corrosion prediction models able to operate even in the scenarios where supervised information is not available for a given target pipeline. As mentioned in Section 1.1, this constitute the ultimate objective of this Part of the dissertation which is organized as follows: in *Chapter* 3, we will start by describing all the necessary steps to integrate the aforementioned data sources (geometrical and fluid-dynamical) and validate their integration in the context of both oil and gas pipelines (see Section 3.1). The quality of the integration procedure will be assessed through a cross-correlation analysis carefully evaluated by domain experts, and, a set of feature selection experiments will be used to better understand the corrosion phenomenon across the different pipelines. Then, we will proceed developing and appropriately testing an ML-based predictive model of the corrosion leveraging the integrated data sets (see Section 3.2). Finally, in *Chapter* 4, we will leverage TL to design an algorithmic solution for corrosion prediction when supervised information is not available for the target pipeline [Canonaco et al., 2021c] which constitutes an important step toward corrosion prevention through ML-based tools and techniques. Indeed, TL allows overcoming the lack of supervised information on the target pipeline by exploiting a set of source pipelines where supervised information is available, which is a very common real-world scenario for oil and gas companies. In more detail, the proposed approach will rely on the joint use of a transductive TL technique and an Importance Weighted Cross-Validation (IWCV) technique; the former to build the predictive model onto the target pipeline and the latter to rank the sources w.r.t. their estimated performance over the pipeline of interest. Remarkably, all the experiments will be carried out on a set

of real-world pipelines.

2.2 Related Literature

Recently, there has been some work trying to leverage ML approaches in order to tackle the corrosion prediction problem. In De Masi et al. [2017] they use a mutual information based approach to select the most important features with respect to the corrosion phenomenon and then they apply a neural network to predict the corrosion level on the pipeline using the features having more than a certain value of mutual information with respect to the corrosion level. Information on the data set size are not available. Moreover, it is unclear whether they have evaluated the neural network performance in Cross-Validation (CV) or on a test set. In De Masi et al. [2014], they also use a neural network to perform predictions of the phenomenon's behavior in terms of corrosion rate from one period to another, metal loss and area of defect. They apply a feature selection procedure based on the sensitivity of the network output with respect to infinitesimal input variations. Unfortunately, the neural network is trained on a data set consisting of 150 instances obtained by a groundless selection of a subset from the starting pipeline composed of 1700 samples. It is again unclear whether they asses the performance of the network on a test set or in CV. In Liao et al. [2012], instead, they build a neural-network-based prediction model on a data set sampled from different pipelines (similar in terms of physical and operating characteristics). The data set is split in a training part and a testing part, one to build the model itself the other to asses its performances. They select the features to use in order to build the predictive model via a Grey Relational Analysis. Unfortunately, the data set used is composed of 116 samples.

Therefore, to the best of our knowledge, it appears that ML techniques are not properly leveraged in this real-world scenario because either experiments are incorrectly carried out or the data set used is incredibly small. Furthermore, none of these works take into account the fact that the supervised information, composed of corrosion measurements along the pipeline profile, is hardly available for a given infrastructure of interest due to the huge cost associated to the procedure that retrieves it. This make standard SL techniques useless in real-world corrosion prediction settings for oil and gas companies without circumventing the need for supervised information coming from the pipeline of interest.

CHAPTER 3

A Machine Learning Approach for the Prediction of Internal Corrosion in Pipeline Infrastructures

In the context of this *Chapter*, we will start by reviewing how to build a rich and descriptive data set out of the available sources of data for a pipeline of interest. The resulting data sets will be evaluated by domain experts and used to build a predictive model of the corrosion for each pipeline. These steps are necessary to reach the final goal of this part of the dissertation consisting in designing an ML-based model able to soften the assumption of available supervised information coming from the target pipeline infrastructure.

The work herein presented is taken from Canonaco et al. [2020b] and Canonaco et al. [2021b], where I am the main researcher.

3.1 Data Set Creation

As mentioned in Section 2.1, a proper integration of the different sources of data (geometrical and fluid-dynamical) is crucial to describe the corrosion phenomenon. Therefore, in this section, we will go through all the steps needed to accomplish this goal, uncovering all the pitfalls hidden inside the integration procedure and describing how to avoid them.

As already mentioned, we have two sources of data (depicted in Figure 3.1), the geometry of the pipeline and the fluid-dynamical simulations, containing information that can

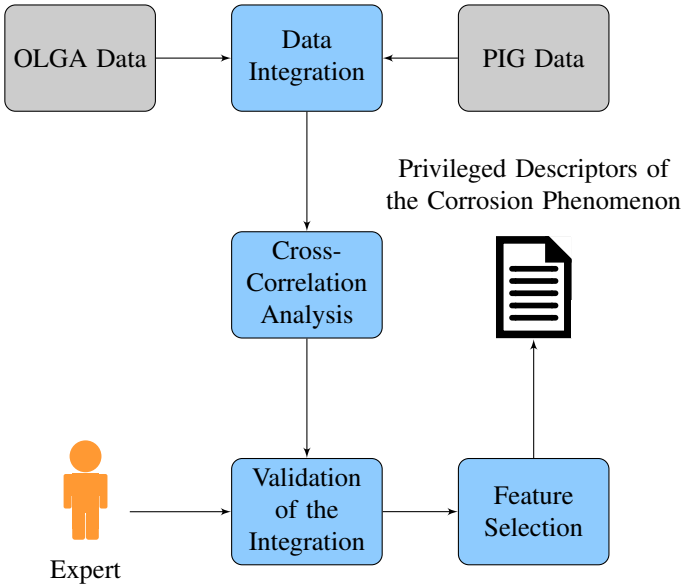


Figure 3.1: *Comprehensive scheme of the approach.*

help to model the corrosion phenomenon. The geometrical data come from pigging operations, which means that a PIG is sent through the pipeline to harvest data about the geometry (the PIG also collects data about the corrosion itself). The fluid-dynamical data comes from OLGA, a dynamic multiphase flow simulator [Schlumberger]. These two sources of data cannot be directly integrated due to some limitations in the simulator, which prevent us from obtaining fluid-dynamical information for each corrosion point harvested by the PIG in the pipeline. In other words, we can perform fluid-dynamical simulations down to a certain granularity under which OLGA returns us non-stationary solutions. Therefore, we need to devise a proper strategy to associate fluid-dynamical information with corrosion and geometrical data coming from the PIG. In order to do this, we will split our pipeline into bars whose beginning and end are information yielded by the PIG. More precisely, the beginning and end of a bar coincide with its welding points joining it with the previous and next bar, respectively. Hence, given a bar identified by the PIG, we just need to summarize all the corrosion and fluid-dynamical information associated with that bar. For what concerns the corrosion information, we will assume the maximum peak of corrosion within the bar as a representative. The peak of corrosion is intended in terms of corroded thickness percentage w.r.t. the thickness of the bar itself. If we have two corrosion points with the same corroded thickness, we choose the one with a wider area. At this point, we need to choose fluid-dynamical representatives for the bar. We have three main cases to consider:

1. multiple fluid-dynamical points fall within the same bar (see Figure 3.2a);
2. no fluid-dynamical point falls within the bar, but the bar is neither at the beginning nor a the ending of the pipeline (see Figure 3.2b);
3. no fluid-dynamical point falls within the bar and the bar is the first or the last one of the pipeline (see Figure 3.2c).

In all these three cases we need to be careful to preserve the physical relationships among all the fluid-dynamical variables when choosing the representative for the bar. For instance, in the first case is not appropriate to perform a simple arithmetic average to reach our goal. We need to average w.r.t. the volume associated with each point. Table 3.1 reports all the fluid-dynamical variables used in the context of gas pipelines, whereas in Table 3.2 we can find those used in the context of oil pipelines. For what concerns the second case, the simplest solution is to interpolate the nearest next and previous fluid-dynamical points to prevent duplication. Unfortunately, in the context of oil pipelines, this straightforward solution, in some cases, could lead to unacceptable results according to the domain experts. Therefore, only for oil pipelines, if the next or previous point is closer to the current empty bar than a certain threshold, the duplication is applied (this threshold is set to 0.25 m). Unfortunately, in the third case, we cannot do anything but duplicate the nearest fluid-dynamical point both for oil and gas pipelines.

(a) *Multiple simulated fluid-dynamical points within the same bar.*

(b) *No simulated fluid-dynamical point within a given pipeline bar.*

(c) *No simulated fluid-dynamical point within a given bar, but the bar is either at the beginning or at the ending of the pipeline.*

Figure 3.2: *Critical configurations of the simulated fluid-dynamical points within the bars of the pipeline.*

3.1.1 Datasets Description

In our scenario, we applied the proposed integration strategy on two gas pipelines and two oil pipelines, namely: GP1, GP2, OP1, and OP2. For each of these pipelines, we gathered information from both OLGA and the PIG. Table 3.3 reports all the geometrical variables, which are the same for oil and gas pipelines, whereas Table 3.4 details summary information about the pipelines. Observe that both oil and gas pipelines show a morphological taxonomy highlighted by the type in Table 3.4, representing two different shapes of the pipeline.

Table 3.1: *Fluid-dynamical variables in the context of gas pipelines.*

OLGA Variable	Meaning	Unit of Measurement
ID_{flow}	flow regime	-
PT	pressure	$bara$
GT	total mass flow rate	kg/s
QT	total volumetric flow rate	m^3/s
GG	gas mass flow rate	kg/s
QG	gas volumetric flow rate	m^3/s
UG	gas velocity	m/s
$TAUWG$	gas wall shear stress	Pa
$GLWVT$	total water mass flow rate including vapor	kg/s
$QLTWT$	water volumetric flow rate	m^3/s
$UWTCONT$	water continuous velocity	m/s
$TAUWWT$	water film wall shear stress	Pa
$HOLWT$	water hold-up	-
$INCL$	horizontal inclination	$degree$
TM	temperature	$Celsius$
A	section area	m^2

Table 3.2: *Fluid-dynamical variables in the context of oil pipelines.*

OLGA Variable	Meaning	Unit of Measurement
ID_{flow}	flow regime	-
PT	pressure	$bara$
GT	total mass flow rate	kg/s
QT	total volumetric flow rate	m^3/s
GG	gas mass flow rate	kg/s
QG	gas volumetric flow rate	m^3/s
UG	gas velocity	m/s
$TAUWG$	gas wall shear stress	Pa
$GLWVT$	total water mass flow rate including vapor	kg/s
$QLTWT$	water volumetric flow rate	m^3/s
$UWTCONT$	water continuous velocity	m/s
$TAUWWT$	water film wall shear stress	Pa
$HOLWT$	water hold-up	-
$INCL$	horizontal inclination	$degree$
TM	temperature	$Celsius$
A	section area	m^2
$GLTHL$	total oil mass flow rate	kg/s
$QLTHL$	oil volumetric flow rate	m^3/s
$UHLCONT$	oil continuous velocity	m/s
$TAUWHL$	oil film wall shear stress	Pa
$HOLHL$	oil hold-up	-

Table 3.3: *Geometrical variables in the context of oil and gas pipelines.*

Geometrical Variable	Meaning
$Odometry$	Odometry of a given bar
Bar_{init}	Odometric point where the bar begins
Bar_{end}	Odometric point where the bar ends
Lat	Latitude of the bar
$Long$	Longitude of the bar
$Elev$	Elevation of the bar
$Bar\ length$	Length of the bar

Table 3.4: *Label distribution across the different pipelines. In the column Low, Medium, and High, we show the respective number of low, medium and high corroded points (i.e., with a peak depth percentage p such that $0.03 \leq p < 0.08$ for what concern Low, $0.08 \leq p < 0.3$ for what concern Medium and $p \geq 0.3$ for what concern High).*

Pipeline	Type	Data Points	Low	Medium	High
GP1	Gas type I	1720	33	462	170
GP2	Gas type II	910	38	194	56
OP1	Oil type I	1442	152	909	256
OP2	Oil type II	1047	542	280	14

3.1.2 Validation of the Integration: Cross-correlation Analysis

After the integration of the two aforementioned sources of data, we need to assess the relationships among the variables within the resulting data set. In order to do so, we performed a cross-correlation analysis for each couple of features we have in the reconstructed data sets (i.e., the data sets for GP1, GP2, OP1, and OP2). Notice that some variables coming from the OLGA simulator were excluded from this analysis because they showed an unexpected behavior according to the domain experts, while some others were removed because redundant or constant. In both cases, features have been removed from the data sets. In particular, the removed features in the oil and gas pipelines are: $GLWVT$, QT, GT, QG, GG (removed only in gas pipelines since it is constant), Bar_{init} and Bar_{end} (both redundant). Notice that the first four variables mentioned above were removed because they showed unsatisfactory behavior according to the domain experts. Furthermore, ID_{flow} has been removed from this analysis because it is a categorical variable.

The outcome of the cross-correlation analysis is represented in a graph as follows: given a graph $G = (V, E)$, where V is the set of nodes and E is the set of arcs of the graph, every node in V represents a feature of our data set and an arch exists in E between a couple of nodes v_i and v_j if and only if the cross-correlation between the associated features is greater than or equal to an expert-defined threshold T, which in our specific

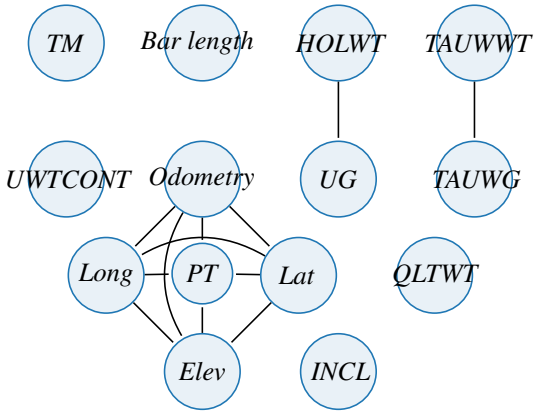


Figure 3.3: *Cross-correlation analysis on GP1.*

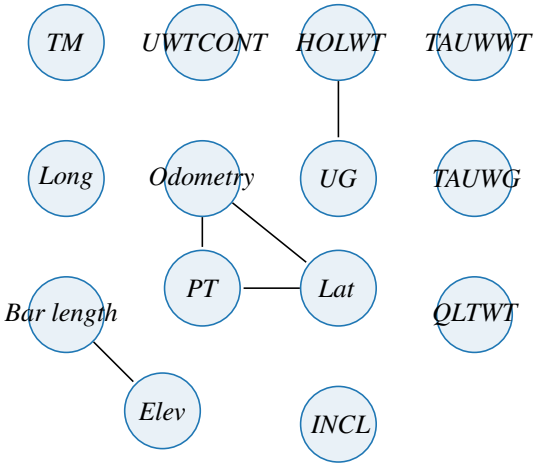


Figure 3.4: *Cross-correlation analysis on GP2.*

case is set to 0.9 (in order to highlight strong relationships). The cross-correlation between couples of features is computed in the following way [Alippi et al., 2013]:

$$R_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i} \cdot C_{j,j}}}, \tag{3.1}$$

where $C_{i,j}$ represents the covariance between features i and j, and $C_{i,i}$ represents the variance of the feature i. The overall procedure is described in Algorithm 1.
Experimental results, that are given in Figure 3.3, 3.4, 3.5 and 3.6, are notably interesting and relevant.

In particular, from the cross-correlation analysis in gas pipelines GP1 and GP2, we see two main clusters of features. More precisely, the first cluster comprises UG, $HOLWT$,

$TAUWG$, $UWTCONT$, and $TAUWWT$, which are related to each other through the velocity of the two fluids (water and gas). The second cluster comprises $Odometry$, $Latitude$, $Longitude$, $Elevation$ and PT. In this cluster, the pressure is related to the other variables because the depth varies w.r.t. the odometry. The slight differences between Figure 3.3 and 3.4 are due to different flow regimes within the pipelines together with the fact that the length of the bars increases (or decreases) w.r.t. the $Elev$ in GP2.

In the context of oil pipelines OP1 and OP2, we still have the cluster of features associated with the geometry and pressure together with the cluster of variables associated with the velocities of the fluids composing the blend. Moreover, in the case of oil pipelines, an additional cluster comprising GG (Gas mass flow rate), $GLTHL$ (Total oil mass flow rate), TM (Temperature), $QLTHL$ (Total oil volume flow rate) exists. Interestingly, this cluster is correlated with LAT, $LONG$, and $ODOMETRY$. The reason is now commented. In a pipeline, the temperature of the input fluid of the pipeline (comprising the different phases in thermal equilibrium) could be higher than the one of the seabed (e.g., pipelines operating on the seabed as OP1 and OP2). For this reason, the fluid cools down over the pipeline. Hence, TM results to be correlated to the $ODOMETRY$. Similarly, the $GLTHL$ (Total oil mass flow rate) and GG (Gas mass flow rate) result to be correlated with LAT, $LONG$ and $ODOMETRY$ since the oil can evaporate or condense over the pipeline due to pressure variations. This implies that the gas/oil equilibrium changes over the pipeline and, for this reason, a correlation among $GLTHL$ (Total oil mass flow rate), GG (Gas mass flow rate), $ODOMETRY$ and PT could exists (see Figure 3.5 and 3.6).

Thanks to the interaction with domain experts, we were able to validate the proposed integration phase through the results reported above, which highlights the preservation of the involved fluid-dynamical physics.

Algorithm 1 Cross-correlation graph

Input: cross-correlation matrix R, number of features N, user defined threshold T
$E = R$
for i in $1 \cdots N$ **do**
  for j in $1 \cdots N$ **do**
    if $E(i, j) \geq T$ **then**
      $E(i, j) = 1$
    else
      $E(i, j) = 0$
    end if
  end for
end for
return E

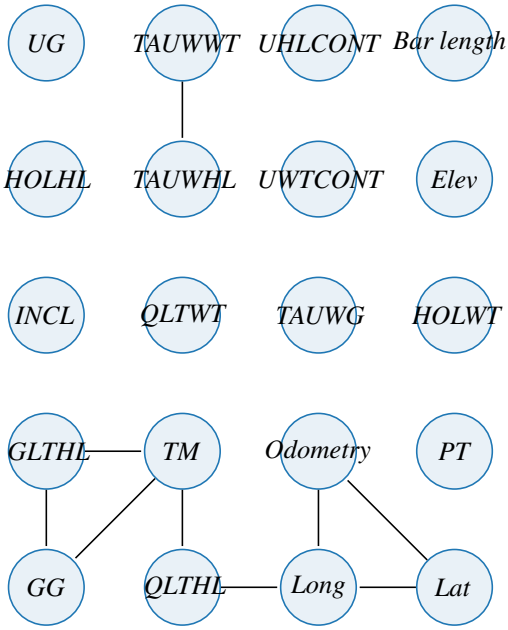


Figure 3.5: *Cross-correlation analysis on OP1.*

Table 3.5: *Feature Selection on gas pipelines.*

	GP1	GP2
Bar length		✓
HOLWT		✓
PT		✓
Odometry	✓	✓
x	✓	✓
y	✓	✓
z	✓	
x'	✓	

3.1.3 Gaining Insights about Corrosion: Feature Selection

In this section, we describe the analysis performed to identify the features that could represent the corrosion phenomenon in a privileged way. In order to do this, we used the feature selection algorithm proposed in Alippi et al. [2001]. This algorithm comprises two phases described in Algorithm 2 and 3, respectively. For what concerns the first phase, the algorithm starts by analyzing all the subsets of features whose cardinality is 1. For each subset, we compute the performance in terms of k-fold CV accuracy (while leave one out was used in Alippi et al. [2001]), choosing only the best W subsets. Now, this

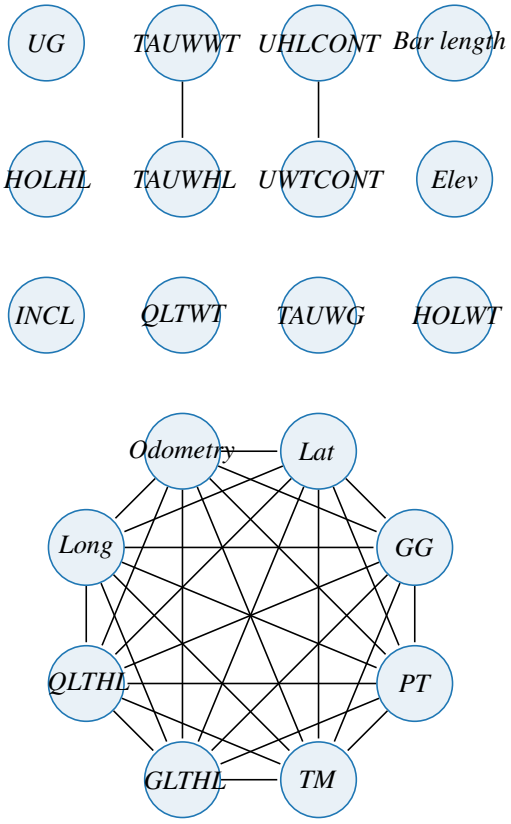


Figure 3.6: *Cross-correlation analysis on OP2.*

procedure is repeated on all the subsets of dimension two over the set given by the union of the best W subsets previously spotted. The first phase iterates until convergence, which happens when the cardinality of the subsets is equivalent to the cardinality of the union set. The second phase of the algorithm consists in adding one left-out feature at a time to the set yielded by the first phase: if the feature increases the performance, then it is added to the solution, otherwise, it is discarded. Once the second algorithm iterates over all the left-out features, if no left-out feature has been added to the current set of features then the algorithm ends, otherwise it iterates again over the remaining left-out features. For a comprehensive description of this algorithm see Alippi et al. [2001].

The results of the execution of this algorithm (Phase 1 and 2) on the gas and oil data sets are shown in Tables 3.5 and 3.6, respectively. Observe that, Lat, $Long$ and $Elev$ where replaced by their equivalent representation in Cartesian coordinates (x, y and z) so as to allow an easier computation of the first- and second-order derivatives (represented as x', y', z' and x'', y'', z'' respectively). These derivatives show the rate of change of the pipeline profile along a fixed component (x, y, or z). As we can see from Table 3.5, it seems that

Table 3.6: *Feature Selection on oil pipelines.*

	OP1	OP2
ID flow	✓	
HOLHL	✓	
QLTHL	✓	
Odometry	✓	
QLTWT	✓	
x	✓	✓
TAUWHL		✓
TM		✓

some features are associated with the corrosion modeling when we have a type I pipeline, whereas some others are associated with a more general description of the corrosion. The same comment arises in the context of oil pipelines as shown in Table 3.6.

Algorithm 2 Feature Selection Phase 1

Input: user defined threshold W, folds to use k, evaluation model M, alive features A initially set to all the feature of the dataset, n=1
while $|A| > n$ **do**
  compute all the subsets of A whose cardinality is n
  compute the performance in k-fold CV of each subset using model M
  select the best T subsets and put their union into A
  $n = n + 1$
end while
return the subset associated to the best performance

Algorithm 3 Feature Selection Phase 2

Input: solution given by phase 1 S, folds to use k, evaluation model M, left-out features L, added feature $AF = True$
while AD **do**
  $AD = FALSE$
  for f in L **do**
    compute the performance in k-fold CV of $S \cup f$ using model M
    if performance improve **then**
      $S = S \cup f$, $AD = TRUE$, remove f from L
    end if
  end for
end while
return the subset associated to the best performance

3.1.4 Discussion

Building a data set aiming at representing a given phenomenon is a tricky task, especially when the available information is scarce and we cannot directly measure all the variables influencing the corrosion phenomenon we want to capture. However, even though we do not have the direct measurements of the fluid-dynamical variables influencing the given phenomenon sometimes, a simulator for those variables is available. Hence, our problem required understanding how the data integration phase should be guided by the physics of the phenomenon in order to guarantee a reliable physical/fluid-dynamical representation. This reasoning implied discussions with domain experts in order to devise a strategy to check the soundness of the obtained representation, which, in this context, required to apply a cross-correlation analysis and evaluate the results. Once the data sets have been correctly integrated, we started to analyze their properties in order to find out more about the corrosion phenomenon, which was accomplished through a feature selection procedure. Notice that every step of this investigation was constantly supervised by domain experts validating the definition of the data set and the performed analysis.

3.1.5 Conclusion

In this Section, we tackled the problem of integrating the two different sources of data available for corrosion prediction. We uncovered some of the pitfalls hidden in the integration procedure showing how to avoid them. We have, also, evaluated the reconstructed data sets through a cross-correlation analysis, which helped us checking the physical relationships among the variables at hand. Finally, we applied a feature selection algorithm in order to identify privileged descriptors of the corrosion phenomenon across different pipelines.

3.2 Corrosion Prediction

In the previous Section, we have devised a strategy to properly integrate fluid dynamics and geometry for any given pipeline. Now, given an integrated data set, we would like to focus on the corrosion prediction problem leveraging SL techniques.

3.2.1 Problem Formulation

From an ML perspective, in order to tackle the corrosion prediction problem, we need three steps to follow. At first, we need to build a data set representing the phenomenon to be modeled. Secondly, we need to apply a learning algorithm to this data set in order to get a predictive model of the phenomenon. Finally, we need to assess the predictive performance of the learned model. In this Section, we will describe these three steps one after the other (for a summary see Figure 3.7).

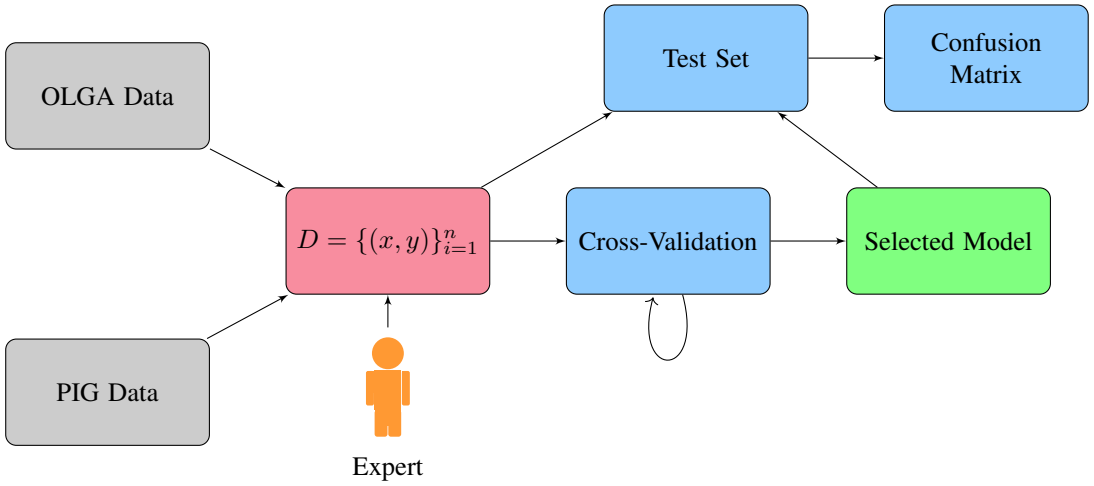


Figure 3.7: *Comprehensive scheme of the approach (in gray the sources of data which together with the red block are associated to the data set creation, in green the selected model and in violet the evaluation procedure).*

The Data Set

As mentioned in Section 2.1 and more thoroughly in Section 3.1, we have an incompatibility between the two available sources of data, which are the geometry of the pipeline and the fluid dynamics. This incompatibility is due to the fact that one source of data is simulated the other is sampled directly from the infrastructure. This integration problem was tackled in Section 3.1, where we proposed to split the pipeline into bars and summarize all the corrosion and fluid dynamical information related to each bar in such a way as to preserve the physical relationships among all the descriptors of the corrosion phenomenon. Now, leveraging this integration procedure we can build a rich and descriptive data set faithfully representing the corrosion phenomenon in a fixed given pipeline. Once the integration has been performed on the two sources of data, we have a set of couples $D = \{(x_i, y_i)\}_{i=1}^{n}$ where x_i represent the vector whose components are the descriptors of the corrosion phenomenon and y_i is the scalar representing the corrosion level (intended as a percentage of the pipe thickness being corroded). It is noteworthy to point out that x_i is made of geometrical and fluid-dynamical components which are described in Table 3.3 and 3.1 respectively, whereas y_i is transformed into a categorical variable with four possible classes which are derived from the corrosion level according to the thresholds given by the domain experts that are reported in Table 3.7. This latter categorization of the corrosion level is done since companies managing pipeline infrastructures are mainly interested in being able to classify the corrosion level in a certain number of categories directly related to the rupture risk in a given point of the pipe.

Table 3.7: *Categories and Thresholds.*

Threshold	Category
$y < 0.03$	Absent
$0.03 \leq y < 0.08$	Low
$0.08 \leq y < 0.30$	Medium
$y \geq 0.30$	High

Classification of Internal Corrosion

The corrosion classification problem is formalized as follows:

$$\theta^* \in \arg\min_{\theta} \mathbb{E}_{(x,y)\sim P} \left[l(x, y, f(x, \theta)) \right], \tag{3.2}$$

where $x \in \mathbb{R}^m$ is the m-dimensional vector representing the input given to the classification model, $y \in 1, \ldots, C$ represents the output to be predicted, P is the joint distribution from which all the couples (x, y) are sampled, $\theta \in \mathbb{R}^d$ is the d-dimensional vector parametrizing the classification model $f(x, \theta)$ and l is a suitable loss function to be optimized by the learning algorithm in order to find the optimal classification model. Of course, the distribution P is not available in practice, but we have a finite data set, $D = \{(x_i, y_i)\}_{i=1}^n$, sampled from it. Therefore, we resort to find:

$$\hat{\theta} \in \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} l(x_i, y_i, f(x_i, \theta)), \tag{3.3}$$

minimizing the empirical risk.

In our scenario, we employed three different types of classification algorithms to solve the above formalized problem: Support Vector Machines (SVM) [Scholkopf and Smola, 2018], Neural Networks [Bishop, 2006] and XGBoost [Chen and Guestrin, 2016]. The first one is a kernel-based tool that involves solving a convex optimization problem to determine the optimal parameters of the predictive model during the training phase. For what concerns SVM the main hyperparameters are: the kernel (radial basis functions were used in this context), the kernel coefficient γ, and the regularization parameter C. The second one consists of a series of layers (stacked one after the other) made of neurons with a non-linear activation function. This model is trained with gradient-descent-based optimization algorithms and its hyperparameters are: the number of neurons in each layer and the number of hidden layers (in this work, we used a single hidden layer). The third one is an efficient tree-boosting algorithm, hence it falls into the ensemble approaches. XGBoost has lots of different hyperparameters. To name a few of them, we have: *max depth*, *column sample by tree* and *number of estimators*.

Performance Evaluation

In order to properly assess the performance of the learning algorithms, we need to split our data set in two. One part representing the training set and the other the test set. The first part will be used to train the predictive model, whereas the second one to assess its performance. We split the data set into an 80% training set part and a 20% test set part. The hyperparameters maximizing the performance of a learning algorithm are chosen via CV on the training set. Then the best predictive model obtained for each learning algorithm is allowed to predict on the test set and its performance is reported as a confusion matrix (for completeness, we report also the performances measured in accuracy and *F1-score*). A confusion matrix is a $C \times C$ matrix where the rows represent true labels and the column the predicted ones. Therefore, on the diagonal, we have the per-class accuracy of the algorithm, whereas, in the other cells, we can see how the algorithm confuses one class with another.

3.2.2 Experiments

Data Sets Description

In the context of this work, we will apply the classification algorithms on three pipelines, namely: P1, P2 and P3. For each of these three pipelines, we have data coming from the PIG and OLGA. These two sources are integrated as mentioned in Section 3.1. Subsequently, some features have been removed from the data sets because, according to the domain experts, they showed unexpected behavior, whereas some others were removed due to redundancy. More specifically, $GLWVT$, QT, GT, QG, $Bar\ length$ and GG removed under domain expert advice, Bar_{init} because redundant. The GPS coordinates (Lat, $Long$ and $Elev$) are transformed into their equivalent Cartesian coordinates (x, y and z), then their first and second-order derivatives are computed (and added to the feature set), which represent the rate of change of the pipeline profile along a fixed component. Moreover, under the suggestion of the domain experts, the samples associated with the beginning or the ending of the pipeline were removed because they have a completely different behavior in terms of corrosion. Concluded this enrichment step, the features are normalized. Finally, in Table 3.8, we can see some summary information on the three pipelines. Notice that the pipelines have a morphological taxonomy represented by Type in the previously mentioned table.

3.2.3 Results

A grid search in CV over the training set (corresponding to the 80% of the whole data set) is performed to select the best hyperparameters for each of the classification algorithms mentioned in Section 3.2.1. *F1-score* is the figure of merit taken into account for the grid search. Performances of the optimal parameter configuration on each algorithm

Table 3.8: *Label distribution across the different pipelines.*

Pipeline	Type	Data Points	Low	Medium	High
P1	Type I	1702	32	459	170
P2	Type II	861	36	186	53
P3	Type I	3321	144	609	19

Table 3.9: *F1-score associated to the optimal hyperparameter configuration for each algorithm across the different pipelines.*

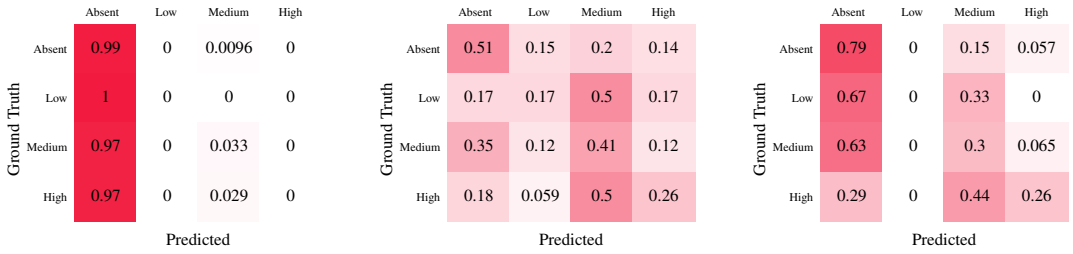
Pipeline	XGB		SVM		NN	
	mean	std	mean	std	mean	std
P1	**0.62**	0.03	0.51	0.04	0.59	0.02
P2	**0.77**	0.04	0.72	0.06	0.74	0.05
P3	**0.74**	0.02	0.58	0.02	0.73	0.02

are reported in Table 3.9. After the best parametrization is obtained, the model is run onto the test set to assess its performance. Therefore, for each available pipeline, we report the associated confusion matrix representing the performance obtained by the three classification algorithms on the test set.

As we can see from Figures 3.8 and 3.10 XGBoost seem to predict nearly always absent, whereas in Figure 3.9 it misses on the low component of the phenomenon. In the case of neural networks, instead, we have a similar behavior in the context of pipeline P3 (even though, to be precise, this model seems to better capture the medium component of the phenomenon w.r.t. XGBoost), but not for what concern P1 and P2. Indeed in P1 neural networks miss on the low component of the phenomenon, whereas in P2 all the components are captured even though some of them partially. Both neural networks and XGBoost are quite polarized toward the absent class, which is not the case of SVM, indeed, it is able to capture, even though not perfectly, also the other corrosion components of the phenomenon. Finally, if we take into account the accuracy, then the best model seems to be the one produced by XGBoost (see Table 3.10). In terms of *F1-score*, instead, the neural network seems to have a slightly better performance w.r.t. the others (see Table 3.11). However, this can be misleading if we do not take into account the confusion matrices which shed some light on the per-class behavior of the employed models.

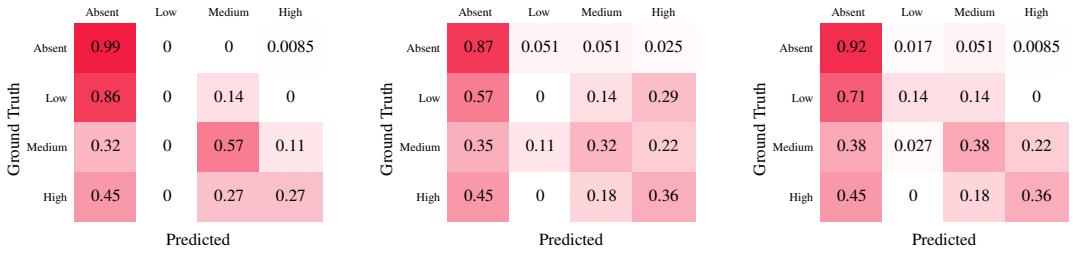
Top vs Bottom

The corrosion phenomenon within a pipeline can be split into two different components, one related to the top portion of a given pipe the other to the bottom. In the context of the analyzed pipelines, we have approximately 3%, 1.7%, and 8.6% of top corrosion in P1, P2, and P3 respectively. This makes the corrosion phenomenon almost completely dominated



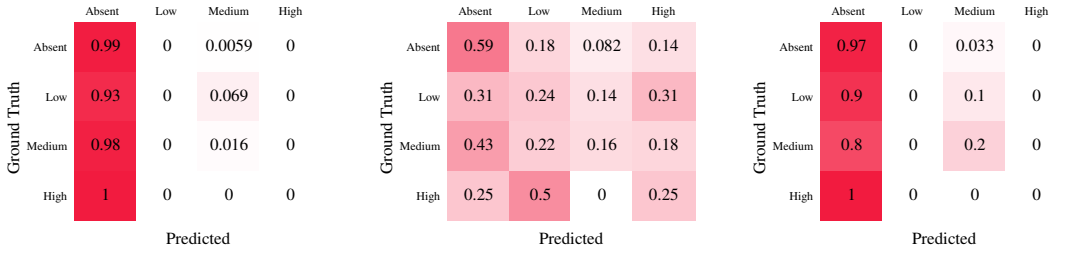
(a) *XGB* **(b)** *SVM* **(c)** *NN*

Figure 3.8: *Multi-class confusion matrices for the three classification algorithms on P1.*



(a) *XGB* **(b)** *SVM* **(c)** *NN*

Figure 3.9: *Multi-class confusion matrices for the three classification algorithms on P2.*



(a) *XGB* **(b)** *SVM* **(c)** *NN*

Figure 3.10: *Multi-class confusion matrices for the three classification algorithms on P3.*

Table 3.10: *Testing performances in accuracy across the different pipelines.*

Pipeline	XGB	SVM	NN
P1	**0.62**	0.45	0.60
P2	**0.82**	0.69	0.74
P3	0.77	0.50	**0.78**

Table 3.11: *Testing performances in F1-score across the different pipelines.*

Pipeline	XGB	SVM	NN
P1	0.48	0.50	**0.57**
P2	**0.78**	0.68	0.72
P3	0.67	0.57	**0.72**

by the behavior at the bottom of the pipeline. Therefore, distinguishing between the two phenomena is nearly irrelevant from the domain expert point of view, at least in these three pipelines.

3.2.4 Discussion

Modeling the corrosion phenomenon using an ML-based approach is a tricky task, which starts with the need to create a data set faithfully representing the phenomenon to be modeled, continues with the choice of the learning algorithm to approximate the phenomenon, and ends with the evaluation of the performance obtained with the chosen learning algorithm.

As highlighted in the previously described results, deciding how to evaluate the performance of a model is a crucial step in the ML framework. Indeed, switching from one figure of merit to another could affect the identification of the best ML model as we have seen comparing accuracy to *F1-score* in Tables 3.10 and 3.11 respectively. Moreover, the experimental results described above show that, through the performance evaluation via confusion matrices, there are models able to capture only partially the corrosion phenomenon, while other models seem to provide a more complete characterization of the phenomenon under inspection (see for instance Figure 3.10 where XGBoost and the neural network polarize toward the absent class, whereas the SVM better catches all the phenomenon's components).

3.2.5 Conclusion

The aim of this *Chapter* was to introduce an ML-based approach for the prediction of corrosion in pipeline infrastructures leveraging a data set built with the aim to compensate for the missing information. Differently from the ML solutions for corrosion prediction

present in the literature (where small data sets have been considered or the performance evaluations have not been properly carried out, hence impairing the claims and the obtained results), the proposed approach formalizes how to create the data set (comprising information coming from the PIG and the simulator), how to effectively train an ML-based model (i.e., SVM, Neural Networks and XGBoost) and how to properly evaluate its performance.

CHAPTER 4

Corrosion Prediction in Pipeline Infrastructures Leveraging Transfer Learning

In this *Chapter*, which is taken from Canonaco et al. [2021c], where I am the main researcher, we will finally turn our attention to the problem of building an ML-based predictive model for the corrosion when there is absence of supervised information for a given target pipeline. This absence is due to the high cost associated with the retrieval of such information and constitute a common use case for oil and gas companies managing such infrastructures.

4.1 From supervised to transfer learning in corrosion prediction

From a machine learning perspective, predicting the presence of corrosion in an oil or gas pipeline requires effective modeling of the relationships between the corrosion-influencing factors and the presence of corrosion in a pipeline. More specifically, as we have seen in *Chapter* 3, a pipeline is divided into bars, i.e., independent segments of the pipeline, and the machine learning task aims at modeling the relationship between the corrosion-description factors of each bar and its level of corrosion.

Let us define a learning task as the tuple $(X, P(x), Y, P(y|x))$, where Y is the label set, $P(y|x)$ is the conditional distribution, X is the feature space and $P(x)$ is the marginal

distribution. Here x represents the feature vector associated with a bar and y its corrosion level. In the context of a learning task, the objective is to learn a function correctly predicting y when evaluated on x.

In this setting, two main problems arise. First, as commented in Sections 2.1 and 3.1, not all the corrosion influencing factors are available. Hence the feature vector x comprises only geometrical and fluid dynamical information about a bar: the former is available given the pipeline on-the-field deployment, while the latter is provided by a fluid-dynamical simulator [Schlumberger]. Second, as highlighted in Section 2.1, the supervised information, i.e., the presence or absence of corrosion in a bar, is provided through an inspection campaign where a PIG passes inside the pipeline collecting a set of corrosion level measurements together with their GPS coordinates along the pipeline profile. This poses a relevant compatibility issue since the geometrical characteristics, mainly represented by the inclination and curvature of the pipeline profile, are not directly compatible with the fluid dynamical descriptors creating a resolution mismatch between the information gathered through the PIG and the simulator [Schlumberger]. More precisely, we can obtain fluid-dynamical data down to a certain granularity under which the simulator starts returning non-stationary solutions. This issue has been highlighted and addressed in Section 3.1 and we will follow this approach in the context of this *Chapter* to build a given data set $D = \{(x_i, y_i)\}_{i=1}^{n}$ with x representing the feature vector, made of fluid dynamical and geometrical components, and y representing the corrosion level.

From the ML perspective, SL techniques, like the ones mentioned in Section 2.2, work under the assumption that the distribution of training and test data is the same. In real-world oil and gas pipeline infrastructures, it may happen that we would like to solve a certain target task $(X_T, P_T(x), Y_T, P_T(y|x))$ for which some of the elements allowing a proper application of SL techniques are missing (e.g., we do not have access to the labels since they are too costly to be retrieved), but we can access another related source task $(X_S, P_S(x), Y_S, P_S(y|x))$ that may help us overcome the difficulties we would have in learning the target task by itself. This implies that we cannot directly build a predictive model onto a given pipeline $(X_S, P_S(x), Y_S, P_S(y|x))$ for which the supervised information is available and then reuse it onto another pipeline $(X_T, P_T(x), Y_T, P_T(y|x))$ for which the supervised information is missing. We emphasize that there is a potential distributional shift to take into account and our predictive models may suffer a performance hindering if this is not properly considered.

In this setting, TL allows us to leverage the source task in order to better approximate the joint distribution on the target task. In the TL literature, several approaches are available differing in how source and target tasks relate to each other (see Pan and Yang [2009] for an extensive treatment). For instance, we can talk about *inductive* TL whenever $(Y_S, P_S(y|x)) \neq (Y_T, P_T(y|x))$ (notice that $(X_S, P_S(x)) = (X_T, P_T(x))$ or $(X_S, P_S(x)) \neq (X_T, P_T(x))$), whereas we have *transductive* TL whenever $(Y_S, P_S(y|x)) = (Y_T, P_T(y|x))$ and $(X_S, P_S(x)) \neq (X_T, P_T(x))$. The former requires access to at least

a set of labeled data from the target context, whereas the latter only requires a set of unlabeled data coming from it. Furthermore, we can classify TL approaches w.r.t. the knowledge being transferred across the two tasks: parameters [Evgeniou and Pontil, 2004, Duan et al., 2012], features [Argyriou et al., 2006, Raina et al., 2007, Pan et al., 2008, Long et al., 2013], samples [Dai et al., 2007, Wu and Dietterich, 2004, Huang et al., 2006] or relational knowledge [Mihalkova et al., 2007, Li et al., 2012]. Finally, TL techniques may be either homogeneous or heterogeneous as they are classified in Weiss et al. [2016], Zhuang et al. [2020], where, in Zhuang et al. [2020], we may find a review of recent works with the main focus on homogeneous approaches.

In the context of this *Chapter*, we will focus on *transductive* TL techniques because companies managing pipelines usually have supervised information only for some of their pipeline infrastructures. In particular, this approach is justified by the assumption that, as commented by the domain experts, the corrosion phenomenon is the same in two different pipelines provided that the influencing factors are equal, which translates into $P_S(y|x) = P_T(y|x)$ (trivially, the label set Y does not change w.r.t. the considered pipeline). This implies that, in the context of corrosion prediction for pipeline infrastructures, we can assume that $(Y_S, P_S(y|x)) = (Y_T, P_T(y|x))$ and $(X_S, P_S(x)) \neq (X_T, P_T(x))$, where the change is given by $P_S(x) \neq P_T(x)$ since the feature space is the same.

4.2 The proposed transfer-learning approach for corrosion prediction

An overview of the proposed approach based on TL for corrosion prediction in pipeline infrastructures is given in Figure 4.1. More specifically, we have a set of source pipelines for which the supervised information is available and a target pipeline where only the influencing factors of the corrosion are accessible. Our goal is to create a predictive model for the target pipeline leveraging the source pipelines thanks to TL. This last step is done in two phases: at first, we rank the models built on a given source w.r.t. an estimate of their performance on the target (Algorithm 4 in Figure 4.1); then the best B sources (being B a tunable parameter of the algorithm) are selected to build a predictive model for the target in a multi-task manner (Algorithm 5 in Figure 4.1). Finally the predictive model $\bar{\theta}$ is used onto the target pipeline to get the corrosion levels.

In the rest of this Section, we will initially discuss how to build a model for the target pipeline when only one source is available, then we will review how to estimate the performance of a model onto the target pipeline, and finally, we will show how to leverage more than one source to build the target model in a multi-task manner.

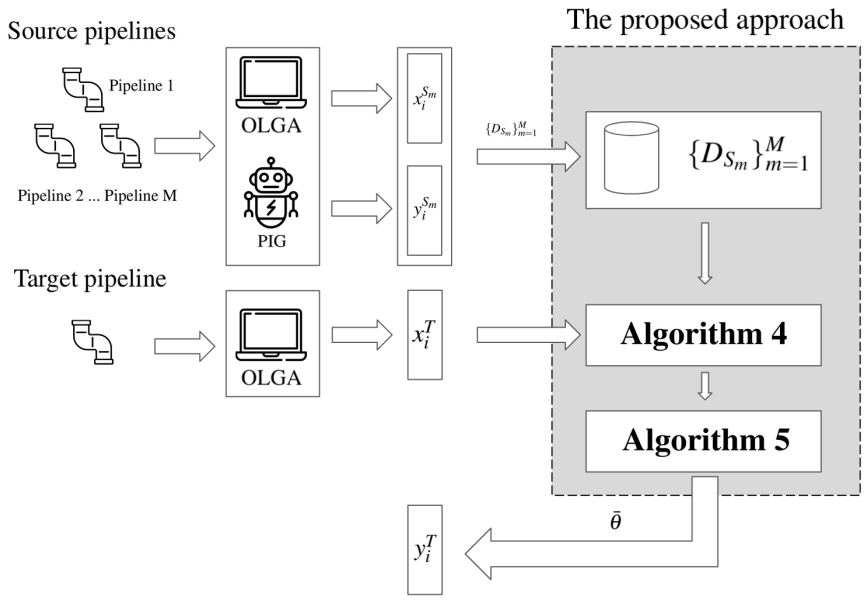


Figure 4.1: *Comprehensive scheme of the approach (this image has been designed using resources from Flaticon.com).*

4.2.1 Transfer Learning from a source to a target pipeline

Given $D_S = \{(x_i^s, y_i^s)\}_{i=1}^{n}$ and $D_T = \{x_i^T\}_{i=1}^{n'}$ representing the source and target pipelines data, the problem we want to tackle is:

$$\theta^* \in \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim P_T(\cdot,\cdot)} \left[l(x, y, f(x, \theta)) \right], \tag{4.1}$$

where θ is the vector parametrazing our predictor $f(x, \theta)$ and l is a fixed loss function measuring the mismatch between predictions and ground truth. We emphasize that samples coming from $P_T(x, y)$ are not available, but we have samples coming from $P_T(x)$ and $P_S(x, y)$. Therefore, by exploiting Importance Sampling (IS) [Fishman, 2013, Hesterberg, 1988], we can rewrite the objective of Equation (4.1) in the following way:

$$\mathbb{E}_{(x,y)\sim P_S(\cdot,\cdot)} \left[\frac{P_T(x, y)}{P_S(x, y)} l(x, y, f(x, \theta)) \right]. \tag{4.2}$$

Now, since we have finite samples, we reformulate our problem by optimizing the empirical version of the above objective function:

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \frac{P_T(x_i^S, y_i^S)}{P_S(x_i^S, y_i^S)} l(x_i^S, y_i^S, f(x_i^S, \theta)), \tag{4.3}$$

which gives us a consistent estimator of $\hat{\theta}$ [Shimodaira, 2000].

The problem of optimizing Equation (4.3) can be tackled only if we can estimate the ratio $\frac{P_T(x,y)}{P_S(x,y)}$. Since we do not have labels coming from the target pipeline, the aforementioned estimation, in a general case, cannot be achieved. In our scenario, we know that, by assumption of the domain experts, $P_S(y|x) = P_T(y|x)$. Hence, by definition of conditional probability, $\frac{P_T(x,y)}{P_S(x,y)} = \frac{P_T(x)}{P_S(x)}$, allowing an estimation of Equation (4.3) on the available data.

In order to estimate the ratio $\omega(x) = \frac{P_T(x)}{P_S(x)}$, we will resort to the approach proposed in Huang et al. [2006], which consists in solving the following Kernel Mean Matching (KMM) optimization problem:

$$\begin{aligned} &\min_{\omega} ||\mu(P_T) - \mathbb{E}_{x\sim P_S(\cdot)}\left[\omega(x)\Phi(x)\right]||, \\ &subject\ to: \\ &\quad \omega(x) \geq 0\ and\ \mathbb{E}_{x\sim P_S(\cdot)}[\omega(x)] = 1, \end{aligned} \tag{4.4}$$

where $\mu(P_T) = \mathbb{E}_{x\sim P_T(\cdot)}\left[\Phi(x)\right]$ and $\Phi : X \to \mathcal{F}$ (being $\mathcal{F}$ a feature space). Then, under the assumptions that P_T is absolutely continuous w.r.t. P_S and $\mathcal{F}$ is a Reproducing Kernel Hilbert Space (RKHS) with universal kernel $k(x,u) = \langle\Phi(x), \Phi(u)\rangle$, we have that a solution to Equation (4.4) is such that $P_T(x) = \omega(x)P_S(x)$. Unfortunately, both $\mu(P_T)$ and $P_S(x)$ are a priori unknown, but we have samples $\{x_i^S\}_{i=1}^n$ and $\{x_i^T\}_{i=1}^{n'}$ to rely on. Considering the empirical version of the objective function in Equation (4.4), under the assumptions that $\omega : X \to [0, W]$ is a fixed function with finite mean and non-zero variance given $x_i^S \sim P_S$ (W is an upper bound on how much the two distributions can be different on a given $x \in X$), $\{x_i^T\}_{i=1}^{n'}$ is an *iid* (independent and identically distributed) set of samples drawn from $P_T(x) = \omega(x)P_S(x)$ and $||\Phi(x)|| \leq R$ for any $x \in X$ then with probability at least $1-\delta$:

$$\begin{aligned} \left|\left|\frac{1}{n}\sum_{i=1}^{n}\omega(x_i^S)\Phi(x_i^S) - \frac{1}{n'}\sum_{i=1}^{n'}\Phi(x_i^T)\right|\right| \leq \\ \left(1 + \sqrt{-\log\left(\frac{\delta}{2}\right)}\right) R\sqrt{\frac{W^2}{n} + \frac{1}{n'}}, \end{aligned} \tag{4.5}$$

giving us an upper bound on the empirical optimization outcome (notice that the larger W, the larger the number of samples needed to obtain meaningful convergence guarantees). Now, letting $K_{i,j} = k(x_i^S, x_j^S)$, $\kappa_i = \frac{n}{n'}\sum_{j=1}^{n'} k(x_i^S, x_j^T)$ and $\bar{\omega}_i = \omega(x_i^S)$, the left hand side of Equation (4.5) squared can be reformulated as follows:

$$\frac{1}{n^2}\bar{\omega}^T K\bar{\omega} - \frac{2}{n^2}\kappa^T\bar{\omega} + const, \tag{4.6}$$

which yields the following optimization problem:

$$\begin{aligned} &\min_{\bar{\omega}} \frac{1}{2}\bar{\omega}^T K\bar{\omega} - \kappa^T\bar{\omega} \\ &subject\ to: \\ &\bar{\omega}_i \in [0, W]\ and\ \left|\sum_{i=1}^{n}\bar{\omega}_i - n\right| \leq n\epsilon. \end{aligned} \tag{4.7}$$

The first constraint in Equation (4.7) provides an upper bound to the degree up to which the two distributions may be different, whereas the second one forces $\omega(x)P_S(x)$ to be close to a probability measure. For a more detailed discussion about KMM please refer to Huang et al. [2006] and the references therein. Solving the optimization problem stated in Equation (4.7) will return us the weight vector $\bar{\omega}$ to correct the shift in distribution between the source and target pipelines.

4.2.2 Ranking the source pipelines

In a real-world scenarios, we may have multiple available source pipelines, $\{D_{S_m}\}_{m=1}^{M}$. Hence, we would like to identify the most appropriate source within the given set. More precisely, we would like to select the source pipeline which allows us to minimize the generalization error defined as:

$$\mathbb{E}_{\{(x_i^T, y_i^T)\}_{i=1}^{n}, u, v}\left[l(u, v, f(u, \hat{\theta}))\right], \tag{4.8}$$

where (u, v) is a test point coming from the target pipeline and not present in the training set. The generalization error is usually estimated through CV, which, in this context, is useless because we do not have access to the target pipeline labels. However, we can obtain this estimate through k-fold Importance Weighted Cross-Validation (kIWCV) [Sugiyama et al., 2007] as follows:

$$\frac{1}{k}\sum_{j=1}^{k}\frac{1}{|D_{S_m}^j|}\sum_{(x,y)\in D_{S_m}^j}\omega(x)l\left(x, y, f\left(x, \hat{\theta}_{D_{S_m}^j}\right)\right), \tag{4.9}$$

where $\hat{\theta}_{D_{S_m}^j}$ is the parametrization learned over the data set $D_{S_m} \setminus D_{S_m}^j$ ($\setminus$ is the set difference operator) and the estimation of $\omega(x)$ can be performed by solving Equation (4.7). The generalization error estimate provided by kIWCV with $k = n$ is almost unbiased and a similar claim can be proved for $k < n$ with a larger bias than that incurred with $k = n$ (for a more thorough treatment see Sugiyama et al. [2007]). Therefore, choosing the source pipeline S_m maximizing the above equation will allow us to get the best performance for our models on the target pipeline. Moreover, from a more general standpoint, Equation

(4.9) allows us to optimize all the hyperparameters of our model to maximize performance on the target pipeline.

In Algorithm 4, we combine KMM, used to estimate $\omega(x)$, and kIWCV. More specifically, at line 4, we compute the importance weights to correct the distributional shift between the current source and the target pipeline. At line 5, we compute the optimal hyper-parameters and their performance through kIWCV. At line 6, the performance and hyperparameters of the current source pipeline are appended to their respective lists, (i.e., $Perf$ and ρ). Finally, in line 8, we return the lists of performances and the corresponding hyper-parameters. This completes the TL framework we will use in the context of corrosion prediction for pipeline infrastructures.

We could extend this solution by considering more than one source pipeline for the corrosion prediction. This can be done through a multi-task learning approach which could use up to all the source pipelines together with their related importance weights to build a model for the target one [Sugiyama et al., 2012, Chapter 9]. However, evaluating the performance of all the possible subsets of source pipelines does not scale well. To avoid this issue we could combine the B best sources according to the ranking we obtain by sorting the results of Algorithm 4 (the effect of B will be experimentally evaluated in Section 4.3). This procedure is reported in Algorithm 5, where, at lines 4 and 5, we concatenate the data sets and the weights. At line 7, we compute the optimal hyperparameters of the learning algorithm, and, finally, at line 8, the best parametrization for the predictor is computed and subsequently returned.

4.3 Experiments

This Section aims to evaluate the effectiveness of the proposed TL approach in real-world scenarios of corrosion prediction within 4 different gas pipeline infrastructures, namely P1, P2, P3, and P4. The section is organized as follows: Section 4.3.1 describes the employed data sets, whereas the experimental results are given in Section 4.3.2.

Algorithm 4 Sources performance evaluation

1: **Input:** Source data sets $\{D_{S_m}\}_{m=1}^{M}$, Target samples $\{x_i^T\}_{i=1}^{n'}$, W, ϵ, k
2: $Perf = []$, $\rho = []$
3: **for** m in $1 \dots M$ **do**
4: $\quad \bar{\omega} = KMM(D_{S_m}, \{x_i^T\}_{i=1}^{n'}, W, \epsilon)$ solving Eq. (4.7)
5: $\quad hyperParams, p = IWCV(D_{S_m}, \bar{\omega}, k)$
6: $\quad Perf = Perf + [p], \rho = \rho + [hyperParams]$
7: **end for**
8: **return** ρ, $Perf$

4.3.1 Data Sets Description

In order to integrate the geometrical and fluid dynamical data, we rely on the approach proposed in Section 3.1. Such an integration will return us a data set $D = \{x_i, y_i\}_{i=1}^{n}$ for each pipeline. Here the vector x_i's components represent the corrosion influencing factors (see Tables 3.3 and 3.1), whereas y_i is the scalar value representing the corrosion level (in terms of bar thickness percentage being corroded). Such a value is transformed into a categorical variable with four possible classes according to the thresholds provided by the domain experts and reported in Table 3.7. After the integration, a data transformation and enrichment is performed as follows. The Lat, $Long$ and $Elev$ are transformed into their equivalent Cartesian coordinates x, y and z. Then their first and second-order derivatives are computed (and added to the feature set), which represent the pipeline profile's rate of change along a fixed component. Additionally, some features have been removed from the data sets because, according to the domain experts, they showed unexpected behavior, whereas some others were removed due to redundancy. More specifically, $GLWVT$, QT, GT, QG, $Bar\ length$ and GG have been removed under domain expert advice, together with Bar_{init} because redundant. Finally, under the domain experts' suggestion, the samples associated with the beginning or end of the pipeline were removed because they have completely different behavior in terms of corrosion. At the end of this enrichment step, the features are normalized. Some summary information on the different pipelines after the above-mentioned transformations is provided in Table 4.1, where we may notice a severe imbalance among the various classes. It is worth noting that the High class is very rare and the Absent class is dominating the others.

4.3.2 Results

For each available target pipeline, we will evaluate the performance of Algorithm 5 in three different configurations: $B = 1$, $B = 2$ and $B = 3$. We used a SVM [Scholkopf and Smola, 2018] with Radial Basis Kernel as $f(x, \theta)$. In Table 4.2, we report the results

Algorithm 5 Multi-task TL

1: **Input:** The B best Source data sets obtained by Algorithm 4 $\{D_{S_m}\}_{m=1}^{B}$, Target samples $\{x_i^T\}_{i=1}^{n'}$, W, ϵ, k
2: $D = [], \Omega = []$
3: **for** m in $1 \dots B$ **do**
4: $\quad \Omega = \Omega + [KMM(D_{S_m}, \{x_i^T\}_{i=1}^{n'}, W, \epsilon)]$ solving Eq. 4.7
5: $\quad D = D + [D_{S_m}]$
6: **end for**
7: $hyperParams, Perf = IWCV(D, \Omega, k)$
8: $\bar{\theta} = TrainLearningAlgorithm(D, hyperParams, \Omega)$
9: **return** $\bar{\theta}$

Table 4.1: *Label (y) distribution across the different pipelines.*

Pipeline	Data Points	Low	Medium	High
P1	861	36	186	53
P2	3321	144	609	19
P3	1215	206	213	3
P4	1681	276	568	33

Table 4.2: *Estimate of the performance (F1-Score) on the target pipeline through Algorithm 4.*

Target	1^{st} Source	2^{nd} Source	3^{rd} Source
P1	P3 0.56	P2 0.56	P4 0.45
P2	P1 0.71	P3 0.60	P4 0.45
P3	P1 0.71	P4 0.45	P2 0.30
P4	P1 0.71	P3 0.58	P2 0.56

Table 4.3: *Accuracy on Target. The Supervised Oracle column is obtained by training an SVM onto a portion of the data set representing this target pipeline and then testing this model onto the held out part.*

Target	Supervised Oracle	First Two Sources $B = 2$		All $B = 3$		1^{st} Source $B = 1$	
		No TL	TL	No TL	TL	No TL	TL
P1	0.69	0.54	**0.59**	0.41	0.41	0.33	0.57
P2	0.50	0.54	0.58	0.35	0.42	**0.71**	0.68
P3	0.52	0.26	0.43	0.44	0.45	0.40	**0.47**
P4	0.38	0.27	0.41	0.35	**0.44**	0.41	0.43

returned by Algorithm 4 when the F1-Score is chosen to perform the 10-IWCV (the model hyper-parameters are not reported for the sake of brevity). We will review each one of the aforementioned configurations of Algorithm 5 one target pipeline at a time. We will first look at the multi-class confusion matrices, then at their binarized version (reported in Appendix A) to check also the corrosion detection capabilities of each solution, and, finally, at the performances in F1-Score and accuracy.

Pipeline P1

As we can see by comparing Figures 4.2a and 4.2d, applying the TL technique gives us an improvement on the class low and the class absent recognition with a degradation effect on the class high and medium. For what concern the solution offered by using the first two sources ($B = 2$) according to IWCV (compare Figures 4.2b and 4.2e), we have an

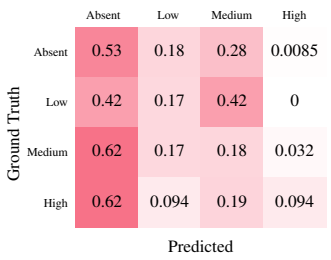
Table 4.4: *F1-Score on Target. The Supervised Oracle column is obtained by training an SVM onto a portion of the data set representing this target pipeline and then testing this model onto the held out part.*

Target	Supervised Oracle	First Two Sources $B = 2$		All $B = 3$		1^{st} Source $B = 1$	
		No TL	TL	No TL	TL	No TL	TL
P1	0.68	0.53	**0.56**	0.45	0.42	0.35	0.51
P2	0.57	0.57	0.60	0.41	0.48	**0.70**	0.68
P3	0.54	0.28	0.45	**0.48**	0.44	0.43	0.47
P4	0.41	0.30	0.38	0.35	**0.41**	0.35	0.36

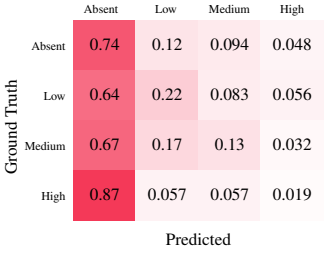
improvement on both the absent and medium-class recognition with a degradation on the class low (the class high reduction is very slight). Using just the first ranked solution ($B = 1$) w.r.t. IWCV, we obtain a model that nearly always predicts absent (see Figure 4.2f). If we take a look at the binarized version of the confusion matrices (see Figure A.1 in Appendix A), then only the solution proposed by leveraging the first two sources w.r.t. IWCV is useful in terms of corrosion detection (compare Figures A.1a, A.1b, A.1c with A.1d, A.1e, A.1f respectively). Finally, by looking at Tables 4.3 and 4.4, we may notice that applying transfer almost always increases our performance w.r.t. these two figures of merit.

Pipeline P2

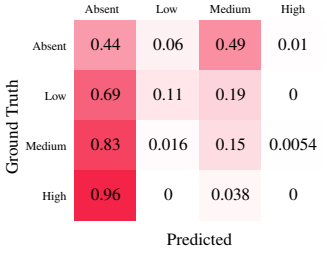
If we compare Figures 4.3a and 4.3d ($B = 3$), we may notice an improvement on the absent class recognition at the expense of a degradation for the classes low and medium. For what concern Figures 4.3b and 4.3e ($B = 2$), we can see an improvement on the recognition of the classes low and absent to which corresponds a slight reduction on the medium and high classes. By comparing Figures 4.3c and 4.3f ($B = 1$), instead, we may see an improvement in the medium class recognition accompanied by a worsening on the absent class. Now, taking a look at the binarized versions of the confusion matrices (please compare Figures A.2a, A.2b, A.2c with A.2d, A.2e, A.2f respectively in Appendix A) only the confusion matrix associated to the usage of the best source ($B = 1$) according to IWCV is meaningful. Finally, by looking at Tables 4.3 and 4.4, we see that applying transfer improves the accuracy and the F1-Score in all the cases except when we choose the best source ($B = 1$) w.r.t. IWCV. Furthermore, notice that, in some cases, the TL technique is able to get an improvement in F1-Score or accuracy w.r.t. to what was obtained by training a model onto a portion of the data set representing this target pipeline and then testing this model onto the held out part (i.e., the Supervised Oracle).



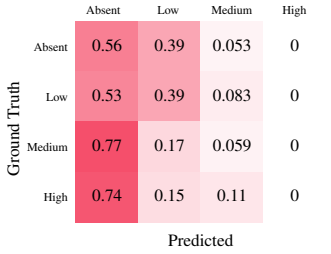
(a) *Confusion Matrix without transfer on P1 using all the sources.*



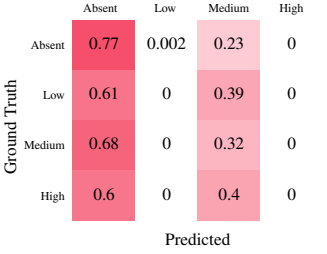
(b) *Confusion Matrix without transfer on P1 using the first two sources according to the IWCV metric.*



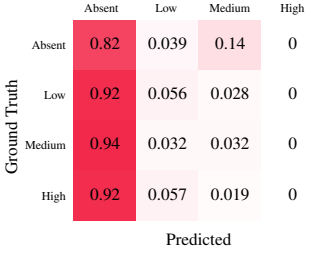
(c) *Confusion Matrix without transfer on P1 using solely the best source according to the IWCV metric.*



(d) *Confusion Matrix with transfer on P1 using all the sources.*

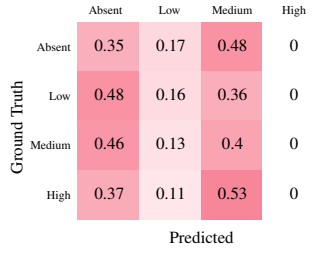


(e) *Confusion Matrix with transfer on P1 using the first two sources according to the IWCV metric.*

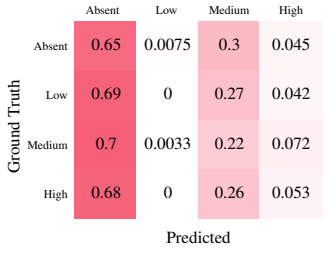


(f) *Confusion Matrix with transfer on P1 using solely the best source according to the IWCV metric.*

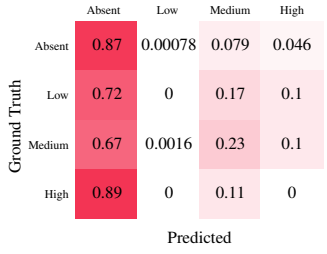
Figure 4.2: *Multi-Class Confusion Matrices on P1.*



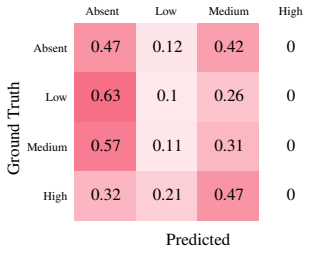
(a) *Confusion Matrix without transfer on P2 using all the sources.*



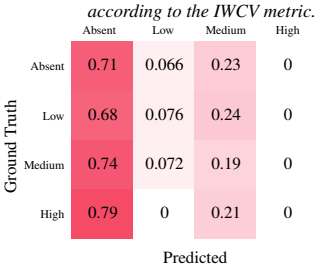
(b) *Confusion Matrix without transfer on P2 using the first two sources according to the IWCV metric.*



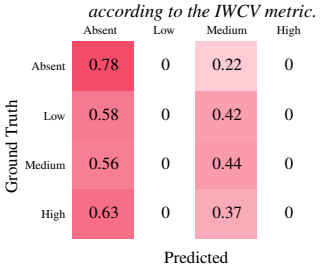
(c) *Confusion Matrix without transfer on P2 using solely the best source according to the IWCV metric.*



(d) *Confusion Matrix with transfer on P2 using all the sources.*

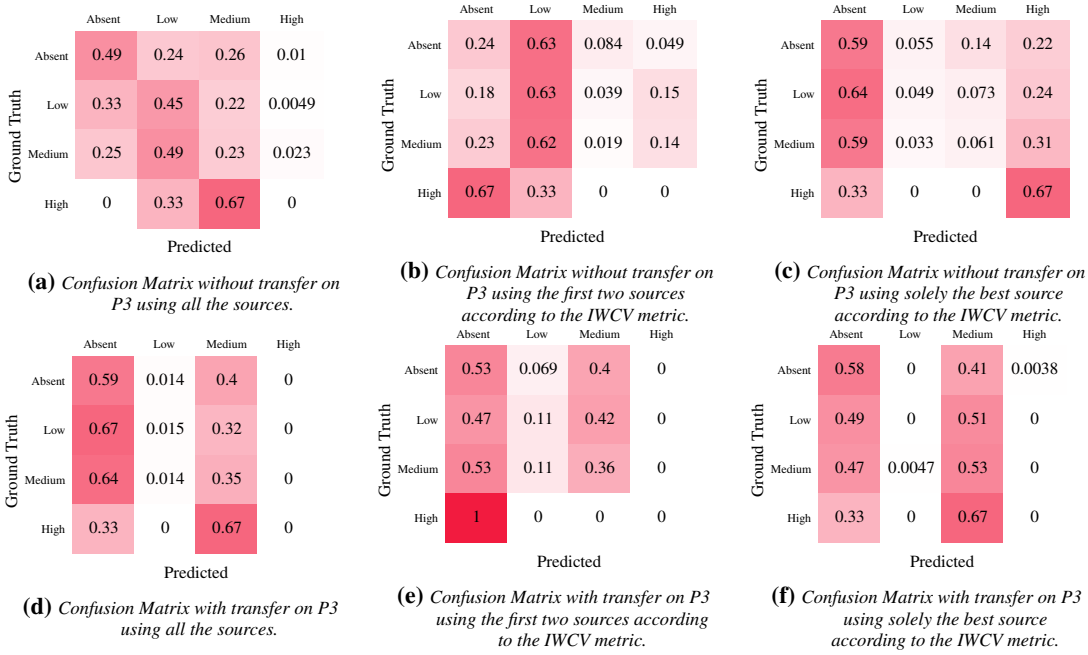


(e) *Confusion Matrix with transfer on P2 using the first two sources according to the IWCV metric.*



(f) *Confusion Matrix with transfer on P2 using solely the best source according to the IWCV metric.*

Figure 4.3: *Multi-Class Confusion Matrices on P2.*



(a) *Confusion Matrix without transfer on P3 using all the sources.*

(b) *Confusion Matrix without transfer on P3 using the first two sources according to the IWCV metric.*

(c) *Confusion Matrix without transfer on P3 using solely the best source according to the IWCV metric.*

(d) *Confusion Matrix with transfer on P3 using all the sources.*

(e) *Confusion Matrix with transfer on P3 using the first two sources according to the IWCV metric.*

(f) *Confusion Matrix with transfer on P3 using solely the best source according to the IWCV metric.*

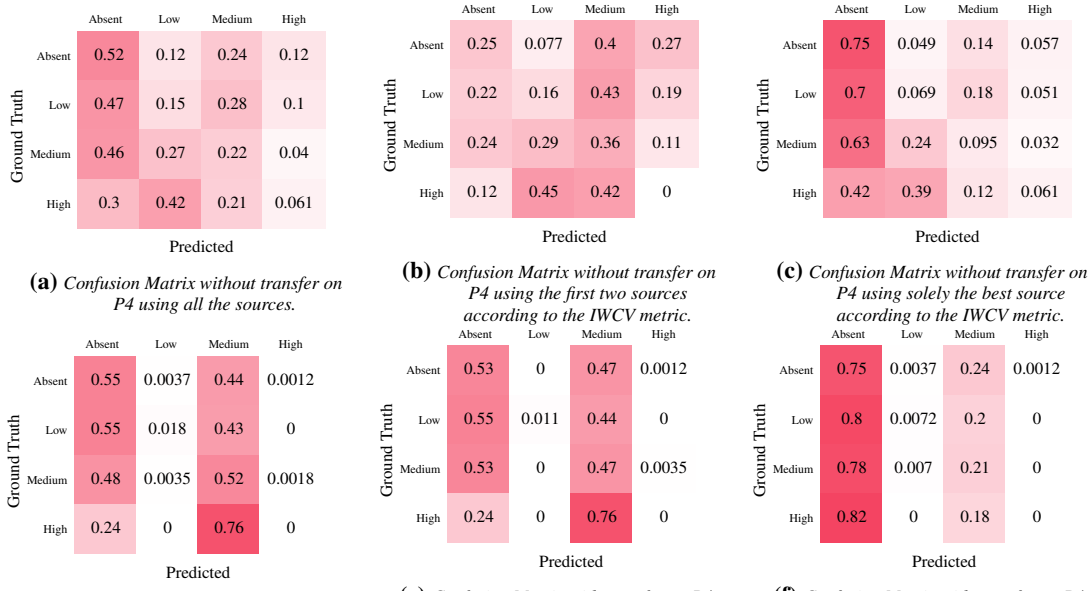
Figure 4.4: *Multi-Class Confusion Matrices on P3.*

Pipeline P3

Comparing Figure 4.4a with 4.4d ($B = 3$) and Figure 4.4b with 4.4e ($B = 2$), we may see as the TL technique improves the absent and medium classes recognition at the cost of increasing misclassifications within the low class. Differently, when comparing Figure 4.4c with 4.4f ($B = 1$), we have the same behavior as before with also an increment on the misclassification error for the class high. In the context of the binarized versions (please see Figures A.3a, A.3b, A.3c compared with A.3d, A.3e, A.3f, respectively, in Appendix A), we have that both the confusion matrices represented in Figures A.3e and A.3f are meaningful. Also in the case of this target pipeline, we have that applying transfer almost always improves accuracy and F1-Score (see Tables 4.3 and 4.4).

Pipeline P4

As we can see from Figure 4.5a compared with 4.5d ($B = 3$) and Figure 4.5b compared with 4.5e ($B = 2$), applying the TL technique allows us to improve the recognition performance both on the absent and medium classes with a degradation effect on the low class (in Figure 4.5d a slight degradation on the high class is perceivable w.r.t. Figure 4.5a). In the context of Figures 4.5c and 4.5f ($B = 1$), we have the same behavior described for Figures 4.5a and 4.5d. Here, the only difference is that the absent class true positive rate remains the same. For what concern the binarized version (see Figure A.4 in Appendix



(a) *Confusion Matrix without transfer on P4 using all the sources.*

(b) *Confusion Matrix without transfer on P4 using the first two sources according to the IWCV metric.*

(c) *Confusion Matrix without transfer on P4 using solely the best source according to the IWCV metric.*

(d) *Confusion Matrix with transfer on P4 using all the sources.*

(e) *Confusion Matrix with transfer on P4 using the first two sources according to the IWCV metric.*

(f) *Confusion Matrix with transfer on P4 using solely the best source according to the IWCV metric.*

Figure 4.5: *Multi-Class Confusion Matrices on P4.*

A), in the context of this target pipeline, choosing the best source ($B = 1$) w.r.t. IWCV does not produce a meaningful confusion matrix, whereas the other two approaches do (please compare Figures A.4a, A.4b, A.4c with A.4d, A.4e, A.4f, respectively). Finally, as we may see from Tables 4.3 and 4.4, the TL technique always improves the two figures of merit surpassing or matching the performances obtained by training a model onto a portion of the target pipeline and then testing it onto the held out test set (i.e., the Supervised Oracle).

4.4 Discussion

In the previous section, we have seen how the three configurations ($B = 1, 2, 3$) of the proposed solution behaved in terms of corrosion classification onto a target pipeline for which the labels are not available. The lack of labels was bridged with the aid of a TL technique described in Section 4.2.1. We have shown models' performance along four distinct components: the multi-class confusion matrices, their binarized version (to assess the detection capabilities of the developed models), the accuracy, and the F1-Score. Among the three different configurations, the one built using the first two best sources ($B = 2$) according to IWCV seemed to be the most stable, especially concerning the binarized version of the confusion matrices. This is a reasonable solution to trade off the intrinsic uncertainty in identifying the best source pipeline with the need to limit the number of sources to be used.

In the context of corrosion classification for oil and gas pipelines (the tackled application scenario), it would be also useful to get access to the confidence with which the model is predicting a certain label. To achieve this, for what concerns SVMs, we can capture the probabilities for each class given a fixed sample. More precisely, multiclass probability estimates are computed by combining all pairwise probability estimates $r_{i,j}$ for class i and j. So, given all the $r_{i,j}$, the estimate of $p(y = i|x)$ is obtained by solving a linear system, see Wu et al. [2004] for further details. The pairwise probability estimates are obtained by logistic regression on the score of the SVM [Platt et al., 1999]. It is noteworthy to point out that this technique to produce probabilities can be seamlessly integrated into the proposed TL approach.

4.5 Conclusion

In this *Chapter*, we tackled the problem of building a predictive model for the corrosion phenomenon in the context of a pipeline infrastructure for which the supervised information is not available (see Section 1.1 for a summary of the original contributions of this dissertation). To achieve this goal we used a KMM-based TL technique to leverage labeled data coming from a source pipeline infrastructure. Moreover, in a context where different labeled source pipelines are available, we combined KMM, IWCV, and multitask learning to produce a model for the target infrastructure by selecting the appropriate sources.

As possible future directions, we would like to mention the need to acquire time-varying data for a set of fixed pipelines to develop models accounting for time variations of the corrosion phenomenon both in the TL context and in the supervised learning one. Furthermore, taking into account the pipelines' material and chemical composition of the blend flowing within the pipelines is another important future direction. As of right now, the material is homogeneous among our pipelines and we do not possess data describing the chemical composition of the blend flowing through them. However, it would be relevant to extend our methodology in this direction, maybe giving more importance to those source pipelines having similar material and chemical composition of the blend w.r.t. the target pipeline.

Part II

Reinforcement Learning in Non-Stationary Environments

CHAPTER 5

Introduction and Related Literature

5.1 Introduction

In the previous *Chapters*, we have dealt with the problem of corrosion in pipeline infrastructures which constitutes an intrinsic non-stationary phenomenon. Unfortunately, the time-variability of corrosion could not be taken into account due to a lack of data representing the phenomenon through time. However, this has motivated us to investigate, in different ML sub-fields, new solutions able to weaken the stationarity assumption learning algorithms usually come equipped with. In this Part of the dissertation, we will focus on RL tackling the second contribution of this dissertation (see Section 1.1). Indeed, RL literature [Sutton and Barto, 2011] usually assumes the task assigned to the agent to be stationary. This assumption is not likely to hold in real-world applications, where the system to be controlled may be subject to different variations over time [Padakandla, 2021]. For instance, in the context of finance, applying RL under the assumption of stationary markets would impair the performance of our agent in the long run due to seasonality or market evolution usually intrinsic to this kind of scenario. Similarly, while controlling a water reservoir system, the agent must be able to take into account shifts to the system's dynamics induced through the decades by climate change [Giuliani et al., 2016]. Finally, also in the context of robotic systems, stationarity assumptions could impair the attainable performances because the agent is not prepared to deal with faults affecting sensors or

actuators.

Being able to relax the stationarity assumption would both highly increase the applicability of RL in real-world scenarios and allow us to build more sophisticated agents. To this end, in *Chapter* 6, we will present an active-adaptive approach that is able to endow an RL agent with the capability to detect and promptly react to non-stationarities affecting its performance [Canonaco et al., 2020a] hence producing greater average returns w.r.t. a standard RL algorithm whenever this kind of non-stationarity occurs. This handles non-stationarities associated with the current task being tackled. However, while solving an RL problem, there could be available some historical knowledge that could be leveraged via TL in order to speed up the learning phase on the current target task. This available historical knowledge could be produced by a non-stationary process, hence requiring to account for an intrinsic time-variant structure when transferring it to the current task. This last setting is never considered by the related literature. Therefore, in *Chapter* 7, we will propose a TL algorithm for RL able to account for such time-variant structures intrinsic to the available historical knowledge [Canonaco et al., 2021d]. Accounting for this time variance will allow our proposed solution to outperform its time-invariant counterpart [Tirinzoni et al., 2018a] that represents a state-of-the-art algorithm from the TL for RL literature.

In the remainder of this *Chapter*, we will review the related literature in the context of non-stationary RL.

5.2 Related Literature

In recent years, a range of solutions designed to deal with non-stationarity in RL have been proposed. An extension of the MDP model [Puterman, 2014], called Hidden Mode Markov Decision Process (HMMDP), where non-stationarity is modeled through a Hidden Markov Chain is proposed by Choi et al. [2000]. The HMMDP is learned through a variant of the Baum-Welch algorithm. In such a scenario, once a model of the environment, including the non-stationary dynamics, is learned, traditional model-based RL techniques could be considered. Unfortunately, the number of working modes or, equivalently, the number of changes in the environment must be known a priori. This assumption rarely holds in the real world. In order to overcome the assumption of having a fixed and a priori known number of environment changes, a solution based on the estimation of both the reward function and the environment transition-function was proposed by Da Silva et al. [2006]. The main drawback of this solution lies in the way it performs the change detection, which is not theoretically-grounded (i.e., a heuristic is proposed) and based on several problem-dependent parameters. The previously mentioned solution was extended by using a CUmulative SUM (CUSUM) [Basseville et al., 1993] sequential statistical test to perform the change detection in the environment [Hadoux et al., 2014]. Unfortunately, this solution is meant to operate on finite MDPs. Furthermore, along the rationale of

Da Silva et al. [2006] and Hadoux et al. [2014], Alegre et al. [2021] developed a model-based RL algorithm able to deal with continuous non-stationary environments, whereas Padakandla et al. [2020] proposed an extension to Q-Learning [Watkins and Dayan, 1992] that is able to detect context changes. The latter solution, in addition to being designed for finite MDPs, it requires to know the change pattern among models in order to update the correct Q-function when the context change is detected.

In order to deal with non-stationarity in RL, Sutton et al. [2007] proposed tracking. This technique is based on customizing the policy to the current situation via an adaptive learning rate. In other words, this solution can meta-learn the step-size to adequately adapt to the scenario it is facing. This technique does not detect the environment change, but simply mitigates the potentially catastrophic effect of non-stationarity on the algorithm performance.

Repeated Update Q-Learning (RUQL) [Abdallah and Kaisers, 2016] is another way to indirectly address non-stationarity in the RL task. This algorithm aims at solving the policy-bias problem of Q-Learning [Watkins and Dayan, 1992], which severely injures performance in a particularly noisy or non-stationary environment. A different line of research has been recently pursued by Gajane et al. [2018] and Ortner et al. [2019]. These solutions are based on a passive adaptation to the changing environment through a sliding window. However, a fixed number of changes is assumed by Gajane et al. [2018], whereas an upper-bound on the amount of variation the reward function (or the state-transition function) can undergo is assumed to be known by Ortner et al. [2019]. Along this line of research, Cheung et al. [2020] devised a sliding window approach to RL in non-stationary MDPs together with a bandit over RL framework to remove the dependency of their algorithm on the variation budget. Moreover, Gajane et al. [2018], Ortner et al. [2019], and Cheung et al. [2020] only deal with finite-state MDPs.

Domingues et al. [2021], instead, propose an algorithm where time-dependent kernels are leveraged in order to recover a regret upper bound for continuous non-stationary environments and Chandak et al. [2020b] propose a policy gradient algorithm which strives to optimize the future performance of the policy assuming that smooth changes in the environment imply smooth changes in a given policy performance. Furthermore, Chandak et al. [2020a], building on Chandak et al. [2020b], propose an RL algorithm that strives to ensure safety, intended as an high-probability performance improvement w.r.t. an already deployed policy, in non-stationary MDPs. Additionally, Lecarpentier and Rachelson [2019] introduce the concept of Lipschitz Continuous Non-Stationary MDP for which they propose a planning algorithm that aims to be robust w.r.t. the worst-case evolution of the MDP itself.

Finally, fast-adaptation algorithms based on meta-learning [Al-Shedivat et al., 2017, Nagabandi et al., 2018] are also able to deal with non-stationary environments. Anyhow, these algorithms assume to have access to the distribution over the tasks to face (through which they construct a meta-model able to nearly immediately adapt to new tasks coming

from this distribution).

CHAPTER 6

Model-Free Non-Stationarity Detection and Adaptation in Reinforcement Learning

In this *Chapter*, we present a theoretically-grounded change-detection mechanism to detect non-stationarities related to performance degradation during the learning process of any RL algorithm. This tool, called NSD-RL, is able to detect changes in both the reward function and in the state transition function of the task the RL algorithm is learning, as long as the non-stationarity implies a variation in the agent performance.[1] More specifically, NSD-RL is hierarchically composed of two main modules. The lower module is a statistical hypothesis test coupled with an IS [Hesterberg, 1988, Fishman, 2013] strategy, where the IS strategy will be used to transform two independent batches of data to allow the change detection. The upper module sequentially analyses the outcome of the low-level module by means of a CUSUM approach to detect non-stationarities in the RL problem. The proposed NSD-RL allows to trigger, following the active-adaptive approach introduced by Ditzler et al. [2015], the reaction to the detected change and the adaptation of the RL algorithm by resetting the first and second moments of the Adam [Kingma and Ba, 2014] optimizer. The effectiveness of the proposed NSD-RL is tested in two well-known continuous control RL tasks under the effect of different kinds of non-stationarities.

[1] We are not interested in non-stationarities not affecting the agent's performance since they are not so crucial in a real-world scenario.

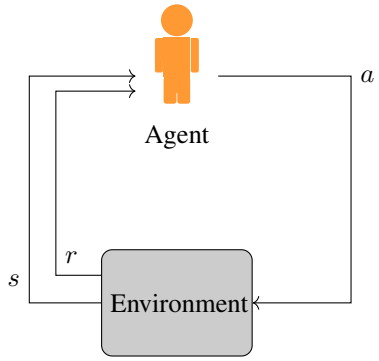


Figure 6.1: *Reinforcement Learning paradigm.*

The content of this *Chapter* is taken from Canonaco et al. [2020a], where I am the main researcher, and is organized as follows. In Section 6.1, we describe the preliminaries and formulate the RL problem in non-stationary environments. In Section 6.2, we introduce the policy selection to support non-stationarity detection, while Section 6.3 details the optimization of the Renyi divergence for policy selection. The change-detection and adaptation mechanism for RL is introduced in Section 6.4. Experimental results are given in Section 6.5 and conclusions are drawn in Section 6.6.

6.1 Preliminaries and Problem Formulation

In this section we initially introduce the RL framework needed in the context of this *Chapter*; we then provide the theoretical background of the IS technique and we conclude by providing a formulation of the problem of RL in non-stationary environments.

6.1.1 Reinforcement Learning Background

In a generic RL setting [Sutton and Barto, 1998], we have an agent that interacts with the environment (see Figure 6.1). The agent, while being in a certain state of the environment, executes an action, receives a certain reward associated with that action, and observes the next state of the environment. This setting can be modeled via a discrete-time continuous MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, p_0\}$ [Puterman, 2014], where $\mathcal{S}$ and $\mathcal{A}$ represent the state space and the action space, respectively, either continuous or discrete. $\mathcal{P}$ represents the Markovian transition function, where $\mathcal{P}(s'|s,a)$ is the transition density from state s to state s' given that the action a is executed. $r = \mathcal{R}(s,a) \in [-R, R]$ represents the expected reward for the pair (s,a). $\gamma \in [0,1)$ and p_0 are the discount factor and the initial state distribution, respectively. The agent's behavior is modeled through a policy π, where $\pi(\cdot|s)$ describes a probability density function over the action space $\mathcal{A}$ given the state being currently visited.

We are considering episodic MDPs with effective horizon H, hence a trajectory τ can be a finite sequence of states and actions $(s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1})$, where $s_0 \sim p_0$. Following a given policy, we can sample a trajectory from the environment. We denote by $p(\tau|\pi)$ the density distribution induced by policy π on the set $\mathcal{T}$ of all the possible trajectories defined as:

$$p(\tau|\pi) = p_0(s_0)\pi(a_0|s_0)\prod_{t=1}^{H}\mathcal{P}(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t).$$

$\mathcal{R}(\tau) = \sum_{t=0}^{H-1}\gamma^t\mathcal{R}(s_t, a_t)$ is the total discounted reward associated with trajectory τ. All policies can be ranked based on their expected total discounted reward: $J(\pi) = \mathbb{E}_{\tau\sim p(\cdot|\pi)}[\mathcal{R}(\tau)]$. Solving an RL task modeled through an MDP $\mathcal{M}$ means finding $\pi^* \in \arg\max_\pi\{J(\pi)\}$.

The most interesting application of NSD-RL refers to the *continual learning setting*. In this setting, Policy Gradient (PG) [Sutton et al., 2000] techniques offer general and flexible solutions to RL problems and, for this reason, we will rely on this family of algorithms to present the proposed solution.

Policy gradient methods focus on searching for the best-performing policy over a set of parametrized policies $\Pi_\theta = \{\pi_\theta : \theta \in \mathbb{R}^d\}$, with π_θ differentiable w.r.t. θ. For brevity, the performance of a parametric policy will be denoted by $J(\theta)$ or equivalently by J_θ. Furthermore, the probability of a trajectory τ will be denoted by $p(\tau|\theta)$ or equivalently by $p_\theta(\tau)$ (on some occasions, $p_\theta(\tau)$ will be replaced by p_θ omitting the dependence on τ for the sake of readability). Gradient ascent is used to find a locally optimal policy. The policy gradient is defined as [Sutton et al., 2000, Peters and Schaal, 2008]:

$$\nabla J(\theta) = \mathop{\mathbb{E}}_{\tau\sim p(\cdot|\theta)}\left[\nabla \log p_\theta(\tau)\mathcal{R}(\tau)\right]. \tag{6.1}$$

At each iteration $i > 0$, a batch $\mathcal{D}_i^N = \{\tau_j\}_{j=0}^N$ of $N > 0$ trajectories is collected using policy π_{θ_i}. The policy is then updated as $\theta_{i+1} = \theta_i + \alpha\widehat{\nabla}_N J(\theta_i)$, where α is a step size and $\widehat{\nabla}_N J(\theta)$ is an estimate of Eq. (6.1) on $\mathcal{D}_i^N$, i.e.,

$$\widehat{\nabla}_N J(\theta) = \frac{1}{N}\sum_{j=1}^{N} g(\tau_j|\theta), \quad \tau_j \in \mathcal{D}_i^N, \tag{6.2}$$

where $g(\tau_j|\theta)$ is an estimate of $\nabla \log p_\theta(\tau_j)\mathcal{R}(\tau_j)$. Depending on how we define the policy gradient estimator, we obtain different RL algorithms.

6.1.2 Importance Sampling

The idea behind the proposed NSD-RL is to evaluate the performance of a fixed policy using a pair of estimators fed with data coming from two different iterations of the RL

algorithm. Therefore, we need a mechanism to take into account the fact that the data were sampled with different policies.

Let us assume that we want to find $\mathbb{E}_{x\sim P}[f(x)] = \int_{\mathcal{D}} f(x)p(x)dx$ where p is a probability density function on $\mathcal{D} \subseteq \mathbb{R}^d$ with $p(x) = 0,\ \forall x \notin \mathcal{D}$, then:

$$\int_{\mathcal{D}} f(x)p(x)dx = \int_{\mathcal{D}} \frac{f(x)p(x)}{q(x)} q(x)dx = \mathbb{E}_{x\sim Q}\left[\frac{f(x)p(x)}{q(x)}\right], \tag{6.3}$$

where q is a positive probability density function on $\mathbb{R}^d$ such that $supp(q) \supset supp(p)$ and $\mathbb{E}_{x\sim Q}[\cdot]$ denotes expectation for $x \sim Q$. The IS technique introduces a multiplicative correction coefficient that compensates the fact that we are sampling from Q instead of sampling directly from P.

Importance sampling allows off-policy evaluation in RL [Thomas et al., 2015, Thomas and Brunskill, 2016]. In the off-policy evaluation settings, two policies, called behavioral π^B and target π^T, are involved. In this context, we aim at estimating the performance of the target policy π^T on samples collected using the policy π^B. We use IS to correct the fact that we sampled the trajectories using π^B and obtain an unbiased estimate of $J(\pi^T)$:

$$J(\pi^T) = \mathop{\mathbb{E}}_{\tau\sim p(\cdot|\pi^T)}[\mathcal{R}(\tau)] = \mathop{\mathbb{E}}_{\tau\sim p(\cdot|\pi^B)}\left[\omega_{T/B}(\tau)\mathcal{R}(\tau)\right], \tag{6.4}$$

where $\omega_{T/B}(\tau) = \frac{p(\tau|\pi^T)}{p(\tau|\pi^B)} = \prod_{t=0}^{H} \omega_{T/B}(s_t, a_t)$ and $\omega_{T/B}(s_t, a_t) = \frac{\pi^T(a_t|s_t)}{\pi^B(a_t|s_t)}$. In the context of our NSD-RL, we will use the *per-decision* version of the above IS estimator that exploits the fact that a given reward should not be weighted according to the future of that trajectory, but only w.r.t. the likelihood of the trajectory up to that point [Precup et al., 2000].

6.1.3 Non-Stationarity in Reinforcement Learning: Problem Formulation

We can model non-stationarity in any RL task through a change in the Markovian state transition function $\mathcal{P}$ or in the reward function $\mathcal{R}$. In the most general scenario, $\mathcal{P}$ and $\mathcal{R}$ may be both affected by non-stationarity (possibly at the same time). Therefore, there exists an iteration i^* of the RL algorithm after which the trajectories sampled from the environment will be, partially or totally, associated to a new task. All the trajectories in i^* are associated with the new task if the transition to this new task takes place at the beginning of the sampling procedure performed at step i^* of the RL algorithm itself, otherwise, just a subset of them will be associated to the new task. Since NSD-RL is not influenced by the latter situation we have just described, we will formulate non-stationarity in RL by assuming that the change in either $\mathcal{P}$ or $\mathcal{R}$ (or both) occurs at the beginning of the sampling process of a given optimization step of the algorithm. In other words, we can formalize non-stationarity in RL tasks as follows: for any $i < i^*$ we have $\mathcal{M}_1 =$

$\{\mathcal{S}, \mathcal{A}, \mathcal{P}_1, \mathcal{R}_1, \gamma, p_0\}$, whereas for $i \geq i^*$ we have $\mathcal{M}_2 = \{\mathcal{S}, \mathcal{A}, \mathcal{P}_2, \mathcal{R}_2, \gamma, p_0\}$, where $\mathcal{P}_1 \neq \mathcal{P}_2 \vee \mathcal{R}_1 \neq \mathcal{R}_2$. We are assuming that p_0 is not affected by the non-stationarity.

The goal of the proposed NSD-RL is to promptly detect changes in $\mathcal{M}$ without introducing false positive or negative detections. Such a change-detection represents also a crucial information to support the next adaptation and learning phase of the RL algorithm.

6.2 Policy Selection to support Non-Stationarity Detection

As mentioned in Section 6.1.3, if the task we are trying to solve is non-stationary, then it may happen that, between steps i and $i-1$, $\mathcal{P}$ or $\mathcal{R}$ (or both) may change. The first question we have to address is how to detect a change that can occur at any time during the execution of our RL algorithm? Answering this question is crucial to support an effective reaction and adaptation of the policy in a non-stationary environment. In order to reach our goal, we first need to be able to detect a change between two arbitrary fixed steps of the RL algorithm. Therefore, given data collected through policy π_{θ_i} at step i and data collected through policy $\pi_{\theta_{i-k}}$ at step $i-k$, we need to test $\mathcal{H}_0$*: there is no change in* $\mathcal{M}$ *between* i *and* $i-k$ against $\mathcal{H}_1$*: there is a change in* $\mathcal{M}$ *between* i *and* $i-k$. Before resorting to a statistical test, we need to spot the figure of merit on which to apply the test. This figure of merit is meant to operate on two independent data sets available in the two different steps of the RL algorithm. Therefore, by using the IS technique described in Section 6.1.2, we can write:

$$\bar{J}(\pi_\mu) = \mathop{\mathbb{E}}_{\tau \sim p(\cdot|\pi_\mu)} \left[\bar{\mathcal{R}}(\tau)\right] = \mathop{\mathbb{E}}_{\tau \sim p(\cdot|\pi_{\theta_i})} \left[\omega_{p_\mu / p_{\theta_i}}(\tau)\bar{\mathcal{R}}(\tau)\right] \tag{6.5}$$

$$\bar{J}(\pi_\mu) = \mathop{\mathbb{E}}_{\tau \sim p(\cdot|\pi_\mu)} \left[\bar{\mathcal{R}}(\tau)\right] = \mathop{\mathbb{E}}_{\tau \sim p(\cdot|\pi_{\theta_{i-k}})} \left[\omega_{p_\mu / p_{\theta_{i-k}}}(\tau)\bar{\mathcal{R}}(\tau)\right], \tag{6.6}$$

where $\bar{\mathcal{R}}(\tau) = \sum_{t=0}^{H-1} \mathcal{R}(s_t, a_t)$ is the expected total undiscounted reward. If there are no changes in $\mathcal{M}$ between iteration i and $i-k$, the two expected values are equal, otherwise they are different.[2] Since we cannot exactly compute the expected values in Eq. (6.5) and (6.6), we resort to the associated estimators:

$$\hat{\bar{J}}_{\mu/\theta} = \frac{1}{N} \sum_{j=0}^{N} \omega_{\mu/\theta}(\tau_j)\bar{\mathcal{R}}(\tau_j), \tag{6.7}$$

where N is the number of trajectories sampled using π_θ. Then, our goal is to properly choose the policy π_μ in order to have an effective hypothesis test able to detect changes in $\mathcal{M}$ between iteration i and $i-k$. From Metelli et al. [2018] we know that:

$$\mathbb{V}ar\left[\hat{\bar{J}}_{\mu/\theta}\right] \leq \frac{1}{N} ||\mathcal{R}||_\infty^2 d_2(p_\mu || p_\theta), \tag{6.8}$$

[2]Notice that if the transition between the tasks happens in the middle of the sampling procedure at step i, the two expected values will still be different.

Table 6.1: *Estimated type I error of the bootstrap test [Efron and Tibshirani, 1993, chap. 16] under $\mathcal{H}_0$ w.r.t. different choices of π_μ. The two sampling policies, $\pi_{\theta_{i-k}}$ and π_{θ_i}, are $\mathcal{N}$(10,13) and $\mathcal{N}$(-1,4), respectively. First row is associated with policy π_μ chosen optimizing Equation (6.9).*

π_μ	Mean Type I error	Std. Dev.	Obj. Fun. (6.9)
$\mathcal{N}$(-0.3487, 4.846)	0.0462	0.0206	3.95
$\mathcal{N}$(4, 4.5)	0.0578	0.0206	9.21
$\mathcal{N}$(8, 2)	0.109	0.0315	31.96
$\mathcal{N}$(8, 5)	0.1687	0.0333	128237.5

where $d_2(p_\mu||p_\theta)$ is the exponetiated Renyi divergence [Rényi, 1961] of the distribution induced by policy π_μ from the distribution induced by policy π_θ. Therefore, choosing the policy π_μ such that:

$$\pi_\mu^* \in \arg\min_{\pi_\mu} d_2(p_\mu||p_{\theta_i}) + d_2(p_\mu||p_{\theta_{i-k}}) \tag{6.9}$$

will allow us to minimize the upper bound on the variance of the estimators of Eq. (6.5) and (6.6), hence increasing the power of our hypothesis test.

The optimization task in Eq. (6.9) aims at picking a policy π_μ such that we do not have unbounded weights when using IS, which means that the estimators of Eq. (6.5) and (6.6) converge smoothly in the number of samples. Note that having unboundend weights might induce the estimators of Eq. (6.5) and (6.6) to abruptly change as we increase the number of samples [Robert and Casella, 2013, chap. 3]. In this case, we cannot rely on the smooth convergence properties of the estimators. In other words, we will likely end up with an estimate very far from the real value, which in turn severely affects the ability of the hypothesis test to keep the type I error under control (see Table 6.1). We should also observe that if the two sampling policies, π_{θ_i} and $\pi_{\theta_{i-k}}$, are very far from each other (in terms of Renyi divergence) there will be no policy π_μ able to induce good behavior in the IS procedure (an example of this behavior is experimentally given in Table 6.2). Observe that, in order to produce both Table 6.1 and 6.2, we have used the *"Guess a Number"* task. This is a single state task where the reward function is $\mathcal{R}(a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(a-\mu)^2}{2\sigma}}$ for a fixed μ and σ which define the task. The agent will get higher rewards executing actions which are as close as possible to μ. Therefore, the optimal policy consists in always playing $a = \mu$. As an example, in the context of the above described experiments, we have $\mathcal{R}(a) = \frac{1}{10\sqrt{2\pi}} e^{-\frac{(a+4)^2}{20}}$. This setting, while being rather simple, already confirms the complexity of selecting an evaluation policy without side effects on the type I error.

We emphasize that the estimators for Eq. (6.5) and (6.6) share the expected value but they are characterized by different and unknown probability distributions. In fact, we do not have any a priori information about the family of probability distributions the estimators for Eq. (6.5) and (6.6) belong to, e.g., we cannot assume they are Gaussians. Hence, in

Table 6.2: *Estimated type I error of the bootstrap test [Efron and Tibshirani, 1993, chap. 16] under $\mathcal{H}_0$ increasing the distance of policy $\pi_{\theta_{i-k}}$ from π_{θ_i} which instead remains fixed to $\mathcal{N}$(-1,4). The mean policy π_μ is always chosen optimizing Equation (6.9).*

π_{θ_i}	Mean Type I error	Std. Dev.	Obj. Fun. (6.9)
$\mathcal{N}(1, 13)$	0.0478	0.0231	3.12
$\mathcal{N}(10, 13)$	0.0462	0.0206	3.95
$\mathcal{N}(35, 13)$	0.0334	0.017	47.23
$\mathcal{N}(50, 13)$	0.0087	0.009	586.86
$\mathcal{N}(100, 13)$	0	0	7931313070349.74

order to support our analysis, we need a statistical hypothesis test able to detect variations in the expected value without making any assumption about the underlying probability distributions. We emphasize that not satisfying the assumptions of the statistical hypothesis test would induce critical issues in controlling the type I error. In our specific case, such a problem would severely affect the sequential analysis characterizing NSD-RL that will be described in Section 6.4. For these reasons, we resort to a bootstrap hypothesis test proposed by Efron and Tibshirani [1993, chap. 16] which, being a test for detecting variations in the mean of two arbitrary distributions, perfectly fits into the context we are working with. Hence, in the scenario of RL in non-stationary environments, we define the hypothesis test as a function

$$Te(\mathcal{D}^N_{\mu/\theta_i}, \mathcal{D}^N_{\mu/\theta_{i-k}}, \alpha) = \begin{cases} +1, & \text{if we reject } \mathcal{H}_0 \\ -1, & \text{otherwise} \end{cases} \tag{6.10}$$

where $\mathcal{D}^N_{\mu/\theta_i}$ and $\mathcal{D}^N_{\mu/\theta_{i-k}}$ are the data sets sampled at iterations i and $i-k$, respectively, after applying IS, and $\mathcal{H}_0$ represents the null hypothesis, i.e., the two expected values are equal. If the output of this function is +1, then a change is detected between iterations i and $i-k$ of the RL algorithm given a confidence level α otherwise no change occurred. We would like to stress the fact that NSD-RL is able to detect only changes affecting the performance of the agent, here represented by the expected total undiscounted reward. However, changes not affecting the performance are less relevant to be detected in a real-world scenario.

6.3 Renyi Divergence Optimization

In the previous section, we defined an objective function aiming at selecting the policy π_μ to be used in the hypothesis test defined in Eq. (6.10). In this section, we propose a way to optimize such an objective function. Notice that computing the Renyi divergence between two distributions over trajectories is, of course, intractable also given the transition density

of the task we are currently solving. For this reason the following estimator was introduced in Metelli et al. [2018]:

$$\hat{d}_2(p_\mu||p_\theta) = \frac{1}{N}\sum_{j=1}^{N}\prod_{t=0}^{H-1} d_2(\pi_\mu(\cdot|s_{\tau_j,t})||\pi_\theta(\cdot|s_{\tau_j,t})). \tag{6.11}$$

If we plug Eq. (6.11) into the optimization problem stated in Eq. (6.9) we get:

$$\begin{aligned}\pi_\mu^* \in \arg\min_{\pi_\mu} \hat{d}_2(p_\mu||p_{\theta_i}) + \hat{d}_2(p_\mu||p_{\theta_{i-k}}) = \\ = \arg\min_{\pi_\mu} \frac{1}{N}\sum_{j=1}^{N}\left(\prod_{t=0}^{H-1} d_2(\pi_\mu(\cdot|s_{\tau_j,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau_j,t}))+ \right. \\ \left. \prod_{t=0}^{H-1} d_2(\pi_\mu(\cdot|s_{\tau_j,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau_j,t}))\right).\end{aligned} \tag{6.12}$$

Now we can solve a separate optimization problem for each trajectory since we have independent variables for each $s_{\tau_j,t}$:

$$\begin{aligned}\arg\min_{\pi_\mu}\left(\prod_{t=0}^{H-1} d_2(\pi_\mu(\cdot|s_{\tau,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau,t}))+ \right. \\ \left. \prod_{t=0}^{H-1} d_2(\pi_\mu(\cdot|s_{\tau,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau,t}))\right),\ \forall \tau \in \mathcal{D}_i^N \cup \mathcal{D}_{i-k}^N.\end{aligned} \tag{6.13}$$

We reformulated the problem in the following way to allow independent optimization over all the states $s_{\tau,t}$:

$$\begin{aligned}\arg\min_{\pi_\mu}\big(d_2(\pi_\mu(\cdot|s_{\tau,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau,t})) + d_2(\pi_\mu(\cdot|s_{\tau,t})||\pi_{\theta_{i-k}}(\cdot|s_{\tau,t}))\big), \\ \forall \tau \in \mathcal{D}_i^N \cup \mathcal{D}_{i-k}^N \forall t = 0 \cdots H-1.\end{aligned} \tag{6.14}$$

We emphasize that in RL with PG we have two popular choices for the set of smoothly parametrized policies Π_θ: Gibbs policies and Gaussian policies. The first one is generally used when we have a finite number of actions executable on the environment by the agent, while the second one is employed whenever the set of executable actions is infinite. Since we are dealing with finite-horizon MDPs, each trajectory τ, sampled from the environment, will have a maximum number H of visited states. For what concern Gaussian (or Gibbs) policies, we have a parametrized Gaussian (or Gibbs) distribution for which the

Renyi divergence can be computed analytically in each state. Since we have a per-state analytical form of the Renyi divergence, we will find the π_μ only for those states stored in $\mathcal{D}_i^N$ and $\mathcal{D}_{i-k}^N$. It is worth noting that we do not have the parametrized Gaussians (or Gibbs) generated by policy $\pi_{\theta_{i-k}}$ in the context of the states in $\mathcal{D}_i^N$ and the same holds for π_{θ_i} in the states in $\mathcal{D}_{i-k}^N$. However, the missing parametrizations can be computed straightforwardly without any interactions with the environment. Notice that π_μ is only used for non-stationarity detection, and not to perform actions on the environment. In Section 6.3.1, we will focus on Gaussian policies, whereas, in Section 6.3.2, we provide an optimal solution to the problem described in Eq. (6.14) in the context of Gibbs policies.

6.3.1 Optimization for Gaussian Policies

Once we fix a state s, we have an analytical expression for the two terms of the objective function in Eq. (6.14) [Burbea, 1984]:

$$d_2(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_{\theta_i},\Sigma_{\theta_i})) = \frac{|\Sigma_{\theta_i}|}{\sqrt{|\Sigma||\Sigma^*_{\theta_i}|}} e^{(\mu-\mu_{\theta_i})^\top \Sigma^{*-1}_{\theta_i}(\mu-\mu_{\theta_i})}$$

$$d_2(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_{\theta_{i-k}},\Sigma_{\theta_{i-k}})) = \frac{|\Sigma_{\theta_{i-k}}|}{\sqrt{|\Sigma||\Sigma^*_{\theta_{i-k}}|}} e^{(\mu-\mu_{\theta_{i-k}})^\top \Sigma^{*-1}_{\theta_{i-k}}(\mu-\mu_{\theta_{i-k}})},$$

where $|\cdot|$ denotes the determinant of a matrix, $\Sigma^*_{\theta_i} = 2\Sigma_{\theta_i} - \Sigma$ and $\Sigma^*_{\theta_{i-k}} = 2\Sigma_{\theta_{i-k}} - \Sigma$ assuming that both $\Sigma^*_{\theta_i}$ and $\Sigma^*_{\theta_{i-k}}$ are positive-definite. In order to match this assumption, we restrict the optimization procedure over all the possible solutions having a diagonal Σ. In this way, we only need to satisfy the constraints on the diagonal elements of Σ:

$$\sigma^{jj} < \min(\sqrt{2}\sigma^{jj}_{\theta_i}, \sqrt{2}\sigma^{jj}_{\theta_{i-k}}) \wedge \sigma^{jj} > 0 \; \forall j = 1 \dots dim(\mathcal{A}), \tag{6.15}$$

where $dim(\mathcal{A})$ denotes the action space dimension.[3] Now we can resort to any optimization procedure present in the literature provided that it supports bounds on the objective function domain.

6.3.2 Optimization for Gibbs Policies

Following the rationale of the previous section, we will show how to optimize the Renyi divergence with Gibbs policies on a per state basis. Given a state s, we have a parametrized categorical distribution assigning a certain probability to each action available in s. Therefore, denoting with $P_{\theta_i} = (p_{\theta_i,1}, \dots, p_{\theta_i,n})$ and $P_{\theta_{i-k}} = (p_{\theta_{i-k},1}, \dots, p_{\theta_{i-k},n})$ the categorical distributions parametrized by θ_i and θ_{i-k}, respectively, in state s we can write Equation (6.14) as follows:

[3]Notice that in a mono-dimensional action space the restricted optimization problem is equivalent to the not restricted one.

$$\underset{P_\mu}{\arg\min} \quad \sum_{h=0}^{n} \frac{p_{\mu,h}^2}{p_{\theta_i,h}} + \sum_{h=0}^{n} \frac{p_{\mu,h}^2}{p_{\theta_{i-k},h}} \tag{6.16}$$

$$subject\ to: \sum_{h=0}^{n} p_{\mu,h} = 1, \tag{6.17}$$

$$p_{\mu,h} \geq 0 \ \forall \ h. \tag{6.18}$$

If we do not take into account the constraint (6.18), we have a relaxed problem which can be solved by using Lagrangian multipliers:

$$L = \sum_{h=0}^{n} \frac{p_{\mu,h}^2}{p_{\theta_i,h}} + \sum_{h=0}^{n} \frac{p_{\mu,h}^2}{p_{\theta_{i-k},h}} + \lambda \left(\sum_{h=0}^{n} p_{\mu,h} - 1 \right), \tag{6.19}$$

taking derivatives we have:

$$\frac{\partial L}{\partial p_{\mu,h}} = 2 \left(\frac{1}{p_{\theta_i,h}} + \frac{1}{p_{\theta_{i-k},h}} \right) p_{\mu,h} + \lambda = 0 \ \forall \ h \tag{6.20}$$

$$\frac{\partial L}{\partial \lambda} = \sum_{h=0}^{n} p_{\mu,h} - 1 = 0, \tag{6.21}$$

and now solving:

$$p_{\mu,h} = \frac{1}{\left(\frac{p_{\theta_i,h} + p_{\theta_{i-k},h}}{p_{\theta_i,h} p_{\theta_{i-k},h}} \right) C} \tag{6.22}$$

$$\lambda = -\frac{2}{C}, \tag{6.23}$$

where

$$C = \sum_{h=0}^{n} \frac{p_{\theta_i,h} p_{\theta_{i-k},h}}{p_{\theta_i,h} + p_{\theta_{i-k},h}}. \tag{6.24}$$

Since the objective function in Eq. (6.16)) is convex and the solution we have just found also satisfies the constraint in Eq. (6.18), then the solution we found is also optimal and unique for the non-relaxed problem.

6.4 Change-Detection and Adaptation Mechanism for Reinforcement Learning

The hypothesis test aiming at detecting a non-stationarity between step i and $i-k$ defined in Eq. (6.10) is here extended to operate sequentially by introducing a sequential change-detection mechanism based on the well-known and theoretically-grounded CUSUM [Basseville et al., 1993, Roveri, 2019]. More specifically, the proposed CUSUM-based NSD-RL mechanism operates as follows. Let $\bar{i}$ be the reference iteration initially set to a point where the agent has reached convergence. $\bar{i}$ represents the iteration at which we activate the NSD-RL change detection mechanism. The sequential analysis of $\mathcal{M}$ is performed over windows of length $2k$ (being $k \in \mathbb{N}^+$) and relies on the computation of a figure of merit m_i able to take into account the outcome of the hypothesis test $Te(\mathcal{D}^N_{\mu/\theta_i}, \mathcal{D}^N_{\mu/\theta_{i-k}}, \alpha)$ applied sequentially to detect a non-stationarity, i.e.,

$$m_i = max\left(0, m_{i-1} + Te(\mathcal{D}^N_{\mu/\theta_i}, \mathcal{D}^N_{\mu/\theta_{i-k}}, \alpha)\right), \tag{6.25}$$

with $i = \bar{i}+k, \ldots, \bar{i}+2k-1$ and being $m_{\bar{i}+k-1} = 0$. This allows us to take into account the first window ranging from $\bar{i}$ to $\bar{i}+2k-1$. Once we have analysied the first window, the algorithm switches to the next window and m_{i-1} withhold the last value of the figure of merit on the previous window. Notice that this window-based approach allows us to keep independent all the different tests we perform in the sequential analysis.

A change is detected at the i-th iteration when,

$$m_i \geq K, \tag{6.26}$$

being $K \in \mathbb{N}^+$ a change-detection threshold, which is set at design time. The choice of K is crucial to trade-off false positive detections (i.e., detections of changes before i^*) and false negative detections (i.e., changes are not detected by the change-detection mechanism). In our analysis, such a choice is supported by the theoretical analysis of the mean time to a false positive detection, i.e., the Average Run Length (ARL_0), provided in Roveri [2019] stating that

$$ARL_0(\alpha) = \underline{u}(I - P_\alpha)^{-1}\underline{1} \tag{6.27}$$

where I is the $(K+1)\times(K+1)$ identity matrix, P_α is the $(K+1)\times(K+1)$ matrix defined as follows

$$P_\alpha = \begin{bmatrix} 1-\alpha & \alpha & 0 & \ldots & 0 \\ 1-\alpha & 0 & \alpha & \ldots & 0 \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix},$$

$\underline{1}$ is the $(K+1)$-dimensional vector of ones, and $\underline{u}$ is the $(K+1)$-dimensional vector defined as $\underline{u} = [1, 0, \ldots, 0]$, being α the confidence level of the hypothesis test stated in Eq. (6.10). In our scenario $\text{ARL}_0(\alpha)$ refers to the mean number of executions of the

hypothesis test before the NSD-RL raises a false positive detection. Setting the expected ARL_0 (that is application-specific) allows to identify the corresponding value of the threshold K.

The proposed NSD-RL change-detection mechanism operating on a generic PG algorithm is shown in Algorithm 6. More specifically, Lines 4 and 21 refer to the PG implementation. Lines 5 - 20 implement the computation of the figure of merit m_i as described above. More precisely, in Line 6 we store the data in order to perform the hypothesis tests, in Line 9 we compute π_μ solving the optimization problem stated in Eq. (6.14), in Line 10 the hypothesis test $Te(\mathcal{D}^N_{\mu/\theta_i}, \mathcal{D}^N_{\mu/\theta_{i-k}}, \alpha)$ is evaluated and m_i is computed as described in Eq. (6.25), in Line 12 we move to the next window and in Line 13 we reset the buffer. Finally, in Line 15, the value of m_i is tested w.r.t. K to detect a change. Once the change has been detected, in Line 17 we reset all the configurations for the next change-detection phase and in Line 18 we react to the change to compensate as fast as possible the loss in performance. In the context of this work, we adopted a straightforward adaptation technique that consists in resetting the Adam [Kingma and Ba, 2014] optimizer by erasing its history related to the first and second moments, allowing it to forget what it currently knows about the behavior of the gradients in the previous task, which in turn implies a greater reactivity of the optimizer in the new task.

6.5 Experiments

In this section, we evaluate the improvement in performance of NSD-RL over G(PO)MDP [Baxter and Bartlett, 2001], which is a traditional non-adaptive RL algorithm. More precisely, G(PO)MDP is a refinement of REINFORCE [Williams, 1992] exploiting the fact that the current reward does not depend on future actions. In other words the gradient estimator of G(PO)MDP performs a proper credit assignment, which may imply a variance reduction on the gradient estimate itself. G(PO)MDP is coupled with the average discounted reward baseline to further reduce the gradient estimator variance. In order to make a fair comparison the learning algorithm used in PG-NSD-RL is also G(PO)MDP. Both PG-NSD-RL and G(PO)MDP use Adam [Kingma and Ba, 2014] as optimizer endowing them with an adaptive learning rate. The considered RL tasks are *Pendulum-v0* [Brockman et al., 2016] and *Mountain Car* [Duan et al., 2016] that are widely used RL tasks in the related literature. *Pendulum-v0* consists of a classical pendulum swing-up problem. The goal of the agent is to keep the pendulum in an upright position via the application of forces to the pendulum. The observation space is a 3-dimensional vector composed of $\cos\varphi$, $\sin\varphi$, and the pole velocity $\dot{\varphi}$. The monodimensional action is the force applied to the pendulum by the agent. The reward $\mathcal{R}(s,a) = -(\varphi^2 + 0.1\dot{\varphi}^2 + 0.001a^2)$. *Mountain Car*, instead, consists in escaping a valley via the application of limited tangetial forces. Due to this limitation, the car has to alternately drive up along the two slopes of the valley in order to gain sufficient momentum to overcome gravity. The observation space is

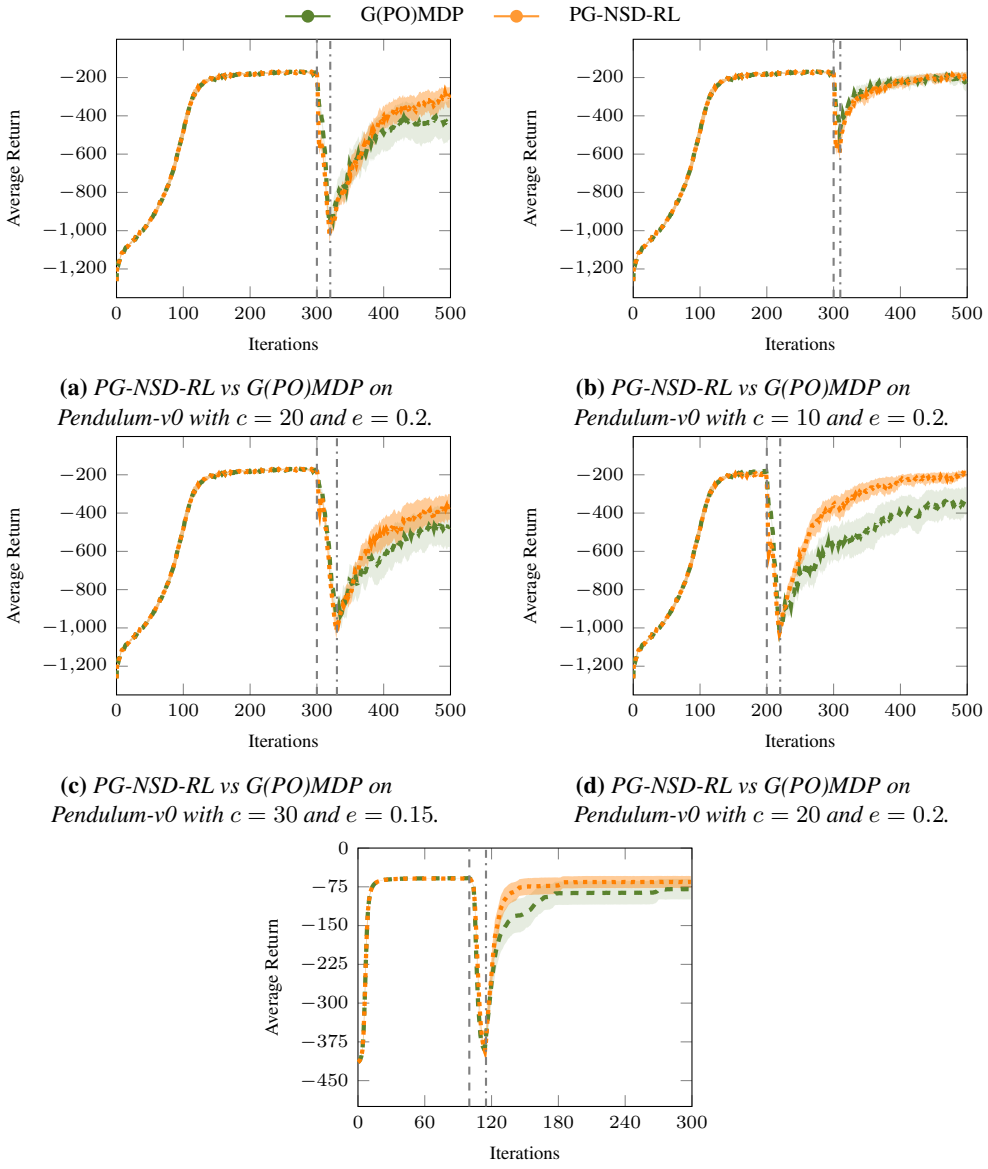
Algorithm 6 PG-NSD-RL

Input: change-detection threshold K, confidence level α, step size η, policy initialization θ_0, batch size N, number of epochs I, reference epoch $\bar{i}$, distance between epochs k
$B = \{\phi\}$
for $i = 0$ **to** $I - 1$ **do**
  sample N trajectories $D_i^N = \{\tau_j\}_{j=1}^N$ from $p(\cdot|\theta_i)$
  if $i >= \bar{i}$ and $i < \bar{i} + k$ **then**
    $B = B \cup (\mathcal{D}_i^N, \theta_i)$
  end if
  if $(i >= \bar{i} + k)$ **then**
    compute π_μ according to Eq. (6.14) and apply IS on both $\mathcal{D}_i^N$ and $\mathcal{D}_{i-k}^N$
    $m_i = max\left(0, m_{i-1} + Te(\mathcal{D}_{\mu/\theta_i}^N, \mathcal{D}_{\mu/\theta_{i-k}}^N, \alpha)\right)$, see Eq. (6.25)
    if $i - \bar{i} == 2k - 1$ **then**
      $\bar{i} += 2k$
      $B = \{\phi\}$
    end if
    if $m_i > K$ **then**
      Change detected
      Reset configuration for the next detection
      Resetting Adam
    end if
  end if
  $\theta_i = \theta_i + \eta\widehat{\nabla}_N J(\theta)$
end for
return π_{θ^*}

made of a 2-dimensional vector composed of the horizontal position, x, and the horizontal velocity, $\dot{x}$, of the car. The reward function is $\mathcal{R}(s, a) = -1 + height$, where $height$ is the car's vertical offset. The G(PO)MDP parametrization is shown in Table 6.3. Notice that this configuration is shared by the baseline (i.e., vanilla G(PO)MDP) and by our proposed algorithm (PG-NSD-RL). Moreover, while our algorithm is at regime, Adam's learning rate is fixed to 10^{-5} in order to prevent the policies from different iteration to be too far from each others, whereas, when our algorithm detects a change, Adam is reset to the initial conditions stated in Table 6.3.

The experiments have been organized as follows. Each run comprises at most $I = 500$ learning iterations for *Pendulum-v0* and $I = 300$ for *Mountain Car*. The learning process begins at iteration $i = 0$. The algorithm NSD-RL is activated at iteration $\bar{i}$. A change in the state transition function $\mathcal{P}$ is introduced at iteration $i^* = \bar{i} + 50$. Let $\hat{i}$ be the iteration at which NSD-RL detects a change, we define a false positive detection when $\hat{i} < i^*$ and a false negative detection when $\hat{i} > I - 1$. A correct detection is considered when $i^* \leq \hat{i} \leq I - 1$ and, in this case, we compute the detection delay as $\hat{i} - i^*$.

Let $s = [\cos\varphi, \sin\varphi, \dot{\varphi}]$ be the vector representing the state in the *pendulum-v0* task, to



(a) *PG-NSD-RL vs G(PO)MDP on Pendulum-v0 with* $c = 20$ *and* $e = 0.2$.

(b) *PG-NSD-RL vs G(PO)MDP on Pendulum-v0 with* $c = 10$ *and* $e = 0.2$.

(c) *PG-NSD-RL vs G(PO)MDP on Pendulum-v0 with* $c = 30$ *and* $e = 0.15$.

(d) *PG-NSD-RL vs G(PO)MDP on Pendulum-v0 with* $c = 20$ *and* $e = 0.2$.

(e) *PG-NSD-RL vs G(PO)MDP on Mountain Car with* $c = 15$ *and* $e = 0.09$.

Figure 6.2: *Comparison of on-line performance over the iterations of the optimization algorithm, with 90% t-student confidence intervals. The first vertical line (dashed) highlight the injection point, whereas the second vertical line (dotted-dashed) highlight the end of the transient part of the anomaly.*

model non-stationarity we have considered an additive clamped-ramp perturbation affect-

Table 6.3: *G(PO)MDP experimental configuration.*

Parameter	*Pendulum-v0*	*Mountain Car*
Neural Network hidden weights	(32,32)	(32,32)
Neural Network activation function	$\tanh$	$\tanh$
Batch size N	100	100
Task horizon	200	500
Discount factor γ	0.99	0.99
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Adam α	0.005	0.005

ing $\dot{\varphi}$ after i^* defined as follows:

$$s = \begin{cases} [\cos\varphi, \sin\varphi, \dot{\varphi}] & \text{if } i < i^* \\ [\cos\varphi, \sin\varphi, \dot{\varphi} + \nu] & \text{otherwise ,} \end{cases} \tag{6.28}$$

where

$$\nu = \begin{cases} e(i - i^*), \text{ if } i - i^* <= c \\ e \cdot c \text{ otherwise,} \end{cases} \tag{6.29}$$

being e the speed of the anomaly and c the duration of its transient component. In particular, we considered three different configurations of ν, i.e., $(e = 0.2, c = 20)$, $(e = 0.2, c = 10)$ and $(e = 0.15, c = 30)$. Moreover, we have considered two different onset points of the anomaly $i^* = 200$ and $i^* = 300$. In the context of *Mountain Car*, the perturbation is still an additive clamped-ramp similarly to what defined above, but it affects $\dot{x}$. For this task we have considered one configuration of ν, i.e., $(e = 0.09, c = 15)$. Furthermore, for this experiment the onset point of the anomaly is $i^* = 100$. The parameter K of NSD-RL has been set to 3, while $\alpha = 0.05$ and $k = 10$ in all the experiments. For all the configurations, we considered 50 runs with different seeds and average results are shown in Figure 6.2.

Two main comments arise. In Figures 6.2a, 6.2c, 6.2d and 6.2e, we can see how PG-NSD-RL shows a better behavior in terms of average undiscounted return after i^*. This is due to a prompt detection combined with the adaptation guaranteeing a higher plasticity of PG-NSD-RL. Interestingly, in Figure 6.2b we show that PG-NSD-RL and G(PO)MDP have similar performances. This is due to the the fact that the anomaly does not induce a relevant change on the environment. Furthermore, in Figures 6.2a and 6.2d, we can see how changing the onset point of the anomaly has a huge impact on performance. Indeed, in the first case the neural network parametrizing the policy reaches a worse configuration of the weights w.r.t. the second case due to a greater number of regime updates, which in some sense could be thought as overfitting the environmental noise in the task.

Table 6.4: *NSD-RL performance in terms of False Positive Rate (FPR), False Negative Rate (FNR) and Detection Delay (DD) in different scenarios.*

ν	i^*	Task	FPR	FNR	DD	DD std.
$e = 0.2, c = 20$	300	*Pendulum-v0*	0	0	4.26	0.955
$e = 0.15, c = 30$	300	*Pendulum-v0*	0	0	4.76	1.141
$e = 0.2, c = 20$	200	*Pendulum-v0*	0	0	3.74	0.743
$e = 0.09, c = 15$	100	*Mountain Car*	0	0	4.66	0.839

In Table 6.4, we show some indicators assessing the quality of the detection phase of the proposed change detection algorithm. As we can see, FPRs are equal for all the configurations. This is reasonable since false positive detections do not depend on the type of change. Moreover, we can see how the detection delay increases from $(e = 0.2, c = 20, i^* = 300)$ to $(e = 0.15, c = 30, i^* = 300)$. This is due to the fact that e is smaller in the second configuaration (i.e., $e = 0.15, c = 30$) inducing a more gradual drift which is, of course, more subtle to be detected. Notice that in Table 6.4 we have not reported the results relative to the configuration $(e = 0.2, c = 10, i^* = 300)$ since they are equivalent to those of $(e = 0.2, c = 20, i^* = 300)$. This is reasonable since we have only decreased the duration of the transient part and the maximum detection delay in the configuration $(e = 0.2, c = 20, i^* = 300)$ is 7. Finally, as expected, the two configurations, $(e = 0.2, c = 20, i^* = 200)$ and $(e = 0.2, c = 20, i^* = 300)$, have very similar average and standard deviation for the detection delay. In the last row of Table 6.4 we show how our NSD-RL algorithm is able to properly detect non-stationarities in another task.

6.6 Conclusions

The aim of this *Chapter* was to introduce a change-detection mechanism to detect changes in a RL problem and integrate it into a RL algorithm to deal with non-stationary tasks. The proposed change-detection mechanism relies on the joint use of a statistical hypothesis test (aiming at comparing the expected value of the reward at two different iterations) and a CUSUM-based sequential mechanism to detect changes in the MDP. Whereas the adaptation phase relies on resetting the history associated to the moments of the Adam optimizer in order to increase the plasticity of the algorithm. The proposed solution is theoretically-grounded and has been successfully tested in two well-known RL tasks showing performance improvements over G(PO)MDP.

Potential future directions for this work may encompass the design of a transfer learning algorithm to be included into the adaptation phase of the current proposed solution, and, also, extending the detection mechanism with a diagnostic part able to identify and characterize the type, temporal evolution, and magnitude of the change (representing valuable information that could be leveraged by the adaptation phase).

CHAPTER 7

Time-Variant Variational Transfer for Value Functions

In addition to the stationarity assumption, RL algorithms require a huge amount of experience to achieve effective results [Vinyals et al., 2019, Silver et al., 2018, OpenAI et al., 2019], hence, in most cases, it is impractical to directly apply an RL algorithm onto a real system because the experience collection would be incredibly slow. This translates into the need for sample efficient RL algorithms, which could be built, among all other alternatives, through TL [Taylor and Stone, 2009, Lazaric, 2012]. In a nutshell, TL enables an RL algorithm to reuse knowledge coming from a set of already solved tasks in order to speed up the learning phase on related new ones.

Depending on what kind of knowledge representation is being transferred, we have different TL algorithms in the related literature.[1] Therefore, in order to perform the transfer, we may have algorithms leveraging policies or options [Fernández and Veloso, 2006, Konidaris and Barto, 2007], samples [Taylor et al., 2008, Lazaric et al., 2008, Tirinzoni et al., 2018b, 2019], features [Barreto et al., 2017, Lehnert and Littman, 2018], value-functions [Taylor et al., 2007, Tirinzoni et al., 2018a] or parameters [Killian et al., 2017, Nagabandi et al., 2018, Du and Narasimhan, 2019]. In the classical TL setting, the source

[1] Observe that the TL literature discussed in this *Chapter* differs from that one reviewed in Section 4.1 because of the differences we have between RL and SL.

and target tasks usually come from the same distribution, hence the Bayesian framework particularly fits because we can iteratively refine the prior knowledge coming from the source tasks as more evidence from the target is collected. Following this rationale, in Wilson et al. [2007], under the assumption that the tasks share similarities in their MDP representation, a hierarchical Bayesian solution is proposed, whose main drawback lies in the need to solve an auxiliary MDP in order to perform actions on the task currently faced. Another methodology, along this line of research, has been developed in Lazaric and Ghavamzadeh [2010], which still leverages hierarchical Bayesian models, but this time assuming the tasks share commonalities through their value functions. Furthermore, in Doshi-Velez and Konidaris [2016], a Bayesian framework able to adapt optimal policies to variations of the task dynamics is developed. They use a latent variable, which, together with the state-action couple, entirely describes the system dynamics. The uncertainty over the latent variable is modeled independently of the uncertainty over the state. This limitation is overcome in the extension to their framework proposed in Killian et al. [2017]. In Perez et al. [2020] another extension to Doshi-Velez and Konidaris [2016] is proposed, which accounts for multiple variation factors that potentially also come from the reward function. A more general and efficient approach is instead developed in Tirinzoni et al. [2018a], which iteratively refines the distribution over optimal value functions by means of a variational procedure as more experience from the target task is collected.

Leveraging the available historical knowledge when it exposes an intrinsic time-variant structure is never considered in the related TL literature and it is the objective of this *Chapter*. Therefore, inspired by the work of Tirinzoni et al. [2018a], we will present a TL algorithm for RL able to model time variations in the distribution inherent to the task generating process. In addition, we will provide a theoretical comparison between our solution and the time-invariant approach of Tirinzoni et al. [2018a] promising a performance improvement in our favor. Finally, we will provide an experimental comparison of the two approaches in three different RL environments with three distinct temporal dynamics and in a real-world scenario represented by a water reservoir system.

All the work herein presented is taken from Canonaco et al. [2021d], where I am the main researcher.

7.1 Preliminaries

In this section, we extend the setting introduced in Tirinzoni et al. [2018a] by adding a time-variant distribution over the tasks. We introduce basic RL concepts and some notation in Section 7.1.1, and we describe the variational approach to transfer in Section 7.1.2.

7.1.1 Reinforcement Learning Background

Let us consider a time-variant distribution $\mathcal{D}_t$ over tasks. We model each task $\mathcal{M}_t$ coming from $\mathcal{D}_t$ as a discounted MDP [Puterman, 2014], which is defined as a tuple $\mathcal{M}_t = \{\mathcal{S}, \mathcal{A}, \mathcal{P}_t, \mathcal{R}_t, p_0, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state space and the action space, respectively, $\mathcal{P}_t$ is the Markovian transition function with $\mathcal{P}_t(s'|s,a)$ being the transition density from state s to state s' given that the action a is executed on the environment. The reward function is defined as $\mathcal{R}_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, assumed to be uniformly bounded by a constant $R_{max} > 0$. Finally, p_0 and $\gamma \in [0, 1)$ are the initial state distribution and the discount factor, respectively. Therefore, for each task t our goal is to find a deterministic policy, $\pi_t : \mathcal{S} \to \mathcal{A}$, maximizing the long-term return over a possibly infinite horizon. In other words, this means being able to get $\pi_t^* \in \arg\max_\pi J_t(\pi)$, where $J_t(\pi) = \mathbb{E}_{\mathcal{M}_t,\pi}[\sum_{h=0}^{\infty} \gamma^h \mathcal{R}_t(s_h, a_h)]$. The optimal policy π_t^* is a greedy policy w.r.t. the optimal value function, i.e., $\pi_t^*(s) = \arg\max_a Q_t^*(s,a)$ for all s, where $Q_t^*(s,a)$ is defined as the expected return obtained by taking action a in state s and then following the optimal policy afterward. From now on, for the sake of readability, we will drop t whenever this does not imply ambiguity.

In this context, we focus on a set of parametrized value functions, $Q = \{Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R} | \theta \in \mathbb{R}^p\}$, also called Q-functions. We assume that each $Q_\theta \in Q$ is uniformly bounded by $\frac{R_{max}}{1-\gamma}$. An optimal Q-function is also the fixed point of the optimal Bellman operator [Puterman, 2014], which is defined as follows: $TQ_\theta(s,a) = \mathcal{R}(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}}[\max_{a'} Q_\theta(s', a')]$. Therefore, a measure of optimality for a value function during learning is its Bellman error, defined as $B_\theta = TQ_\theta - Q_\theta$. Of course, if $B_\theta(s,a) = 0 \; \forall (s,a) \in \mathcal{S} \times \mathcal{A}$, then Q_θ is optimal, which implies that minimizing the squared Bellman error, $||B_\theta||_\nu^2$, is a good objective for learning (where ν is the distribution over $\mathcal{S} \times \mathcal{A}$, assumed to exist). In practice, the Bellman error is not used, since it requires two independent samples of the next state s' for each couple (s,a) [Maillard et al., 2010, Sutton and Barto, 2011]. For this reason, usually, the Bellman error is replaced by the Temporal Difference (TD) error $b(\theta)$, which corresponds to an approximation of the former using one sample $\langle s_h, a_h, r_h, s_{h+1} \rangle$, so $b_h(\theta) = r_h + \gamma \max_{a'} Q_\theta(s_{h+1}, a') - Q_\theta(s_h, a_h)$. Therefore, given a set $D = \langle s_h, a_h, r_h, s_{h+1} \rangle_{h=1}^N$, the squared TD error on D is $||B_\theta||_D^2 = \frac{1}{N} \sum_{h=1}^N b_h(\theta)^2$ (with a little abuse of notation w.r.t. the definition of the Bellman error).

7.1.2 Variational Transfer of Value Functions

In the context described above, an optimal solution to an RL problem is a greedy policy w.r.t. an optimal value function that is parameterized by a vector of weights θ. Therefore, we can safely consider a distribution over optimal weights $p(\theta)$ instead of the distribution $\mathcal{D}$ over tasks since the latter induces a distribution over optimal Q-functions [Tirinzoni et al., 2018a]. Now, given a prior on the weights $p(\theta)$ and a data set $D = \langle s_h, a_h, r_h, s_{h+1} \rangle_{h=1}^N$,

the optimal Gibbs posterior that minimizes an oracle upper bound on the expected loss is defined as [Catoni, 2007]:

$$q(\theta) = \frac{e^{-\Psi||B_\theta||_D^2} p(\theta)}{\int e^{-\Psi||B_{\theta'}||_D^2} p(\theta') d\theta'}, \tag{7.1}$$

where $\Psi > 0$, which will be set to $\psi^{-1}N$, for some constant $\psi > 0$ as in Tirinzoni et al. [2018a]. It is worth noting that q becomes a Bayesian posterior every time $e^{-\Psi||B_\theta||_D^2}$ can be interpreted as the likelihood of D. Since the integral at the denominator of Equation (7.1) is intractable, a variational approximation through a parametrized family of posteriors q_ξ, such that $\xi \in \Xi$, is proposed. In this way, it is sufficient to find ξ^* such that q_{ξ^*} minimizes the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951] w.r.t. the Gibbs posterior q, which is equivalent to minimizing the (negative) Evidence Lower BOund (ELBO) defined as [Blei et al., 2017]:

$$\min_{\xi \in \Xi} \mathcal{L}(\xi) = \min_{\xi \in \Xi} \left\{ \mathbb{E}_{\theta \sim q_\xi} \left[||B_\theta||_D^2 \right] + \frac{\psi}{N} D_{KL}(q_\xi(\theta) || p(\theta)) \right\}. \tag{7.2}$$

Therefore, the idea behind the variational transfer of value functions (as shown in Algorithm 7) is to alternate a sampling from the posterior on the optimal value function with the optimization of the posterior via $\nabla_\xi \mathcal{L}(\xi)$, assuming to have already solved a finite number of source tasks $\mathcal{M}_1 \dots \mathcal{M}_n$, which, in turn, implies having the set of their approximate solutions $\Theta_s = \{\theta_1, \dots, \theta_n\}$.[2] The weight resampling in line 8 can be interpreted as a guess on the task that we need to solve based on the current belief. After sampling, the algorithm acts on the RL problem as if such guess was correct (line 9) and then will adjust the belief based on the new experience through the optimization of the variational parameters ξ (lines 12 and 13). Notice that, as long as $\nabla_\xi \mathcal{L}(\xi)$ can be efficiently computed, any approximator for the Q-functions and any prior/posterior distributions can be used. To this end, since the $\max$ operator in the temporal difference error of Equation (7.2) is not differentiable, the *mellowmax* is used instead, which is differentiable and was proven to converge to the same fixed point of the optimal Bellman operator in Tirinzoni et al. [2018a]. From now on, we will denote the mellow Bellman error by $\tilde{B}_\theta$.

7.2 Time-Variant Kernel Density Estimation for Variational Transfer

In the context of this work, we will model the evolution of time over a discrete grid of asymptotically dense time instants. Let $\{\theta_{ij}\}_{j=1}^{M_i}$ be a set of independent solutions for the i^{th} family of tasks, observed at time $t_i = \frac{i}{n}$, $1 \leq i \leq n$, with $\theta_{ij} \in \mathbb{R}^p$ and $\theta_{ij} \sim P(\cdot, t_i)$. Notice that, for the sake of generality, at time t_i, we allow to tackle M_i times the i^{th}

[2] Notice that, in the context of this work, $\mathcal{M}_1 \dots \mathcal{M}_n$ are samples coming from a time-variant distribution, hence independent but not identically distributed.

Algorithm 7 Variational Transfer

Input: Target task $\mathcal{M}_t$, source weights Θ_s
Estimate prior $p(\theta)$ from Θ_s
Initialize parameters: $\xi \leftarrow \arg\min_{\xi \in \Xi} D_{KL}(q_\xi || p)$
Initialize data set $D = \emptyset$
while $True$ **do**
  Sample initial state $s_0 \sim p_0$
  while s_h is not terminal **do**
    Sample weights $\theta \sim q_\xi(\theta)$
    Take action $a_h = \arg\max_a Q_\theta(s_h, a)$
    $s_{h+1} \sim \mathcal{P}_t(\cdot|s_h, a_h)$, $r_{h+1} = \mathcal{R}_t(s_h, a_h)$
    $D \leftarrow D \cup \langle s_h, a_h, r_{h+1}, s_{h+1} \rangle$
    Estimate $\nabla_\xi \mathcal{L}(\xi)$ using $D' \subseteq D$
    Update ξ with $\nabla_\xi \mathcal{L}(\xi)$ using any optimizer (e.g., Kingma and Ba [2014])
  end while
end while

family of tasks represented by the distribution $P(\cdot, t_i)$ with associated probability density function $p(\theta, t_i)$. Furthemore, let M_i be a discrete random variable for each i. Finally, let us introduce a Time-Variant Kernel Density Estimator defined as follows:

$$\hat{p}(\theta, t) = \frac{1}{a_0(-\rho)\bar{N}\lambda|H|^{\frac{1}{2}}} \sum_{i=1}^{n} K_T\left(\frac{t - t_i}{\lambda}\right) \sum_{j=1}^{M_i} K_S(H^{-\frac{1}{2}}(\theta - \theta_{ij})), \tag{7.3}$$

which is based on Hall et al. [2006] and will be used as a prior to model a time-variant distribution on the solved tasks. The factor $a_0(-\rho) = \int_{-\rho}^{1} K_T(t)dt$ is used to recover consistency at the boundaries [Jones, 1993], therefore also in $t = 1$, which represents the time instant that will be used in Algorithm 7 to produce a prior for the current family of tasks. K_T is the temporal kernel, whereas K_S is the multivariate non-negative spatial kernel. Furthermore, H is the spatial kernel bandwidth matrix, $\lambda \in [0, 1]$ is the temporal kernel bandwidth, and $\bar{N} = \sum_{i=1}^{n} M_i$.

Given the following assumptions, also stated in Hall et al. [2006]:

Assumption 7.2.1 (Task independence)**.** *For* $1 \leq i \neq i' \leq n$,$1 \leq j \leq M_i$, *and* $1 \leq j' \leq M_{i'}$, θ_{ij} *and* $\theta_{i'j'}$ *are independent;*

Assumption 7.2.2 (Differentiable density function)**.** $p(\theta, t) : \mathbb{R}^p \times (0, 1] \to \mathbb{R}$ *is twice differentiable for every* t, θ*;*

Assumption 7.2.3 (Bounded derivatives)**.** $p(\theta, t) : \mathbb{R}^p \times (0, 1] \to \mathbb{R}$ *has two bounded derivatives;*

Assumption 7.2.4 (On the spatial kernel)**.** *Let $\alpha = (\alpha_1, \ldots, \alpha_p)$ be a multi-index, with $\alpha_i \geq 0$ for $i = 1, \ldots, p$, $\theta^\alpha = \prod_{i=1}^p \theta_i^{\alpha_i}$ for each $\theta \in \mathbb{R}^p$, and N_0 is an index set where all p components of each member are either 0 or even integers.*

$$\int_{\mathbb{R}^p} K_S(\theta)d\theta = 1, \lim_{||\theta||\to\infty} ||\theta||^p K_S(\theta) = 0,$$
$$\int_{\mathbb{R}^p} \theta^\alpha K_S(\theta)d\theta = \mu_\alpha \leq \infty, \alpha \in N_0,$$
$$\int_{\mathbb{R}^p} \theta^\alpha K_S(\theta)d\theta = 0, \alpha \notin N_0;$$

Assumption 7.2.5 (On the temporal kernel)**.**

$$\int_{-c}^{c} K_T(t)dt = 1, \int_{-c}^{c} tK_T(t)dt = 0,$$
$$\int_{-c}^{c} t^2 K_T(t)dt = \sigma_T \leq \infty;$$

the following theorem holds:

Theorem 7.2.6 (Uniform consistency of the density estimator)**.** *Assume 7.2.1 - 7.2.5. Moreover, assume that K_S is spherically symmetric, with a bounded, Hölder-continuous derivative, that K_T is a compactly supported kernel on a subset of $\mathbb{R}$, that all the M_is are independent and identically distributed random variables with mean $m > 0$ and all moments finite, independent of the θ_{ij}s. Take H and λ such that $|H|^{\frac{1}{2}}(n) \to 0$, $\lambda(n) \to 0$ and $n^{1-\epsilon}|H|^{\frac{1}{2}}\lambda \to \infty$ for some $\epsilon > 0$ as $n \to \infty$, then*

$$\hat{p}(\theta, t) = p(\theta, t) + O\big[(\bar{N}|H|^{\frac{1}{2}}\lambda)^{-\frac{1}{2}}(\log n)^{\frac{1}{2}} + tr(H) + \lambda\big]$$

uniformly in $(\theta, t) \in \mathcal{K} \times \mathcal{I}$, with probability 1, where $\mathcal{K}$ is a compact subset of $\mathbb{R}^p$ and $\mathcal{I}$ is a compact subset of $(0, 1]$.

A proof of the above theorem is shown in Appendix B.1 and leverages the same approach as in Hall et al. [2006] being a weaker version, in terms of convergence rate, of their Theorem 1. This weakening was necessary to obtain an upper bound in closed-form expression of the KL-divergence between the prior and the posterior in Equation (7.2). [3] Indeed, if we choose $q_\xi(\theta) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\theta|\mu_k, \Sigma_k)$, with variational parameter $\xi = (\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K)$, and we choose K_S as a Gaussian kernel, then for a fixed time instant t our prior is a mixture of Gaussians with non-uniform weights. Therefore, through

[3] This upper bound cannot be obtained by directly using the estimator proposed in Hall et al. [2006] because of the negative weights associated with the spatial kernel.

the upper bound on the KL-divergence shown in Appendix B.2 which leverages Hershey and Olsen [2007], we have that the ELBO upper bounds the KL-divergence between the approximate and the exact posterior. Since the covariance matrices of the posterior must be positive definite, we will learn the factor L of their Cholesky decomposition as in Tirinzoni et al. [2018a].

Let us comment on the previous assumptions and their limiting effects on applications. For what concerns Assumptions 7.2.4 and 7.2.5, they do not pose any limit, since, as we know from kernel density estimation theory [Wand and Jones, 1994], the kernel type is not relevant for a good estimate of the density. Assumptions 7.2.2 and 7.2.3, instead, are necessary to have some regularity allowing the time-variant distribution to be learned (without those assumptions the kernel density estimator would not be consistent). The range of time-variant distributions where our approach will be theoretically effective is reduced due to Assumptions 7.2.2 and 7.2.3, but remains still relevant from an application perspective since it allows to solve real problems such as controlling the lake Como water system as shown in Section 7.5.6.

7.3 Finite-Sample Analysis

In order to provide a finite sample analysis of Algorithm 7 based on the prior of Section 7.2, we extend Theorem 2 of Tirinzoni et al. [2018a] to deal with time-variant contexts, enabling also a theoretical comparison between the two respective versions of Algorithm 7. Therefore, considering the family of linearly parametrized value functions, $Q_\theta(s,a) = \theta^T \phi(s,a)$, having bounded weights $||\theta||_2 \leq \theta_{max}$ and uniformly bounded features $||\phi(s,a)||_2 \leq \phi_{max}$, and assuming that only finite data are available, we can bound the expected mellow Bellman error under the variational distribution minimizing Equation (7.2) for any fixed target task $\mathcal{M}_t$ through the following theorem.

Theorem 7.3.1 (Bound on the expected mellow Bellman error)**.** *Let $\hat{\xi}$ be the variational parameter minimizing Equation* (7.2) *on a data set D of N i.i.d. samples distributed according to $\mathcal{M}_t$ and ν. Moreover, let $\theta^* = \arg\inf_\theta ||\tilde{B}_\theta||_\nu^2$ and define $v(\theta^*) = \mathbb{E}_{\mathcal{N}(\theta^*, \frac{1}{N}I)}[v(\theta)]$, with $v(\theta) = \mathbb{E}_\nu[\mathbb{V}ar_{\mathcal{P}_t}[\tilde{b}(\theta)]]$, where $\tilde{b}(\theta) = r + \gamma mellow\text{-}max_{a'} Q_\theta(s',a') - Q_\theta(s,a)$. Then, there exist constants c_1, c_2, c_3 such that with probability at least $1-\delta$ over the choice of D:*

$$\mathbb{E}_{q_{\hat{\xi}}}\left[\left|\left|\tilde{B}_\theta\right|\right|_\nu^2\right] \leq 2\left|\left|\tilde{B}_{\theta^*}\right|\right|_\nu^2 + v(\theta^*) + c_1\sqrt{\frac{\log\frac{2}{\delta}}{N}} + \frac{\boldsymbol{c_2} + \psi p \log N + \psi \boldsymbol{\varphi(\Theta_s)}}{N} + \frac{c_3}{N^2},$$

where

$$\varphi(\Theta_s) = \frac{1}{\sigma^2}\sum_{j:\theta_j \in \Theta_s} \zeta(j) \text{ with} \tag{7.4}$$

$$\zeta(j) = \frac{c_j^{\hat{p}} e^{-\beta||\theta^* - \theta_j||}}{\sum_{j':\theta_{j'} \in \Theta_s} c_{j'}^{\hat{p}} e^{-\beta||\theta^* - \theta_{j'}||}} ||\theta^* - \theta_j||,$$

assuming the matrix H of Equation (7.3) *to be an isotropic covariance matrix with variance σ^2, $\beta = \frac{1}{2\sigma^2}$ and $c_j^{\hat{p}}$ the weight assigned to the j^{th} prior component. Furthermore, we are assuming $M_i = 1$ for each i in our estimator.*

The above theorem shows the difference between the plain mixture version of Algorithm 7 [Tirinzoni et al., 2018a] and our solution which lies in the constant c_2 and in the term $\varphi(\Theta_s)$. Looking at $\varphi(\Theta_s)$, we can shed some light on the different theoretical properties of the two versions. More specifically, in the plain mixture version, the factor $c_j^{\hat{p}}$ does not appear, which implies uniform importance of the source solutions Θ_s w.r.t. the target task. On the other hand, in our version of the algorithm, we can give different importance to each source solution through $c_j^{\hat{p}}$. Increasing the weight of sources similar to the target will reduce $\varphi(\Theta_s)$ implying a faster convergence to the optimal solution. In our time-variant scenario, this weight will be greater on more recent solutions than older ones, potentially enabling a reduction of the term $\varphi(\Theta_s)$ w.r.t. the time-invariant version (shown experimentally in Section 7.5). For what concern c_2, the main difference is due to a different expression of the KL-divergence upper bound, and the usage of non-uniform weights. A proof for the above theorem together with the definition of all the constants is provided in Appendix B.3.

7.4 Related Works

The work presented in this *Chapter* is inspired by Tirinzoni et al. [2018a]. Differently from them, we leverage a time-variant structure underlying the task generating process, which lets us cope with time-variant scenarios. A theoretical comparison between the two solutions is available in Section 7.3 through Theorem 7.3.1, whereas the experimental comparison is in Section 7.5. Furthermore, our work relates both to Wilson et al. [2007], which deals with finite MDPs, and to Lazaric and Ghavamzadeh [2010], which leverages the commonalities in the value function structure, but, in contrast to our work, they do not account for a time-variant distribution. The work done in Doshi-Velez and Konidaris [2016], Killian et al. [2017], Perez et al. [2020] leverage latent embeddings in order to model variations between tasks, which eventually are solved through a model-based RL algorithm, while we propose a model-free approach.

Another related work is Hall and Willett [2015], in which the authors develop a theoretical low-regret algorithm accounting for potential underlying dynamics. However, they use the online learning framework, whereas we are working in a transfer learning setting. Furthermore, in Du and Narasimhan [2019], videos are used to learn a prior (mainly to model the physical dynamics) which is incorporated into a model-based RL algorithm.

In Yang et al. [2020], a single-episode policy-transfer methodology was developed leveraging variational inference, but for contexts in which the differences in dynamics can be identified in the early steps of an episode. In the context of supervised learning, our work relates also to Minku and Yao [2014], which proposes a transfer learning mechanism in the context of a possibly non-stationary environment through a weighting approach, and Du et al. [2019], which, instead, do transfer in non-stationary environments through ensembles. Finally, in the meta-learning framework, Khodak et al. [2019] is able to consider optimal initializations varying through time, Mendonca et al. [2020] provide robustness to distributional shifts during meta-testing through an experience relabeling mechanism, and Fu et al. [2020] develop a Context-based Meta-RL algorithm which leverages contrastive learning and an information-gain-based exploration strategy showing good performances in out-of-distribution tasks. These last three approaches are meta-learning based, while the approach proposed in this *Chapter* considers a transfer learning setting.

7.5 Experiments

In this section, we compare our time-variant solution for transfer learning with the associated non-time-variant solution of Tirinzoni et al. [2018a] in three different domains with three different temporal dynamics and a real-world scenario.[4] The first three domains were chosen from Tirinzoni et al. [2018a] (adding the temporal dynamics) in order to enable a faithful comparison. The real-world problem consists in controlling a water reservoir system, where the temporal dynamic is due to the climate change across the decades. A detailed description of the used parameters together with the analytical expression of the employed temporal dynamics are provided in Appendix B.4.

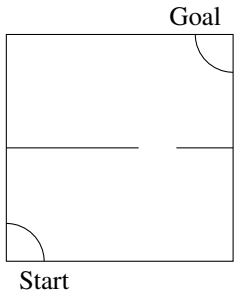
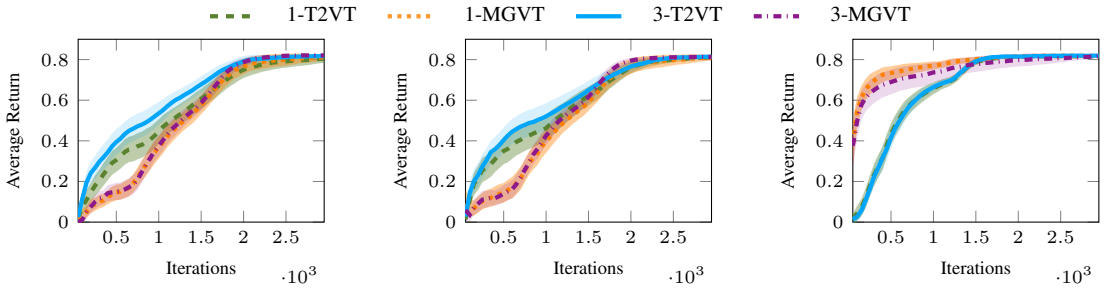


Figure 7.1: *2-Rooms Environment.*

7.5.1 Temporal Dynamics

The distribution over the tasks is usually a given distribution over one or more parameters defining the task itself. Therefore, in order to obtain time variance in such distribution,

[4] Code at https://github.com/AndreaSoprani/T2VT-RL.



(a) *Polynomial dynamic.* **(b)** *Linear dynamic.* **(c)** sin *dynamic.*

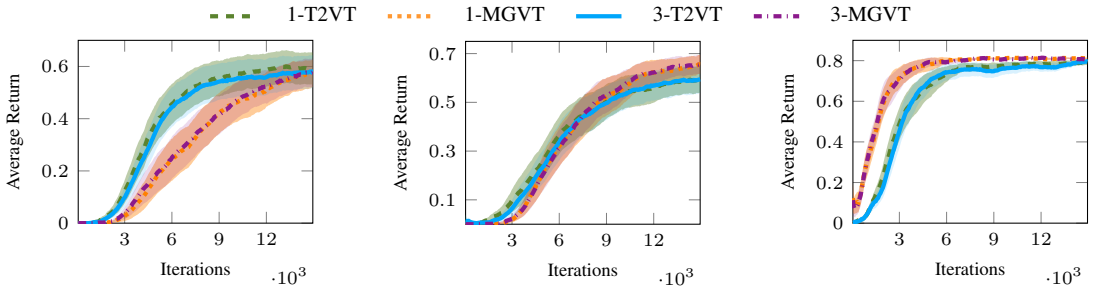
Figure 7.2: *Average return achived by the algorithms with* 95% *confidence intervals computed using* 50 *independent runs in the 2-rooms environment.*

we will change its mean over time according to a certain dynamic. These dynamics are linear, polynomial, and sinusoidal. In the context of these experiments, we will use a time-variant Gaussian distribution, clipping its realizations within the domain of the task-defining parameters (for further details see Appendix B.4). Instead, in the water reservoir system, the temporal dynamic is inherent to the data and, as already mentioned, due to climate change.

7.5.2 Two-Rooms Environment

In this setting, we have an agent navigating two rooms separated by a wall (see Figure 7.1). The agent starts from the bottom-left corner and must reach the opposite one. The only way to reach this goal is to pass through the door whose position is unknown to the agent. The actions available to the agent are *up*, *down*, *left*, and *right*, which let the agent to move in the respective directions by one position, unless he/she hits a wall (in this last case the position remains unchanged). Furthermore, the final position of the agent after a movement action is altered by a Gaussian noise $\mathcal{N}(0, 0.2)$. The state space is modeled through a 10×10 continuous grid. Finally, the reward function is 0 everywhere except in the goal state, where it is 1. The discount factor $\gamma = 0.99$. For this setting, we used linearly parametrized Q-functions with 121 evenly-spaced radial basis features.

We considered source tasks taken at ten different time instants to learn the target, corresponding to the eleventh instant of time. We sampled five tasks from the time-variant distribution for each $i = 1, \dots, 11$. The parameter that defines the task is the door location, hence the time-variant distribution is over that parameter, as we mentioned above. We solve all the source tasks by directly minimizing the TD error, then we exploit the learned solutions to perform the transfer over the target. We compare our time-variant variational transfer algorithm leveraging a c-components posterior (c-T2VT) with the mixture of Gaussian variational transfer using still c-components (c-MGVT) [Tirinzoni et al.,



(a) *Polynomial dynamic.* **(b)** *Linear dynamic.* **(c)** sin *dynamic.*

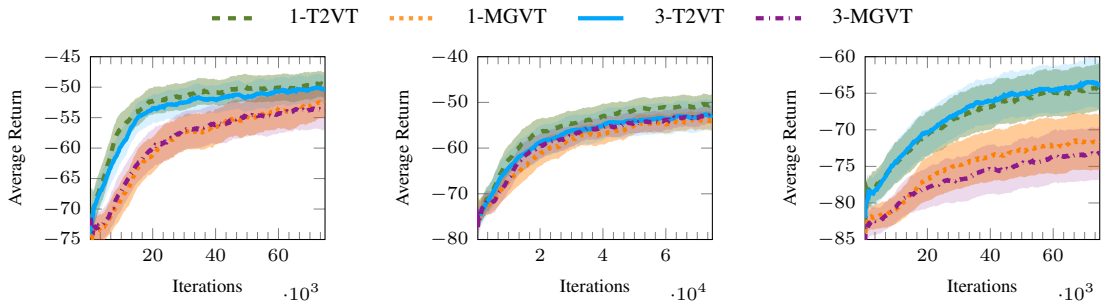
Figure 7.3: *Average return achived by the algorithms with* 95% *confidence intervals computed using* 50 *independent runs in the 3-rooms environment.*

2018a]. More specifically, our time-variant prior will consider the source task solutions as equally spaced samples in the time interval $[0, 1]$, moreover, in order to perform transfer to the eleventh task, we will use the distribution provided by our estimator for $t = 1$. Finally, the temporal kernel will be Epanechnikov [Epanechnikov, 1969, Wand and Jones, 1994] in the context of all the experiments.

The average return over the last 50 learning episodes as a function of the number of training iterations is shown in Figure 7.2, for the time dynamics mentioned in Section 7.5.1. Each learning curve is computed using 50 independent runs, each of which resamples both the source and target tasks, with 95% confidence intervals. For polynomial and linear dynamics, we can see an advantage of our technique in the early learning iterations. The sinusoidal dynamic is designed to disadvantage our technique w.r.t. c-MGVT, indeed, it makes the target task appear twice in the sources. This fact inevitably favors c-MGVT, which will give a higher weight to those source tasks being sampled from the same distribution of the target. Observe that c-MGVT gives uniform weights to all the source tasks, hence increasing the replicas importance within the sources, whereas c-T2VT gives increasing weights the more recent the source solution.

7.5.3 Three-Rooms Environment

This scenario is an extension of the previous one, hence the environmental settings remain the same, the agent has just an additional wall to traverse in order to reach his/her goal. Of course, the position of the door for this additional wall is still unknown to the agent. To increase the complexity of the dynamics, we let the two doors move in opposite directions starting at the two far ends of the room, each door with the same dynamic. In Figure 7.3, we compare c-T2VT with c-MGVT using still 95% confidence intervals. As for the polynomial dynamics, we observe a better performance of c-T2VT w.r.t. c-MGVT, whereas, for the sinusoidal dynamics, we have essentially the same behavior as in the two rooms en-



(a) *Polynomial dynamic.* **(b)** *Linear dynamic.* **(c)** sin *dynamic.*

Figure 7.4: *Average return achived by the algorithms with* 95% *confidence intervals computed using* 50 *independent runs in the Mountain Car environment.*

vironment. Finally, in the linear dynamics, we observe that the difference in performance between the two algorithms is not statistically significant.

7.5.4 Mountain Car

In this section, we consider a classic control problem known as Mountain Car [Sutton and Barto, 2011]. In Mountain Car, the agent is an underpowered car whose goal is to escape a valley. Due to the limitation to its engine, the car has to drive up along the two slopes of the valley in order to gain sufficient momentum to overcome gravity (further details in Appendix B.4.4). In Figure 7.4, we have a comparison between c-T2VT and c-MGVT on the three proposed dynamics. We observe a statistically significant improvement in the polynomial dynamics across the whole learning process for c-T2VT, which also extends to the sinusoidal dynamic case. We would like to highlight the differences between the sinusoidal dynamic in Mountain Car w.r.t. the previous two environments. Here our algorithm is able to perform better due to a bias-variance trade-off in its favor. More specifically, the value functions vary more rapidly in Mountain Car than in the room environments w.r.t. a change in the task-defining parameters. Therefore, our prior estimator has less variance, since it considers only the latest sources, at the cost of a bias increase, because it discards the first task, which has the same parametrization as the target (due to the periodicity of the sin function). c-MGVT considers all the source tasks with the same weight, hence it is able to consider the tasks that have an equivalent parametrization to the target, but are farther behind in the sources' history. This fact decreases the bias at the cost of accepting a greater variance in the prior estimation. In Mountain Car, the trade-off proposed by our algorithm is more advantageous than the one proposed by c-MGVT due to the more rapidly changing behavior of the value functions. As for the linear dynamics, we do not observe a statistically significant difference in performance between the two algorithms, even though the average of 1-T2VT is the best one.

7.5.5 Choosing λ through Maximum-Likelihood

Up to now, we have kept λ and H at given constant values in order to provide a more faithful comparison between c-T2VT and c-MGVT (H was the same in the two algorithms whereas λ was set to 0.3333 leveraging the intuition that the more recent tasks were more important than the older ones). Of course, from the theory of Kernel Density Estimation [Wand and Jones, 1994], we know that appropriately setting these parameters is crucial to get a good estimate of the density. Therefore, an automatic data-driven approach would be desirable. In the context of this work, we propose a maximum likelihood scheme (assuming $M_i = 1 \; \forall \; i$):

$$\arg\max_{\lambda} L_\lambda = \prod_{h=1}^{n} \frac{\hat{p}_{-h}(\theta_h, t_h)}{\hat{p}_{-h}(t_h)}, \text{ where} \tag{7.5}$$

$$\hat{p}_{-h}(\theta_h, t_h) = \frac{1}{a_0(-\rho)(\bar{N}-1)\lambda|H|^{\frac{1}{2}}} \sum_{i \neq h} K_T\left(\frac{t_h - t_i}{\lambda}\right) K_S(H^{-\frac{1}{2}}(\theta_h - \theta_i))$$

$$\hat{p}_{-h}(t_h) = \int \hat{p}_{-h}(\theta_h, t_h) d\theta_h = \frac{1}{a_0(-\rho)(\bar{N}-1)\lambda} \sum_{i \neq h} K_T\left(\frac{t_h - t_i}{\lambda}\right).$$

In Appendix B.4.3, we report the performance achievable with this approach together with a sensitivity analysis w.r.t. the parameter λ for every environment discussed so far. Furthermore, still in Appendix B.4.3, we include some implementation details related to the optimization of the likelihood function in Equation (7.5). Note that, in accordance with what has been done in Tirinzoni et al. [2018a], the spatial bandwidth is set to $10^{-5}I$ which would prevent us from successfully optimizing Equation (7.5) due to numerical issues, hence we set it to I in order to select the best lambda.

7.5.6 Real-World Scenario: Controlling the Lake Como Water System

Lake Como is the third largest lake in Italy thanks to its surface area of $146 \; km^2$ and the fifth deepest in Europe with its maximum depth at 425 meters. It is a lake of glacial origin which has been regulated since 1946 by a human operator to prevent flooding along the lake shores and supply water to the downstream users, which are composed of 4 irrigation districts (total irrigated area of $1400 \; km^2$) and 9 run-of-river power plants (total capacity of $92 \; MW$).

To design the optimal lake operation, we can leverage RL to find an optimal control policy π^* for the water reservoir system of lake Como [Castelletti et al., 2010]. For this setting, the state space includes the day of the year and lake storage volume. The first is encoded as sine and cosine functions of $2\pi\frac{t}{period}$, where t is the day and $period$ is the year's length, which enables accounting for the time-dependency and cyclostationarity of the system, and, consequently, of the operating policy. The second one is governed by the

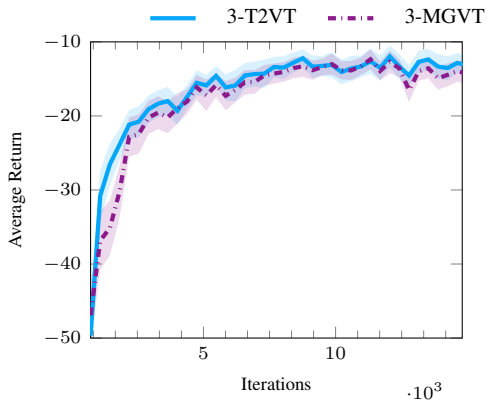


Figure 7.5: *Average return achived by the algorithms with* 95% *confidence intervals computed using* 100 *independent runs on the lake Como environment.*

mass conservation equation $v_{t+1} = v_t + i_{t+1} - \varrho_{t+1}$, where i_{t+1} is the net inflow volume in the time interval $[t, t+1)$ and $\varrho_{t+1} = f_t(v_t, a_t, i_{t+1})$ is the actual release accomplished by the system. The release function $f_t(\cdot)$ accounts for physical and normative constraints on the storage and release [Soncini-Sessa et al., 2007]. Observe that, the actual release depends on the previous storage volume, the policy's decision (corresponding to the amount of water the agent would like the system to release), and the inflow i_{t+1}, which is influencing the system throughout the whole time period $[t, t+1)$. The reward function is composed of three main costs related to water demand, flooding events, and actions feasibility. It is noteworthy to point out the fact that, being the net inflow volume composed of historical data, this setting constitutes an environment incredibly close to the real-world system. Further details are in Appendix B.4.4.

In order to successfully apply RL onto the lake Como water system, we need to carefully take into account the time-variant nature of the net inflow volume, which has changed much since the mid $40s$ due to the climate change our planet is currently undergoing [Giuliani et al., 2016]. Furthermore, if we were to leverage an RL algorithm to control the water reservoir system, TL would be a must to reduce the amount of data needed to reach an optimal behavior and to mitigate the usage of sub-optimal policies onto the system. Since we do not have another time-variant transfer algorithm for RL in the literature, we will again compare T2VT with MGVT to analyze the benefit of accounting for time variance.

Historical data span from 1946 to 2006 and will be split into 12 years chunks, each one representing a task. Hence, the sources will consist of the tasks $[1946, 1957]$, $[1958, 1969]$, $[1970, 1981]$, $[1982, 1993]$, whereas the target is represented by the task $[1994, 2006]$. Results are reported in Figure 7.5, where we compare 3-T2VT coupled with the maximum-likelihood approach of Section 7.5.5 against 3-MGVT. As we can see, there is a beneficial effect on the optimization process by accounting for time-variance in the source

solutions. Indeed, our algorithm performs better than 3-MGVT especially in the early iterations where the performance difference is statistically significant.

7.6 Discussion and Conclusions

In this *Chapter*, we presented a time-variant approach for transferring value functions through a variational scheme, which, together with the work presented in *Chapter* 6, concludes the presentation of our second contribution (see Section 1.1).

In order to deal with a time-variant distribution of the tasks, we have devised a suitable estimator for the prior to be used in the variational scheme providing its uniform consistency over a compact subset of $\mathbb{R}^p \times (0, 1]$. We have, then, provided a finite sample analysis on the performance of the variational transfer algorithm based on our estimator, enabling a theoretical comparison with the time-invariant version of Tirinzoni et al. [2018a]. Finally, we have experimentally proved our algorithm abilities to deal with time-variant distributions even in a real-world scenario represented by the lake Como water system.

Notice that discriminating the source tasks w.r.t. time is an additional step that brings transfer learning approaches and learning in non-stationary environments a bit closer together [Minku, 2019]. It is also important to highlight the fact that, instead of considering time, we could switch to any other variable (e.g., the task-defining parameter) as long as it is available together with each source solution and we can properly remap it into $(0, 1]$. This could enable us to leverage completely different structures in order to perform transfer to the target task. Finally, we would also like to highlight the possibility of using this time-variant transfer paradigm in lifelong learning scenarios [Chen and Liu, 2018] as a potential future direction.

Part III

Non-Stationary Federated Learning and Pervasive Systems

CHAPTER 8

Introduction

In the previous Part of this dissertation, we have weakened the stationarity assumption in some contexts of RL. Specifically, in *Chapter* 6, we have dealt with non-stationarities associated with the current task the RL agent is trying to solve, and, in *Chapter* 7, we considered a TL setting with a non-stationary distribution underlying the task generating process. Now, as mentioned in Section 1.1, we will try to relax the stationarity assumption in the context of FL, which is a promising research area in the ML field. Techniques and solutions belonging to this area operate in distributed scenarios, comprising a server and pervasively distributed clients, aiming at learning a single central model without sending (possibly sensitive) data from the clients to the server. Such a framework allows mitigating the privacy concerns that are nowadays perceived as relevant in distributed ML solutions leveraging data belonging to different users or companies. In the FL field many state-of-the-art solutions are available. Unfortunately, all these solutions assume (implicitly or explicitly) that the data generating process is stationary (hence not changing its statistical behavior over time); an assumption that rarely holds in real-world conditions where concept drifts occur due to, e.g., seasonality or periodicity effects, faults in sensors or actuators, or changes in the users' behavior. Therefore, in *Chapter* 9, we will present an FL algorithm, called Adaptive-FedAVG, able to operate with data generating processes affected by concept drifts. Following a passive approach, Adaptive-FedAVG, which we in-

troduced in Canonaco et al. [2021a], is able to promptly react to concept drifts by adapting the learning rate in order to increase the plasticity of the learning phase itself. This allows our proposed solution to compensate the loss in performance induced by the concept drift resulting in faster-to-converge learning curves.

However, when dealing with pervasive systems, in addition to privacy constraints and non-stationarity, it may happen that we need to deal with computational or memory constraints on the edge devices. This last characteristic, being very stringent, must be taken into account while designing ML-based solutions for such contexts and it is the objective of *Chapter* 10, where we tackle a very specific application in the pervasive systems' world: that one of birdsong detection.

Understanding the distribution of bird species and populations and learning how birds behave and communicate are of great importance in wildlife biology, animal ecology, conservation of ecosystems, and assessing the effects of climate change and urbanization. The temporal and spatial limitations of human observation have motivated significant efforts to develop technology for bird song and vocalization detection and classification. While solutions based on signal processing and machine learning are extant, they are limited in various combinations of speed, computational complexity, and memory use, as well as in detection/classification capability in real-world conditions. Therefore, in *Chapter* 10, we introduce ToucaNet, a deep neural network for birdsong detection based on TL: this enables us to speed up training and shows improved detection accuracy. ToucaNet provides birdsong detection accuracy in line with the best solutions in the literature but with much less computational complexity and memory demand. We also introduce BarbNet, an approximated version of ToucaNet tailored for IoT units. We show the proposed solution's effectiveness and efficiency in terms of detection accuracy and the implementation feasibility in real-world IoT devices, with specific results for the STM32 Nucleo H7 board, which is based on an ARM Cortex-M7 processor. To our best knowledge, this is the first birdsong detection algorithm designed to take into account constraints on memory, computational speed, and power usage of embedded devices pointing the way to cost-effective IoT technology for at-scale intelligent birdsong data collection and analysis in the field [Disabato et al., 2021].

CHAPTER 9

Adaptive Federated Learning in Presence of Concept Drift

Recent years showed a technological and scientific trend aiming at moving the computation (and in particular the intelligent computation) as close as possible to where data are generated. On one hand, this trend is supported by the groundbreaking technological evolution leading to a widespread diffusion of pervasive devices comprising distributed embedded systems, Internet-of-Things, and mobile devices. On the other hand, novel ML solutions able to operate in a distributed way on such pervasive devices have been recently introduced in the literature. This is where FL [Verbraeken et al., 2020] comes into play by training, in a federated way, a global model leveraging data distributed across different devices. To achieve this goal FL algorithms [Konečnỳ et al., 2015, Yang et al., 2019], rely on a server, orchestrating the global learning phase, and pervasive devices that allow a local learning phase operating on (possibly sensitive) data that are not required to be transmitted to the server. In such a pervasive scenario, FL can provide several advantages:

- *Privacy by Default*: On-Device Data Set. Each device keeps the raw data locally and these data are never shared with the server.
- *Improved Latency*: The connection between devices and server is only used to send model updates. In this way, the devices can reduce required network bandwidth and

energy consumption.

- *Training Data*: FL architectures usually include a large number of participant devices, and each of them contains a different amount and type of samples.
- *Massively Distributed*: The number of devices can be much bigger than the number of training samples on each device, which allows to have a much larger sample size during the training phase.
- *Non-independent and identically distributed (iid)*: The data on each individual device can belong to different distributions, so the centralized model can perform the training phase on a wider variety of data. We emphasize that this point is not related to variations in the data generating process over time.

The learning models that have been considered in the FL literature mainly comprise Neural Networks [Kairouz et al., 2019], decision trees [Cheng et al., 2019], linear and logistic regression [Hardy et al., 2017], while the learning algorithms implementing the FL paradigm are detailed in Section 9.1.

Interestingly, both the learning models and the learning algorithms in the field of FL share a common underlying assumption about the stationarity of the data generating process. This assumption, which is often made (implicitly or explicitly in most machine learning algorithms), guarantees that the data generating process does not change its statistical behavior over time. Unfortunately, as described in Ditzler et al. [2015], this assumption rarely holds in real-world scenarios where data coming from the field may be strongly affected by non-stationary phenomena caused, e.g., by seasonality effects, environmental evolution, or changes in the user's habits. The occurrence of such changes in the data generating process, which are also called concept drifts, make previously learned models obsolete, hence having potentially catastrophic effects on the accuracy and trust for the envisaged ML application. We emphasize that, currently, the FL literature does not provide algorithms or models meant to operate in data-generating processes affected by concept drift.

The aim of this *Chapter* is to address this open challenge by introducing Adaptive-FedAVG, a Federated Learning algorithm endowed with an adaptive step size to deal with concept drifts affecting the data-generating process. More specifically, this adaptive step size allows the proposed algorithm to promptly react to concept drifts by means of a passive adaptation mechanism [Ditzler et al., 2015] of the model parameters over the computational rounds of the FL paradigm.

A wide experimental campaign has been carried out in the field of image classification by considering two benchmark data sets, two learning models (a multi-layer perceptron and a Convolutional Neural Network (CNN)), and two different types of drift. Results have been contrasted with Federated Averaging (Fed-AVG) [McMahan et al., 2017, Li et al., 2019], a baseline FL algorithm.

The content of this *Chapter* is taken from Canonaco et al. [2021a], where I am the main researcher, and is organized as follows. In Section 9.1, the related literature is reviewed. In Section 9.2 the proposed Adaptive-FedAVG is introduced and discussed. Section 9.3 describes the experimental campaign, showing that our method performs well under stationary conditions, while it outperforms Fed-AVG in non-stationary settings. Finally, Section 9.4 draws the conclusions.

9.1 Related Works

9.1.1 Federated Learning

In the FL paradigm, each learning round relies on a server that receives local model parameters from a set of clients and aggregates them into a federated model, representing the starting point for the local models of the clients in the next learning round. The basis of FL is that aggregation of different models coming from a heterogeneous setting can strongly improve generalization on unseen data [Verbraeken et al., 2020].

FedAVG [McMahan et al., 2017] is one of the most popular FL algorithms present in literature. The core of FedAVG is to perform the server aggregation by computing a weighted average of the local model parameters. This algorithm is robust to unbalanced and non-iid data, which generally characterize FL settings. We emphasize that, here, non-iid data refer to data that are not iid among the clients, while they are assumed to be iid over time (hence the stationarity of the data generating process is still assumed). More precisely, at each round the algorithm selects a subset of clients that will take part in the current learning phase. Considering a single local epoch performed by each client, and a full batch approach, the algorithm behaves like a federated version of stochastic gradient descent. However, Fed-AVG allows more than one local epoch in the clients' training phase with the possibility to leverage a mini-batch approach. After the clients complete their local learning phase, the parameters of these locally-learned models are sent back to the server, which aggregates them through a weighted average of the associated parameters. Finally, the server sends the aggregated model to all the clients and the learning phase moves to the next round. A convergence analysis for this algorithm is provided in Li et al. [2019], considering full device participation, partial device participation, and the non-iid setting. Being $F_1, ..., F_K$ the objective functions of the K clients, respectively, the following conditions are required to guarantee the convergence of FedAVG:

- $F_1, ..., F_K$ are L-smooth;
- $F_1, ..., F_K$ are all μ-strongly convex;
- The variance of stochastic gradients in each device is bounded;
- The expected squared norm of stochastic gradients is uniformly bounded;

- The learning rate is adjusted with a time-based decay (i.e., $\eta_t = \frac{\eta_0}{1+\gamma t}$).

The last condition turns out to be a significant limitation in the implementation of adaptive solutions, as a small learning rate does not allow plasticity when a concept drift occurs.

In the context of highly heterogeneous networks, an improved version of FedAVG, called FedProx [Li et al., 2018], is proposed to deal with system and statistical heterogeneity (i.e., the variability in the characteristics of the clients and in their data distribution, respectively). Finally, another type of algorithm in the FL context is One-Shot Federated Learning [Guha et al., 2019], whose goal is restricted to training a global model by aggregating the local models in a single round of communication.

9.1.2 Learning in presence of concept drift

The literature of learning algorithms and solutions meant to operate in presence of concept drift is wide and many approaches differing in assumptions, mechanisms, performance, and considered learning models are available [Tsymbal, 2004, Gama et al., 2014]. We emphasize that the goal of these approaches is to define, develop and evaluate adaptive learning models able to react and adapt to concept drift affecting data generating processes. More specifically, as detailed in Ditzler et al. [2015], two approaches for the learning in presence of concept drift are available in the literature: Active and Passive.

The former, also called "detect and react" approach [Alippi, 2014], aims at explicitly detecting the occurrence of a concept drift through Hypothesis Tests [Patist, 2007], Change-Point Methods [Hawkins et al., 2003], or Sequential Hypothesis Tests [Wald, 1945]. Once a concept drift is detected, adaptation mechanisms such as windowing, weighting, and random sampling can be considered [Ditzler et al., 2015].

The latter relies on a continuous model update every time new data arrive, regardless of the occurrence of concept drifts. This continuous update can be performed over a single model (e.g., concept drift very-fast decision trees [Hulten et al., 2001], and online information networks [Cohen et al., 2008]), or by also adding/removing/modifying models from an ensemble (e.g., Streaming Ensemble Algorithms [Street and Kim, 2001]).

9.1.3 Adaptive Optimizers

Adaptive optimizers are suitable tools for efficient stochastic optimization allowing to compute adaptive learning rates during the training phase. Adaptive-FedAVG is based on the concepts behind adaptive optimization algorithms such as AdaGrad [Duchi et al., 2011], RMSProp [Tieleman and Hinton, 2012], and Adam [Kingma and Ba, 2014]. In particular, Adam leverages two different Exponential Moving Averages (EMAs) in order to estimate the first and second moments of the gradient. These two estimates are used to update the model and properly adapt the effective step-size at each optimization step, starting from the ratio between the estimate of the first moment and that of the second mo-

ment. Empirically, it is shown that Adam outperforms several other methods on a variety of different tasks and data sets. Moreover, it is appropriate for non-stationary objectives and very noisy and/or sparse gradient problems.

9.2 The Proposed Algorithm

9.2.1 Problem Formulation

As previously commented, the FL paradigm aims at learning a *single* global model by using data stored on a set of pervasive local devices, called clients (or workers). Being K the total number of clients and F_k the function that the k^{th}-client has to minimize, the global objective function of the FL setting becomes

$$\min_{\theta} F(\theta), \quad \text{where} \quad F(\theta) := \sum_{k=1}^{K} p_k F_k(\theta), \tag{9.1}$$

where p_k is a weight coefficient accounting for the client "importance" and $\sum_k p_k = 1$. This implies that, in principle, clients with larger weights will be considered more during the aggregation phase performed by the server. When clients are equally important, the weights p_ks are set equal to $\frac{1}{K}$ where K is the total number of clients. This is the typical setting in FL and it is the one considered throughout this *Chapter*.

Supposing that n_k training samples are available, at each round, at the k-th client, the local objective function $F_k(\cdot)$ is defined as

$$F_k(\theta) := \frac{1}{n_k} \sum_{j=1}^{n_k} \ell(\theta; x_{k,j}), \tag{9.2}$$

where l is an appropriate loss function to be minimized.

In general, $F_k(\theta)$ is considered to be a non-convex function.

In the context of this work, we will consider a *test-then-train* scenario [Ditzler et al., 2015] for both the global and local objective functions, where the training set of client k at round t is $D_k^t = \{(x_{k,i}^t, y_{k,i}^t)\}_{i=1}^{n_k}$, possibly comprising a single instance $n_k = 1$ or a batch of samples $n_k > 1$. In real-world scenarios, the data generating processes that are observed by the clients might have a non-stationary behavior, e.g., due to space-time-dependent data, user behavior, or evolving phenomena. For this reason, local data sets D_k^ts might be characterized by different behaviors at different rounds ts. In other words, in non-stationary environments, data generating processes might have a time-varying nature, and the probability distribution from which the data sets D_k^ts are sampled may evolve over time.

9.2.2 The proposed Adaptive-FedAVG algorithm

We here introduce the proposed Adaptive-FedAVG algorithm for FL in presence of concept drift. The core of the proposed algorithm resides in an adaptive learning rate to be used by the clients in their local epochs. Indeed, by means of such an adaptive learning rate, the proposed method, which is detailed in Algorithm 8 and 9, shows large plasticity in presence of concept drifts, while guaranteeing the accuracy of FedAVG in the stationary case. This aspect will be detailed in Section 9.3.

More specifically, Adaptive-FedAVG implements a server-side passive approach, in which the server increases the step-size to be used by the clients whenever there is a variability increment in the average model estimate between two consecutive rounds[1]. For this purpose, we estimate the variance of the aggregated-model parameters through an EMA. Then, the effective step-size to be used is computed as an EMA of the estimated-variance ratio between two consecutive computational rounds. This procedure is detailed in what follows by describing the Server-Side and the Client-Side behavior of Adaptive-FedAVG. We emphasize that, thanks to the adaptive learning rate, Adaptive-FedAVG is meant to guarantee the decaying property required for convergence while providing a more reactive behavior in presence of concept drifts.

9.2.3 Adaptive-FedAVG: the Server-Side

The Server-Side behavior of Adaptive-FedAVG is detailed in Algorithm 8. More specifically, at round t, the server receives clients' parameters $\theta_{t,k}$ (Line 8) and aggregates them into $\hat{\theta}_t$ via a weighted average (Line 10). The core of our proposed solution resides in the ability of the learning-rate scheduler to properly set the step-size η_t at each learning round t. In order to implement the adaptive learning rate η_t, we leverage three different EMA estimators:

- (Line 11) The average μ_t of the aggregated parameters (being β_1 the EMA decay rate)

$$\mu_t \leftarrow \beta_1 \mu_{t-1} + (1-\beta_1)\hat{\theta}_t.$$

- (Line 13) The variability S_t of the aggregated parameters (being β_2 the EMA decay rate):

$$S_t \leftarrow \beta_2 S_{t-1} + (1-\beta_2)(\hat{\theta}_t - \hat{\mu}_{t-1})(\hat{\theta}_t - \hat{\mu}_{t-1}).$$

[1] An alternative could be a Client-Side passive/active approach through which clients can adapt their learning rates to react and mitigate changes in the data. This could be done by studying the local losses, and consequently adjusting the effective step-size.

- (Line 15) The ratio between variances across two consecutive rounds γ_t (being β_3 the EMA decay rate):

$$\gamma_t \leftarrow \beta_3 \gamma_{t-1} + (1 - \beta_3) \frac{\hat{S}_t}{\hat{S}_{t-1}}.$$

By properly tuning the exponential decay rates, $\beta_1, \beta_2, \beta_3$, the algorithm supports different trade-offs between stability and plasticity [Grossberg, 1988] to adequately cope with concept drifts. More precisely, β_1 influences the stability/plasticity of the average aggregated model across the various rounds. β_2 affects the way spikes in the aggregated-model variability are captured. Finally, β_3 influences how long the spikes affecting the variance ratio persist in the computation of the step-size η_t.

Algorithm 8 Adaptive-FedAVG: Server

1: **Input:** the number of workers K; the EMA parameters $[\beta_1, \beta_2, \beta_3]$; The initial step-size η_0.
2: initialize $\hat{\theta}_0$
3: $\mu_0 \leftarrow 0$
4: $S_0 \leftarrow 0$
5: $\gamma_0 \leftarrow 0$
6: **for** each round $t = 1, 2, ...$ **do**
7: **for** each client k **in parallel do**
8: $\theta_{t,k} \leftarrow \text{ClientUpdate}(k, \hat{\theta_{t-1}}, \eta_{t-1})$
9: **end for**
10: $\hat{\theta}_t \leftarrow \frac{1}{K} \sum_{k=1}^{K} \theta_{t,k}$
11: $\mu_t \leftarrow \beta_1 \mu_{t-1} + (1 - \beta_1) \hat{\theta}_t$
12: $\hat{\mu}_t \leftarrow \frac{\mu_t}{(1-\beta_1^t)}$
13: $S_t \leftarrow \beta_2 S_{t-1} + (1 - \beta_2)(\hat{\theta}_t - \hat{\mu}_{t-1})(\hat{\theta}_t - \hat{\mu}_{t-1})$
14: $\hat{S}_t \leftarrow \frac{S_t}{(1-\beta_2^t)}$
15: $\gamma_t \leftarrow \beta_3 \gamma_{t-1} + (1 - \beta_3) \frac{\hat{S}_t}{\hat{S}_{t-1}}$
16: $\hat{\gamma}_t \leftarrow \frac{\gamma_t}{(1-\beta_3^t)}$
17: $\eta_t \leftarrow \min(\eta_0, \frac{\eta_0}{t} \hat{\gamma}\ )$
18: **end for**

After updating the moving averages, we can appropriately set the step-size η_t for the next round (Line 17). The $\min$ function between η_0 and $\frac{\eta_0}{t}\hat{\gamma}$ in Line 17 is used to avoid inappropriately big step-sizes when a concept drift occurs. Indeed, since the variance estimate S_t should go to zero in a stationary optimization process, we could have a potentially unbounded spike into the variance ratio γ_t, whenever the concept drift occurs (computed by the EMA in Line 15). Therefore, the role of the $\min$ function is to clamp this spike in order to increase the step-size to be used by the clients at most to a maximum value represented by the learning rate at the beginning of the optimization process. Finally, in Lines 12, 14, and 16, the bias due to the initialization of the EMAs to 0 is corrected. This bias

Algorithm 9 Adaptive-FedAVG: ClientUpdate

1: **Input:** the train set, D_k^t, of worker k at round t; the step-size η_t at round t; the aggregated parameters $\hat{\theta}$.
2: $\mathcal{B} \leftarrow$ (split D_k^t into batches of size B)
3: **for** each local epoch i from 1 to E **do**
4: **for** batch $b \in \mathcal{B}$ **do**
5: $\theta_{t,k} \leftarrow \hat{\theta} - \eta_t \nabla \ell(\hat{\theta}; b)$
6: **end for**
7: **end for**
8: return $\theta_{t,k}$ to server

correction is particularly important in the initial rounds, when initialization has a stronger impact on the estimates [Kingma and Ba, 2014].

At the end of the server-side algorithm, the aggregated model $\hat{\theta}_t$ and the step-size η_t are sent to the clients to allow them the next local learning phase.

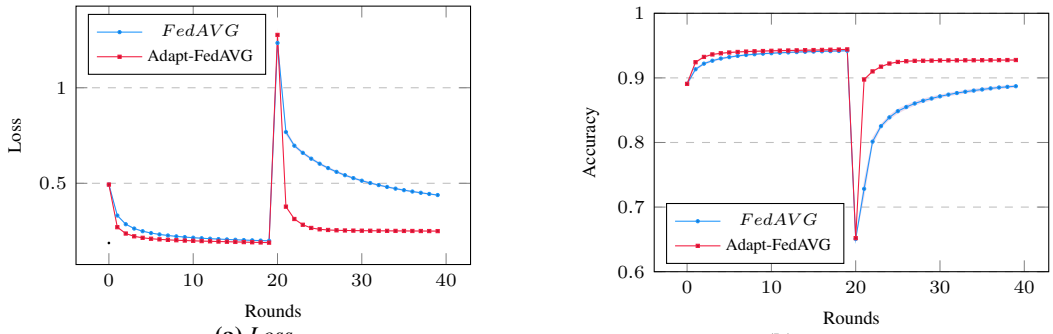
9.2.4 Adaptive-FedAVG: the Client-Side

The Client-Side behavior of Adaptive-FedAVG is detailed in Algorithm 9. More specifically, at each round, the clients that take part in the learning process receive the aggregated parameters $\hat{\theta}_t$ and the current learning rate η_t, along with their next training data set D_k^t (sampled from the field). Then, each client locally performs multiple steps of gradient descent (Line 5) by using D_k^t as training data and iterates the process for multiple local epochs. The choice of the batch-size B and the number of local epochs E is dependent on the specific task to be solved. Finally, once the local epochs are completed, each client sends back the refined model parameters $\theta_{t+1,k}$ to the server.

9.3 Experimental Results

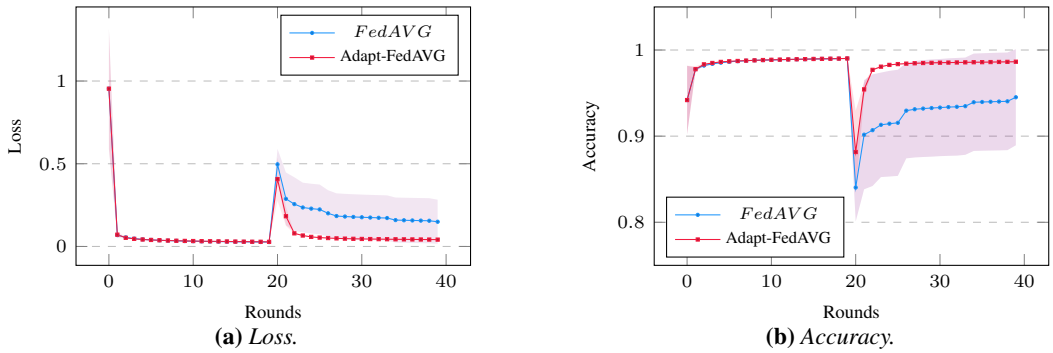
This section aims to describe the experimental campaign that has been carried out to evaluate the effectiveness of Adaptive-FedAVG both in stationary conditions and in presence of concept drifts[2]. For this purpose, we considered image classification as the application scenario and we leveraged two benchmark data sets: MNIST handwritten digits database [LeCun and Cortes, 2010] and CIFAR-10 image classification database [Krizhevsky et al., 2009]. These data sets are commonly used in FL scenarios and they are easy to adapt in order to introduce concept drifts. Moreover, the image classification task is a widely used application scenario in the related FL literature.

[2]The code for the experimental campaign is available at `https://github.com/alexbergamasco96/NS_FederatedLearning`



(a) *Loss.* **(b)** *Accuracy.*

Figure 9.1: *MNIST-MLP: Comparison between FedAVG and Adaptive-FedAVG on MNIST with the class-introduction concept drift at round* 20. $[\beta_1; \beta_2; \beta_3] = [0.5; 0.5; 0.5]$, $E = 5$, $B = 10$, *Number of clients:* 32.



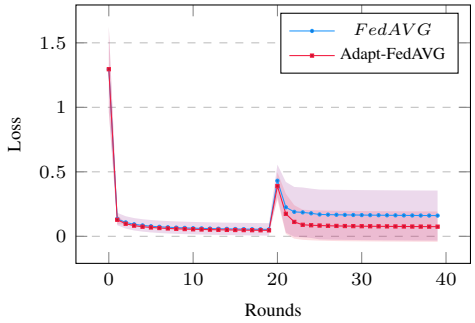
(a) *Loss.* **(b)** *Accuracy.*

Figure 9.2: *MNIST-CNN: Comparison between FedAVG and Adaptive-FedAVG on MNIST with the class-introduction concept drift at round* 20. $[\beta_1; \beta_2; \beta_3] = [0.5; 0.5; 0.5]$, $E = 5$, $B = 10$, *Number of clients:* 100.

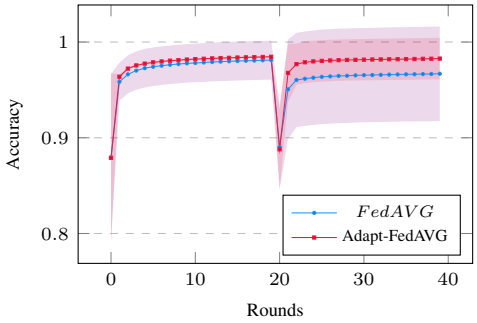
The number K of clients considered in the experimental section ranges in the following set $\{32, 64, 100\}$.

MNIST and CIFAR-10 comprise 60000 and 50000 training samples, respectively, belonging to 10 different classes. The training samples are equally partitioned among the clients, both in terms of the number of samples and in terms of class distribution. After each round, the available training data are shuffled and redistributed among all the clients, maintaining the previously mentioned characteristics, but changing the local data sets.

Throughout all the experiments, we use two different types of concept drift that are applied at a fixed round of the optimization process in order to compare Adaptive-FedAVG and FedAVG both before and after the concept drift. The first type of concept drift, i.e., *class-introduction concept drift*, consists in introducing new classes during the operational life of the image classification application. Therefore, the optimization process starts operating with a data set where only $M < N$ classes are present. Then, after the concept

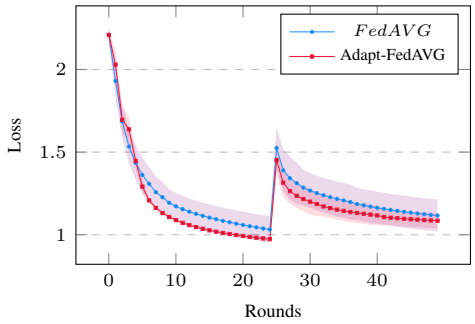


(a) *Loss.*

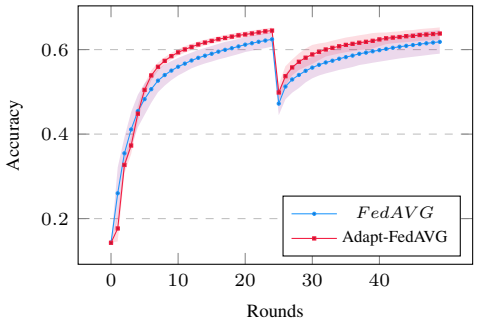


(b) *Accuracy.*

Figure 9.3: *MNIST-CNN: Comparison between FedAVG and Adaptive-FedAVG on MNIST with the class-swap concept drift at round* 20. $[\beta_1; \beta_2; \beta_3] = [0.5; 0.5; 0.5]$, $E = 5$, $B = 10$, *Number of clients:* 100.

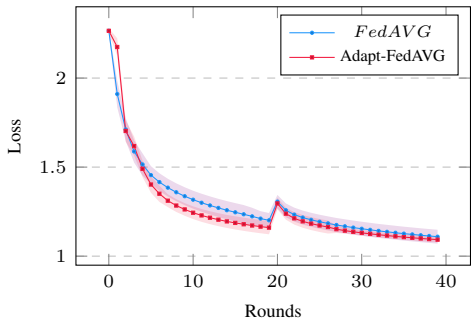


(a) *Loss.*

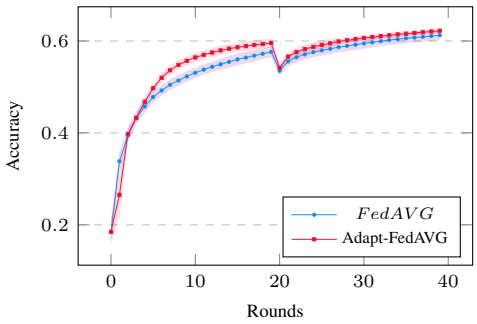


(b) *Accuracy.*

Figure 9.4: *CIFAR-CNN: Comparison between FedAVG and Adaptive-FedAVG on CIFAR-10 with the class-introduction concept drift at round* 25. $[\beta_1; \beta_2; \beta_3] = [0.7; 0.3; 0.7]$, $E = 5$, $B = 32$, *Number of clients:* 64.



(a) *Loss.*



(b) *Accuracy.*

Figure 9.5: *CIFAR-CNN: Comparison between FedAVG and Adaptive-FedAVG on CIFAR-10 with the class-swap concept drift at round* 20. $[\beta_1; \beta_2; \beta_3] = [0.7; 0.3; 0.7]$, $E = 5$, $B = 32$, *Number of clients:* 64.

drift, samples belonging to the remaining $N - M$ classes start to appear as well. In our experiments, since both MNIST and CIFAR-10 contain 10 different classes, we set $M = 7$. The second type of concept drift, i.e., *class-swap concept drift*, consists in swapping the label associated to two classes out of the available ten. In our experiment, after the concept drift occurs, the labels of the first and second classes are swapped. This setting allows us to compare the performance of both algorithms also in a stationary context, which is associated with the first part of the optimization process before the concept drift occurs.

A summary of the different types of experiments is given in Table 9.1, while the number of clients K, the values of $\beta_1, \beta_2, \beta_3$, the round at which the concept drift is inserted are specified in the figure caption for each experiment. The number of runs for each experiment is 40. In each figure, we report mean and standard deviation.

Table 9.1: *A summary of the different types of experiments described in Section 9.3.*

		Data sets	
		MNIST	**CIFAR-10**
Drift	**Class Introduction**	MNIST-MLP (Fig 1) MNIST-CNN (Fig. 2)	CIFAR-CNN (Fig. 4)
	Class Swap	MNIST-CNN (Fig. 3)	CIFAR-CNN (Fig. 5)

Full device participation and iid-partition of data characterize our experiments, while the number of local epochs E, batch-size B, number of clients K and decay parameters of the moving averages $\beta_1, \beta_2, \beta_3$ are task-specific. EMA decay parameters $\beta_1, \beta_2, \beta_3$ are appropriately tuned in order to give good performance both in stationary and non-stationary conditions. Finally, the learning rate is set initially equal for both FedAVG and Adaptive-FedAVG and it has been selected to be optimal for Fed-AVG.

9.3.1 Experiment: MNIST digit recognition

The MNIST digit recognition task is addressed by using two different neural network architectures. The former, called MNIST-MLP, is a multi-layer perceptron with two hidden layers, each of them with a ReLU activation function (the total number of parameters is 197602). The latter, called MNIST-CNN, is a CNN with two convolutional layers ($kernel - size = 5$, the first layer with 32 channels, the second with 64 channels) and a single fully-connected layer (512 neurons) with a ReLU activation function (the total number of parameters is 577922).

The partition of the data is performed in an iid fashion: each local data set contains the same number of samples and has the same distribution of the classes.

Figure 9.1 and 9.2 show how MNIST-MLP and MNIST-CNN react to the first concept drift, i.e., *class-introduction concept drift*, respectively. Several comments arise. First,

as we can see from both figures, Adaptive-FedAVG promptly reacts to the concept drift thanks to the adaptive learning rate and this results in a greater accuracy (and smaller loss) w.r.t. FedAVG after the concept drift. This is due to the fact that Fed-AVG suffers the static learning rate decay, while Adaptive-FedAVG mitigates this problem by passively increasing the step size in response to changes in the data generating process. Second, Adaptive-FedAVG and FedAVG provided similar accuracies before the concept drift (a slightly better performance in the stationary part is achieved by our solution in Figure 9.1). Finally, Figure 9.3 shows the comparison between Adaptive-FedAVG and Fed-AVG in the context of the second concept drift, i.e., the class-swap.

Results are similar in the second type of concept drift, i.e., *class-swap concept drift*, where the proposed Adaptive-FedAVG provides, in average, a larger accuracy and a smaller standard deviation both in the stationary part and after the concept drift.

9.3.2 Experiment: CIFAR-10 image classification

The CIFAR-10 image-classification task is addressed by considering a CNN, called CIFAR-CNN, for both types of drift. The considered CNN is characterized by three convolutional layers (the first with 32 channels, the second and third with 64 channels, $kernel-size = 3$) and a fully-connected layer at the end with 512 units and ReLU activation function (total number of parameters 122050). The partition of the samples among the clients is performed in the same way as in the MNIST experiments, following an iid fashion.

Similarly to the experiment of the MNIST, for the CIFAR-10 image classification task, we compare our algorithm with FedAVG on both the concept drifts previously defined.

In Figure 9.4 and 9.5, we can see how the proposed algorithm outperforms Fed-AVG both in the stationary conditions and after the concept drift in both *class-introduction concept drift* and *class-swap concept drift*. Even in this case, Adaptive-FedAVG outperforms FedAVG both before and after the concept drift, making the proposed algorithm suitable for both stationary and non-stationary environments.

9.4 Conclusions

In this *Chapter*, we presented Adaptive-FedAVG, a novel FL algorithm able to deal with data-generating processes affected by concept drifts. To achieve this goal the proposed solution introduces an adaptive learning rate able to support stability in stationary conditions and plasticity when the process generating the data is affected by concept drifts. The experimental results carried out on two image classification benchmarks show an improvement in both non-stationary conditions and results in line with the literature in stationary ones, thus making our algorithm an appealing choice in real-world FL scenarios. For what concerns future directions a detailed theoretical analysis of the algorithm is of great interest to see if the convergence properties, which have been experimentally shown in this work,

can be recovered from a theoretical standpoint, at least under stationarity assumptions.

CHAPTER 10

Birdsong Detection at the Edge with Deep Learning

In the previous *Chapter*, we focused on a methodological approach to deal with non-stationarities affecting the data generating process of a set of pervasively distributed devices. When dealing with these pervasive systems, as we have already mentioned in *Chapter* 8, we cannot always assume to have enough computational or memory resources on the edge devices, necessitating our algorithmic solutions to satisfy stricter constraints on memory and computational demand. One particular application that falls within the above mentioned highly-constrained scenario is birdsong detection/recognition for in-the-field bird population monitoring.

Bird populations have decreased significantly in many areas of the world in recent decades [Rosenberg et al., 2019]. Furthermore, there is growing evidence that the environmental effects of urban growth and human activities, e.g., increasing noise and night-time light, are at least a partial cause [Injaian et al., 2018, Senzaki et al., 2020]. Therefore, it is essential to accurately survey bird activity to better understand the behavior of species and individuals in a variety of habitats, from urban to wildland. While there has been much progress in the detection and classification of bird vocalizations [Priyadarshani et al., 2018], two primary challenges remain. First, discrimination of bird vocalizations from other sounds—especially challenging in suburban/urban environments and near roads—remains a technological challenge. Recently, this has been addressed with sophisticated

signal processing and machine learning algorithms. Secondly, improved algorithms often require more memory and greater computational load, making it more challenging to deploy cost-effective IoT systems for vocalization sensing in the field at scales supporting comprehensive surveys and more accurate assessment of the distribution of bird species.

There are several excellent algorithmic solutions for birdsong detection and classification, especially those based on Deep Learning (DL); however, they do not take into account the technological constraints of IoT systems. The overarching goal of this *Chapter* is to bridge this gap by introducing a novel algorithmic solution for birdsong detection based on Machine Learning that can be executed on off-the-shelf IoT devices. In more details, the main contributions of this *Chapter* are:

- a ToucaNet birdsong detection neural network providing detection accuracy in line with the state-of-the-art, but halving the memory footprint and reducing computational demand;
- a ToucaNet approximation, the BarbNet, able to be executed on a real-world IoT unit, i.e., the STM32H743ZI Nucleo board.

The content of this *Chapter* is taken from Disabato et al. [2021], where I am one of the two main researchers, and is organized as follows. Section 10.1 investigates the related literature. Section 10.2 proposes the ToucaNet pipeline to detect birdsongs. Section 10.3 instead describes how to approximate the ToucaNet to take into account IoT technological constraints and presents the BarbNet implementation on an STM32H743ZI microcontroller. Finally, Section 10.4 details the experimental results and Section 10.5 draws the conclusions.

10.1 Related Works

The extensive birdsong detection literature has developed solutions that can be grouped into two main families according to the processing stage in which they operate: preprocessing techniques and detection/recognition techniques.

One of the most used preprocessing approaches consists of transforming the acquired waveforms into spectrograms computed through a complex Short Time Fourier Transform (STFT), often rescaled to the Mel-Scale [Frommolt and Tauchert, 2014, Lasseck, 2013, Neal et al., 2011, Potamitis, 2014, Towsey et al., 2012]. Other approaches rely on the extraction of the Mel Frequency Cepstral Coefficients (MFCCs) [Briggs et al., 2009, Dufour et al., 2013, Graciarena et al., 2010, Kogan and Margoliash, 1998, Murcia and Paniagua, 2013], or Discrete Wavelet Transform (DWT)–based features [Bastas et al., 2012]. Approaches aiming at directly processing the waveforms are also emerging [Priyadarshani et al., 2020]. Several works also bring into play noise reduction techniques to mitigate or remove environmental or anthropogenic noise. These techniques can be applied on the

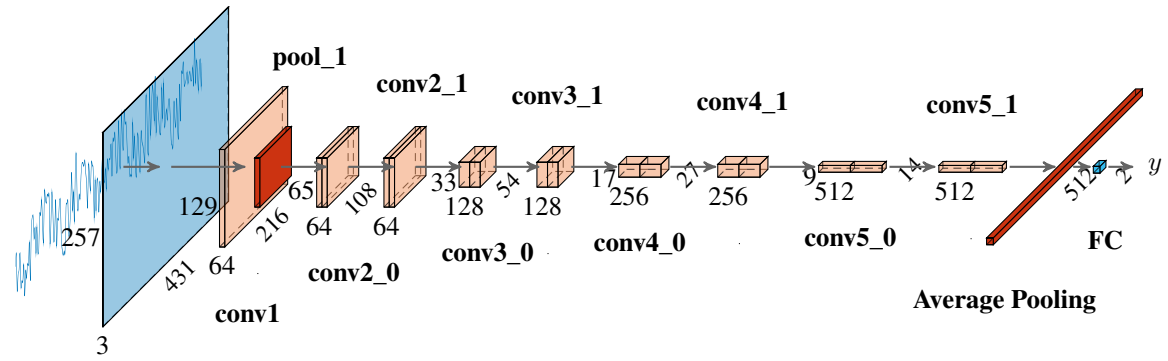
waveform [Priyadarshani et al., 2016] or spectrogram [Lasseck, 2013, Potamitis, 2014]. However, the most common noise sources overlap with low-frequency bird calls (potentially masking them) [Potamitis, 2014], hence reducing the effectiveness of subsequent detection techniques [Aide et al., 2013, Fox et al., 2006].

In the field of birdsong detection/recognition, available techniques can be grouped according to the type of approach: signal processing or Machine/Deep Learning.

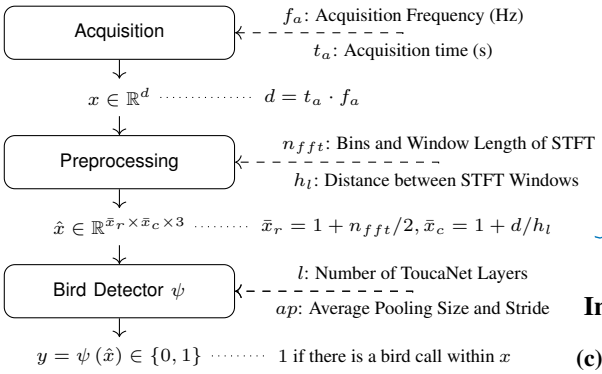
Techniques following the first approach are, e.g., Lasseck [2013] and Potamitis [2014], where segmentation of spectrograms for birdsong detection is proposed, achieving promising results on clear noise-free calls but having reduced performance when the signal is weak compared with interference and noise. Similarly, Neal et al. [2011] and Towsey et al. [2012] attempted to detect time/frequency boxes in spectrograms representing the bird calls. Finally, Frommolt and Tauchert [2014] proposed template matching to detect and isolate calls within spectrograms, whereas Anderson et al. [1996] used dynamic time warping directly on the input waveform.

Machine learning techniques for birdsong detection and recognition have mainly focused on CNNs [Berger et al., 2018, Grill and Schlüter, 2017, Lasseck, 2018, Mukherjee et al., 2018, Ruff et al., 2019], a DL architecture extensively used in image classification tasks. In this setting, Grill and Schlüter [2017] suggested two different types of CNN: working on the spectrogram of the whole audio input and processing spectrograms computed on shorter audio subsequences and then merging the predictions. Lasseck [2018] started from a pretrained CNN, either an Inception v3 [Szegedy et al., 2016] or a ResNet [He et al., 2016], and fine-tuned it to the bird detection problem in a TL fashion [Yosinski et al., 2014]. Vesperini et al. [2018] proposed a capsule-based DL architecture that organizes the convolutional layers in capsules whose outputs are "shared" in such a way that the knowledge acquired in one portion of the spectrogram can be leveraged by other spectrogram locations. Finally, Himawan et al. [2018] and Mukherjee et al. [2018] proposed two recurrent DL architectures to detect bird calls in audio signals. These DL-based approaches usually outperform the traditional signal processing techniques [Lasseck, 2018, Ruff et al., 2019]. For the recognition task, a few techniques focusing on particular groups of bird species are emerging [Brooker et al., 2020, Ferreira et al., 2020, Koh et al., 2019]. Ruff et al. [2019] proposed a CNN-based architecture to detect and recognize six nocturnal owl species, whereas Lee et al. [2012] proposed a classification algorithm for 28 bird species based on Gaussian Mixture Models (GMMs). More specifically, this solution models the likelihood of the considered bird species through GMMs, with classification according to the highest likelihood.

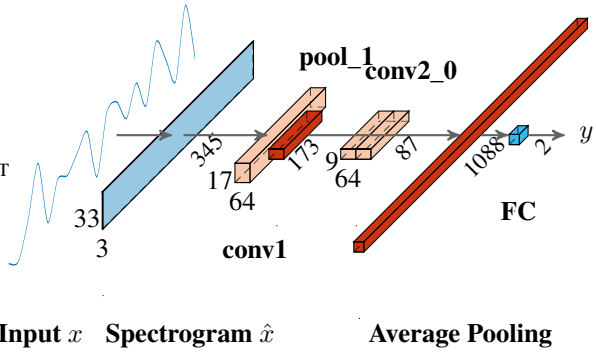
All in all, several solutions for birdsong detection are available in the literature, but none of them focus on implementing such techniques on real-world IoT devices. Interestingly, there are DL techniques for audio sources on microcontrollers [Banbury et al., 2020, Disabato and Roveri, 2020], though not for the considered application scenario.



(a) *The architecture of the ToucaNet Bird Detector ψ, that receives the spectrogram $\hat{x}$ (computed with $n_{fft} = 512$, $h_l = 512$) of a waveform x sampled at $f_a = 22050$ Hz for $t_a = 10$s, and provides its output classification y.*



(b) *A scheme of the pipeline with the approximable parameters.*



(c) *The BarbNet Bird Detector ψ architecture, that receives the spectrogram $\hat{x}$ ($n_{fft} = 64$ and $h_l = 64$) of a waveform x sampled at $f_a = 2205$ Hz for $t_a = 10$s, and provides its output classification y on a STM32H743ZI IoT device.*

Figure 10.1: *Comprehensive scheme of the proposed solution to detect bird calls in audio acquired on the field.*

10.2 The Proposed ToucaNet Bird Detector

The architecture of the proposed ToucaNet bird detector, which is depicted in Figure 10.1b, comprises two main steps. The rest of the section describes both steps in detail.

10.2.1 Acquisition and Preprocessing

Let f_a be the microphone's sampling frequency (in Hertz) and t_a be the acquisition time window (in seconds). Let $x \in \mathbb{R}^d$, with $d = f_a \times t_a$, be the d-dimensional vector acquired by the microphone, representing the waveform to be preprocessed and subsequently pro-

cessed by DL-based birdsong detector.

Once the microphone acquires the waveform x, the preprocessing phase converts it into the corresponding spectrogram $\bar{x}$ computed via the (absolute value of the complex) STFT with n_{fft} bins over constant-length windows spaced by h_l samples, with both n_{fft} and h_l constrained to be powers of two.[1] The outcome of the STFT is a two-dimensional matrix with $\bar{x}_r = 1 + n_{fft}/2$ rows and $\bar{x}_c = 1 + d/h_l$ columns, which is converted into a three-dimensional image by means of a color-map, i.e., $\hat{x} \in \mathbb{R}^{\bar{x}_r \times \bar{x}_c \times 3}$, to enable transfer learning (see below). In the following, $\hat{x}$ will be called a spectrogram.

10.2.2 The DL-based Birdsong Detector

The DL-based bird detector $\psi(\hat{x}, \theta)$ with parameters θ of the ToucaNet receives as input the spectrogram $\hat{x}$ and produces as output its binary classification $y \in \{0, 1\}$, where $y = 1$ if a bird call is present within the waveform, and $y = 0$ otherwise. This formalization can be extended easily to the case of birdsong recognition by simply considering the supervised information y to belong to a discrete set of classes representing the various bird species. In what follows, to simplify the notation, θ will be omitted from $\psi(x, \theta)$.

Since the spectrogram is interpreted as a colored image $\hat{x}$, it is possible to leverage approaches from the image classification field. More precisely, in the context of this work, the ToucaNet birdsong detector ψ is built upon a ResNet-18 [He et al., 2016], leveraging all its convolutional layers (namely, up to layer *conv5_1*). A 9×14 average pooling filter (with stride 1 and no padding) follows the convolutional structure. Finally, a fully-connected classifier labels the 512 extracted features into the desired two classes (Figure 10.1a depicts the ToucaNet architecture in detail).

The convolutional structure of the ToucaNet is initialized with the weights associated to the optimal solution of Resnet-18 on the ImageNet classification task [Deng et al., 2009], hence relying on a TL approach [Yosinski et al., 2014] to speed up the training phase. The fully-connected layer is instead initialized with a uniform distribution in the interval $\left(-1/\sqrt{fc_{in}}, 1/\sqrt{fc_{in}}\right)$, where fc_{in} is the input size of the layer itself, i.e., 512.

To train ψ, an n-dimensional data set $D = \{(x_i, y_i)\}_{i=1}^{n}$ is used and the parameters θ are optimized as

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} l(\hat{x}_i, y_i, \theta), \tag{10.1}$$

where $l(\cdot)$ is a classification loss function and $\hat{x}_i$ is the spectrogram of the waveform x_i. ψ is trained for 14 epochs with stochastic gradient descent, momentum 0.9, and learning rate of $5 \cdot 10^{-2}$, decreased by a factor of 10 after the eighth and the twelfth epochs.

Since our final goal is to move our solution to an IoT unit, the memory footprint m_ψ and the computational load c_ψ of ψ have been carefully evaluated. By extending the for-

[1] The Mel-spectrogram is not considered, since it uses a human-based frequency scale.

malization introduced by Alippi et al. [2018], m_ψ and c_ψ are defined as

$$m_\psi = |\theta| \cdot m_p + m_{in}, \tag{10.2}$$

$$c_\psi = n_{mul}, \tag{10.3}$$

where $|\theta|$ is the cardinality of θ, n_{mul} is the number of multiplications required to compute the output, m_{in} is the memory needed to store the input x (and all its transformations), and m_p is the memory required to store a single parameter (typically a 32-bit floating-point data type).[2]

As shown in Table 10.3 of Section 10.4 and Figure 10.3, the ToucaNet has detection capabilities in line with the best solution available in the related literature given the same acquisition frequency ($f_a = 22050$ Hz) but half its memory footprint ($m_\psi = 44.714$ MB) and about 80% of its computational demand ($c_\psi = 4.255$ billion multiplications).

These computational demand and memory footprint improvements provide a more efficient but not less effective birdsong detection solution. However, these requirements are still too demanding to allow deployment on IoT units. Therefore, in Section 10.3, the proposed ToucaNet is approximated to enable its deployment on off-the-shelf IoT systems.

10.3 BarbNet: the Approximated ToucaNet for IoT Units

To deploy a bird song detector at the IoT edge, its memory and computational requirements must meet the constraints ($\bar{m}$ and $\bar{c}$) of the selected IoT node. To achieve this, ToucaNet has been revised by introducing approximations both at the acquisition-preprocessing and birdsong detection layers, as described respectively in Sections 10.3.1 and Section 10.3.2. We note that reducing the computational demand as much as possible also saves energy and prolongs the IoT device activity in the field. Figure 10.1b depicts the approximation parameters. Finally, Section 10.3.3 introduces BarbNet, the approximated version of ToucaNet that can satisfy the memory and computational constraints of a typical IoT prototyping platform, the STM32H743ZI board by STMicroelectronics.

10.3.1 Approximating the Input

The Acquisition and Preprocessing step of the ToucaNet comprises the STFT to compute the spectrogram $\hat{x}$, whose computational complexity is $O\left(d \cdot n_{fft} \cdot \log\left(n_{fft}\right)\right)$, which is negligible compared to the one required by the DL bird detector. However, the total memory footprint m_{in} of the Acquisition and Preprocessing step is non-negligible and comprises the memory footprint m_x of the input signal x and the memory footprint $m_{\hat{x}}$ of the

[2]The computational complexity is defined in terms of the number of multiplications, the slowest operation, since it is not straightforward to define the concept of computation time. Nevertheless, the time needed to execute a given number of operations can be estimated with a further hyper-parameter modeling the (mean) number of operations carried out per second.

spectrogram $\hat{x}$, i.e.,

$$m_{in} = (m_x + m_{\hat{x}}) \cdot m_p = (d + \bar{x}_r \cdot \bar{x}_c \cdot 3) \cdot m_p. \tag{10.4}$$

Therefore, the memory footprint m_{in} of the Acquisition and Preprocessing step depends on the sampling frequency f_a, the acquisition time t_a, and the STFT parameters n_{fft} and h_l. Consequently, although f_a is application-dependent and, in principle, must be set two times larger than the maximum frequency in the waveforms (following the Nyquist-Shannon sampling theorem), in this context, it needs to be considered as an approximation parameter. Moreover, reducing f_a will also reduce the computational demand c_ψ of the bird detector since the spectrogram will be smaller. Since reducing f_a might introduce aliasing, an anti-aliasing filter (tailored to f_a) is advised before the ADC to limit the input signal bandwidth.

For the other three parameters, t_a, n_{fft}, and h_l, the former is set according to the available data sets (i.e., fixed to $t_a = 10$s), and the latter two are set such that each window of the spectrogram is 30ms and non-overlapping (refer to Section 10.4–Table 10.2 for details) to cover approximately one syllable of a bird call [Hart et al., 2018, Marler and Isaac, 1960, Paliwal et al., 2010].

10.3.2 Approximating the DL-Based Birdsong Detector

The memory footprint and computational demand of the ToucaNet birdsong detector are controlled by reducing the number of parameters θ and the number of features in the input to the ToucaNet fully-connected layer. In the former approach, the ToucaNet birdsong detector is approximated by removing layers from the ResNet-18-based part of the pipeline. The higher the number of layers l kept in the approximated ToucaNet, the higher the detection capability at the expense of a larger memory footprint and more significant computational load. Section 10.4.2 analyses all the configurations using the first five convolutional blocks, i.e., $l \in \{pool1, conv2\_0, conv2\_1, conv3\_0, conv3\_1\}$. The choice of l strictly depends on the IoT unit constraints $\bar{m}$ and $\bar{c}$. For instance, the memory constraint $\bar{m}$ of the STM32H743ZI unit requires the BarbNet (Figure 10.1c) to have $l = conv2\_0$.

To reduce the number of features in input to the final classifier, a dimensionality reduction operator consisting of an $ap \cdot ap$ average pooling layer (stride ap and no padding) is introduced, where ap is an additional hyper-parameter of the model ψ.[3] Although other dimensionality reduction operators, such as PCA [Pearson, 1901] or auto-encoders, could have been considered, the average pooling operator has the advantage of having negligible computational demand and no memory footprint, making it a very appealing choice in this setting.

Table 10.1: *The detailed memory footprint (with a 32-bit data type) and the computational requirements of the BarbNet implemented on the STM32H743ZI. To optimize the memory, two arrays only are used to store the activations (an asterisk marks the activations re-using such arrays).*

	Memory Footprint (KB)	10^6 operations
Audio ($t_a = 10$ s, $f_a = 2205$ Hz)	*172.76	-
Spectrogram ($n_{fft} = 64, h_l = 64$)	*133.42	-
Conv1 (Weights)	37.75	-
Pool1 (Weights)	-	-
Conv1–Pool1 (Activations)	195.75	28.120
Conv2_00 (Weights)	144.00	-
Conv2_00 (Activations)	195.75	28.865
Conv2_01 (Weights)	144.00	-
Conv2_01 (Activations)	*195.75	28.865
Avg Pool with $ap = 5$ (Weights)	-	-
Avg Pool with $ap = 5$ (Activations)	*4.25	0.010
Fully-Connected Classifier (Weights)	8.5	0.002
Fully-Connected Classifier (Activations)	0.008	-
Convolutions Auxiliary Memory	28.50	-
Total	754.26	85.878

10.3.3 A Bird Song Detector on an ARM Cortex-M7: BarbNet

In this work, the target IoT device is the STM32 Nucleo H743ZI2 MCU endowed with a 480 MHz ARM Cortex-M7 processor, 1024 KB of RAM, 2 MB of Flash, and no Operating System. The memory constraint $\bar{m}$ is set to the RAM size, i.e., $\bar{m} = 1024$ KB. The BarbNet (Figure 10.1c) is the approximated version of the ToucaNet satisfying these constraints.[4]

Table 10.1 details the memory and computational demand of BarbNet, assuming a 32-bit floating-point representation for all the weights and activations. More specifically, the BarbNet samples at $f_a = 2205$ Hz and generates spectrograms $\hat{x}$ with $n_{fft} = 64$ and $h_l = 64$. In principle, this small value for f_a might prevent the detection of higher-frequency birdsongs but Section 10.4–Figures 10.2 and 10.3 show that the figures of merit are still good on real benchmarks.

The required memory footprint for both the signal of size $d = 22050$ and the resulting spectrogram ($\bar{x}_r = 33$, $\bar{x}_c = 345$) is 306.18 KB. The first convolutional layer of the ResNet-18, characterized by 64 7×7 filters with stride 2 and padding 3, processes

[3]Please note that setting $ap = 1$ skips this operator.
[4]Please refer to Section 10.4.2 for discussion of other feasible configurations.

Table 10.2: *A summary of the considered acquisition frequencies t_a, along with the details of the resulting spectograms, and its memory occupation $M_{\hat{x}}$, assuming a 32-bit data type.*

f_a (Hz)	n_{fft}	h_l	$\bar{x}_r$	$\bar{x}_c$	$M_{\hat{x}}$ (KB)
1100	32	32	17	344	68.53
2205	64	64	33	345	133.42
4410	128	128	65	345	262.79
8820	256	256	129	345	521.54
17640	512	512	257	345	1039.04
22050	512	512	257	431	1298.05

the computed spectrogram $\hat{x}$. A batch-normalization layer and a 3×3 maximum pooling with stride 2 follow. This block accounts for 28 million operations, requires 37.75 KB of memory to store the weights, and generates $9 \cdot 87 \cdot 64$ activations, requiring 195.75KB of memory. Two 3×3 convolutions compose the second convolutional block (*conv_0*) with 64 filters with stride 1 and padding 1, each followed by a batch normalization. This block does not change the activation size, has a memory footprint of 288KB, and requires nearly 58 million operations. Finally, the average pooling layer has (size and stride) $ap = 5$ and requires 10000 multiplications. The pooling generates, after flattening, 1088 activations that are provided to the fully-connected classifier, which in 2176 operations yields the final classification label, requiring 8.5KB of memory. Consequently, the total number of BarbNet multiplications is 84 million, and its total memory footprint is 755KB.[5] Note that two arrays are used alternatively to store all the inputs and intermediate representations, saving 506.18KB of memory.

To further reduce the computational demand, a few device-dependent optimizations have been implemented in the BarbNet deployed version on the STM32H743ZI unit, relying on the CMSIS-DSP Cortex-M7 processor's instruction set:

- As in the most prominent DL libraries (e.g., Chetlur et al. [2014]), the convolution layer has been converted into a matrix multiplication. The convolutional input patches (i.e., all the activation pieces the convolutional filters are multiplied by) are unrolled and organized as rows in the first matrix of shape $(n_{patches}, f_s)$, with $n_{patches}$ the number of patches and f_s the dimension of each (unrolled) filter. The n_{fft} convolutional filters (of size f_s) are instead rolled out and arranged as columns in the second matrix of shape (f_s, n_{fft}). Finally, the multiplication result of size $(n_{patches}, n_{fft})$ between the *patches* and filter matrices is reshaped to match the convolutional output;
- Since the convolutional filters embrace overlapping patches, the previously described creation of the *patches* matrix will result in a non-negligible increment of the memory footprint. To balance this increment, the maximum number of patches multiplied at

[5] The total number of multiplications is an upper bound since it does not consider all the optimizations.

any given time, $\hat{n}_{patches}$, is defined as an additional parameter of the problem. In our implementation, $\hat{n}_{patches}$ is fixed to the number of columns, which means that to carry out a convolution, the number of employed matrix multiplications equals the number of convolutional input rows;

- Batch normalization and maximum pooling operations are computed simultaneously with the previous convolutional layer;
- The zero-padding (to be done in all the convolutional and pooling layers) is implemented without explicitly instantiating the "padded" array;
- The fully-connected layer is carried out as a matrix multiplication via CMSIS-DSP instructions.

10.4 Experimental Results

This section details the experimental results. Section 10.4.1 describes the employed data sets and figures of merit. Section 10.4.2 evaluates the detection capabilities along with memory and computational demands of the ToucaNet and several approximations (including the BarbNet), whereas Section 10.4.3 compares ToucaNet and BarbNet with the solutions in the literature. Finally, Section 10.4.4 reports the execution time and power consumption of the BarbNet implementation on the STM32H7 IoT device.

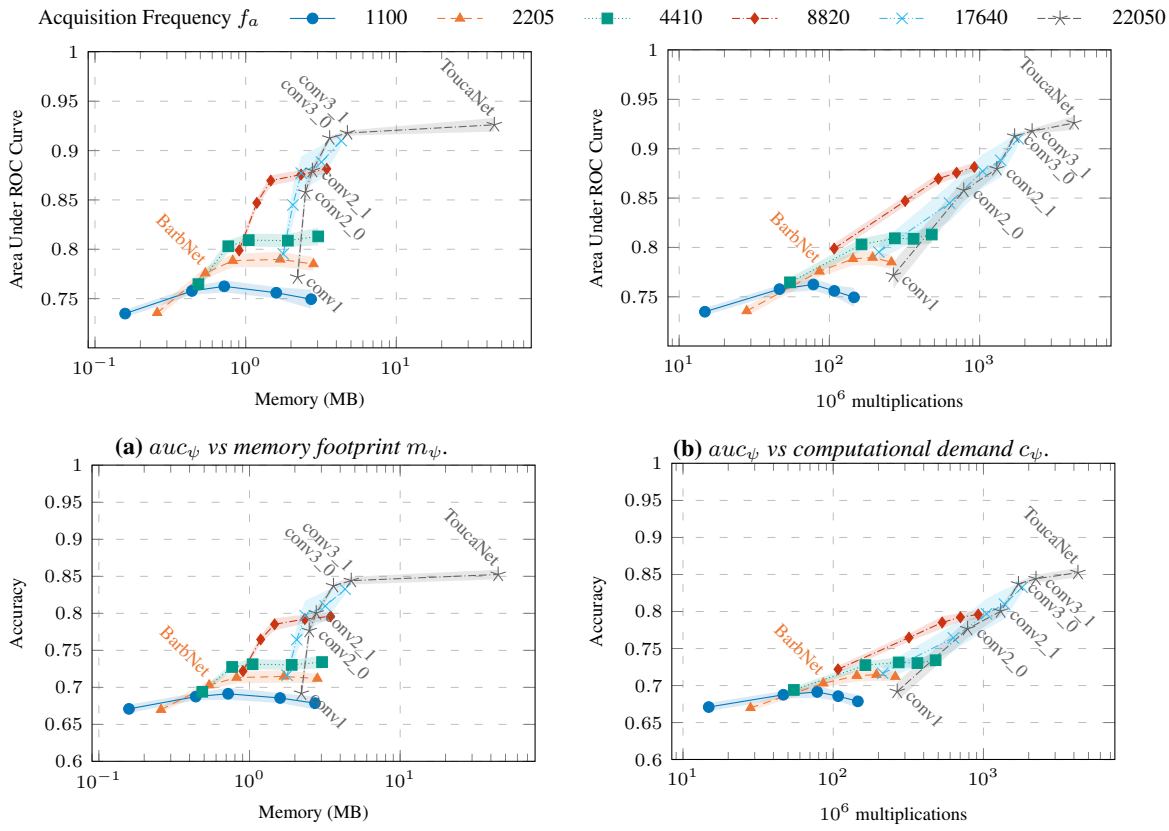
10.4.1 Data Sets and Figures of Merit

The experimental results have been collected on three different data sets [Stowell et al., 2019]: *Warblbr*, composed of approximately 8000 10s recordings from smartphones (in UK); *Free-Field*, ccontaining 7690 10s recordings from a research project; and *BirdVox-DCASE-20k*, namely the *remote monitoring flight calls* data set, that contains 20000 10s audio clips recorded remotely near Ithaca, NY, USA, during autumn 2015. The acquisition time is fixed to $t_a = 10$s to use the available supervised information and compare it with the related literature on the same data sets. Other data sets in the field, such as the *Chernobyl Exclusion Zone (CEZ)* or the *Remote monitoring night-flight calls in Poland*, have not been considered since the labels are not publicly available.

In the comparison of the birdsong detection solutions, the following two figures of merit are employed: the birdsong detection accuracy acc_ψ and the Area Under Curve (AUC) auc_ψ of the Receiver Operating Characteristic (ROC) on the class *bird*. In the experimental results, such figures of merit have been computed through 10-fold CV.

10.4.2 Pareto Frontiers of the ToucaNet and its Approximations

As mentioned in Sections 10.3.1 and 10.3.2, both the sampling frequency f_a and the DL-based detector ψ need to be tailored to the target IoT device. Therefore, in this section, the



(a) auc_ψ *vs memory footprint* m_ψ.

(b) auc_ψ *vs computational demand* c_ψ.

(c) acc_ψ *vs memory footprint* m_ψ.

(d) acc_ψ *vs computational demand* c_ψ.

Figure 10.2: *Outcomes in terms of AUC (a,b) and accuracy (c,d) against the memory footprint and computational demand by the ToucaNet and its approximations A (in terms of acquisition frequency f_a and layer l, annotated for one plot only).*

ToucaNet and a set of its approximations are analyzed in terms of acc_ψ, auc_ψ, m_ψ, and c_ψ. The considered set of approximations is:

$$A = \{(f_a, l) : f_a \in \{1100, 2205, 4410, 8820, 17640, 22050\} \text{ Hz}, \\ \text{and } l \in \{conv1, conv2\_0, conv2\_1, conv3\_0, conv3\_1\}\}.$$

Notice that l defines the ResNet block's considered number of convolutional layers within the ToucaNet (see Figure 10.1a), so if l is $conv2\_1$ then the ResNet block is approximated by considering all the layers up to $conv2\_1$. The BarbNet belongs to the set A since it has $f_a = 2205$ Hz and $l = conv2\_0$.

All the ToucaNet approximations are trained similarly to what is described in Section 10.2.2. The initialization of layers follows the description for the ToucaNet, the train-

Table 10.3: *A comparison of the ToucaNet and BarbNet with the related literature. The complexities are computed according to the description of the proposed solution, whereas the figure of merit is taken as it is since the adopted datasets are the same.*

Solution	Input Details			Memory Footprint m_ψ (MB)		c_ψ	auc_ψ
	f_a (Hz)	t_a (s)	Dimensions	Input	Parameters	10^9 operations	
BarbNet	2205	10	33x345x3	0.215	0.383	0.086	0.778 ± 0.006
BarbNet (2)	17640	10	257x345x3	1.941	0.383	0.631	0.853 ± 0.018
ToucaNet	22050	10	257x431x3	2.108	42.606	4.255	$\mathbf{0.925 \pm 0.008}$
Lasseck [2018]	22050	10	299x299x3	1.864	≈ 92	≈ 5	**0.932**
CRNN in Mukherjee et al. [2018]	44100	10	128x768	2.057	4.452	> 6.460	0.855
CNN in Mukherjee et al. [2018]	44100	10	128x768	2.057	1.127	6.460	0.842
Berger et al. [2018]	44100	10	80x1000	1.897	≈ 4.270	≈ 0.162	0.817
Berger et al. [2018]	44100	10	80x1000	1.897	1.423	0.054	0.803
CapsNet1 Vesperini et al. [2018]	16000	10	501x40	0.687	0.431	-	0.784
CapsNet2 Vesperini et al. [2018]	16000	10	501x40	0.687	1.076	-	0.827
CapsNet3 Vesperini et al. [2018]	16000	10	501x40	0.687	1.617	-	0.837
CapsNetEnsemble Vesperini et al. [2018]	16000	10	501x40	0.687	3.124	-	0.851

ing epochs are 14, the momentum is 0.9, and the learning rate is $5 \cdot 10^{-3}$ decreased by a factor of 10 after the sixth and the tenth epoch.

For each configuration in A, Figure 10.2 reports auc_ψ vs. m_ψ, auc_ψ vs. c_ψ, acc_ψ vs. m_ψ, and acc_ψ vs. c_ψ. In particular, in Figures 10.2a and 10.2b it is possible to observe the Pareto Frontier generated by the considered ToucaNet approximations. This frontier can be leveraged as a criterion to choose the architecture to deploy according to the constraints imposed by a given IoT device. More specifically, in our case, given the memory constraint $\bar{m} = 1024$ KB of the STM32H743ZI unit, there are five solutions on the Pareto Frontier that are suitable for the deployment. The BarbNet is the fourth one with the highest AUC.[6] Finally, as expected, the ToucaNet is the solution achieving the highest accuracy and AUC.

10.4.3 Comparing ToucaNet and BarbNet with the State-of-the-Art Solutions

Table 10.3 and Figure 10.3 compare both the ToucaNet and the BarbNet with state-of-the-art solutions in the birdsong detection literature [Lasseck, 2018, Mukherjee et al., 2018, Berger et al., 2018, Vesperini et al., 2018]. Two main comments arise. First, ToucaNet has detection capabilities in line with the best solution in the literature, i.e., Lasseck [2018], but with half its memory footprint and about 80% of its computational demand. Second, BarbNet is the only solution satisfying the requirements of the STM32H743ZI unit and, when it samples the input signal x at $f_a = 17640$ Hz, equals or exceeds the accuracy and AUC of all the other solutions with similar memory footprint (Figure 10.3).

[6] The fifth and best one is unfeasible when also considering the intermediate activations of the convolutional layers.

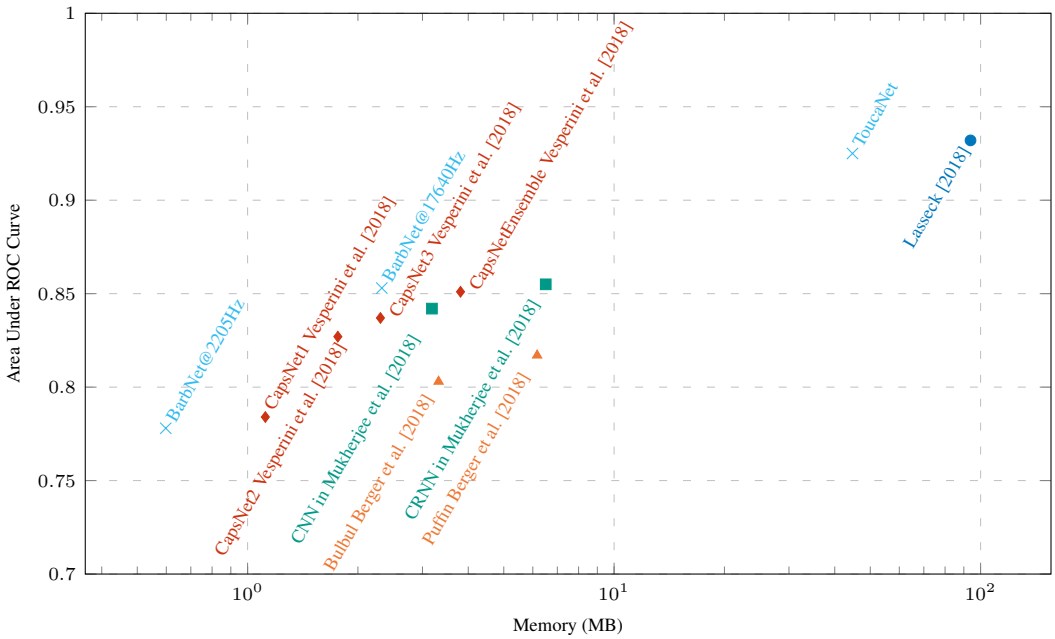


Figure 10.3: *auc_ψ vs memory footprint m_ψ outcomes obtained by the proposed solution working on an STM32H743ZI and other solutions available in the related literature.*

10.4.4 BarbNet on the STM32H7: Execution Time, Energy Consumption, and Lifetime.

A detailed experimental analysis to quantitatively measure the execution time, the energy consumption, and the expected lifetime of the BarbNet running on STM32H7 has been conducted, as detailed in the sequel.

Table 10.4 details the execution time results from audio acquisition to the last layer of BarbNet and shows that BarbNet requires 3.285s to compute the classification y, well below the time needed to acquire the audio signal (10s). Hence, embedded systems endowed with multi-cores or Dynamic Memory Allocation (DMA) mechanisms enable parallel classification and waveform acquisition. In the technological scenario with a single core and DMA not considered, we suggest two STM32H7 running in opposite modes. While the former is acquiring the waveform, the latter predicts (with the BarbNet) on the previously acquired waveform and then sleeps up to the next acquisition step. The lifetime evaluation is based on this "*acquire-classify-sleep*" approach.

In more detail, table 10.5(a) reports the energy required by each step, showing that the most expensive one is the DL-Based Birdsong Detector *computation* step. Given the energy consumptions detailed in Table 10.5(a), Table 10.5(b) analyses the lifetime of the proposed solution on the STM32H743ZI microcontroller with several real Li-Ion batteries.

Table 10.4: *The BarbNet experimental execution timings on the STM32H743ZI, measured with an oscilloscope.*

	Time (ms)
Audio ($t_a = 10$ s, $f_a = 2205$ Hz)	10 000.00
Spectrogram ($n_{fft} = 64, h_l = 64$)	22.80
Conv1–Pool1	432.00
Conv2_00	1 360.00
Conv2_01	1 470.00
Avg Pool ($ap = 5$)	$\ll 1.00$
Fully-Connected Classifier	$\ll 1.00$
Total	3 284.80

Table 10.5: *The energy analysis of the BarbNet deployment on the STM32H743ZI, when considering a 3.3V power supply.*

(a) *Energy Consumption.*

	t (s)	i (μA)	P (mW)	E (J)
Acquisition	10.000	721.95	2.38	0.000052
Computation	3.285	55 000.00	181.50	0.596228
Sleep	6.715	1.95	0.006	0.000043

(b) *System Lifetime with different Li-Ion batteries. The reported battery energy is 75% of its nominal capacity.*

Battery Capacity	Battery Energy	Lifetime	
mAh	J	Hours	Days
100	1 332	9.31	0.39
250	3 330	23.27	0.97
500	6 660	46.54	1.94
1 000	13 320	93.07	3.88
1 200	15 984	111.68	4.65
1 500	19 980	139.61	5.82
2 000	26 640	186.14	7.76
2 500	33 300	232.68	9.69
3 200	42 624	297.83	12.41

To account for battery non-idealities and degradation over time, only 75% of the nominal battery capacity is considered. Interestingly, the considered IoT unit running the BarbNet provides a 7 and 12.4 day lifetime when using a 2000 mAh and 3200 mAh capacity Li-ion battery, respectively.

10.5 Conclusion and Future Work

While the literature on birdsong detection (and recognition) is extensive, none of the proposed algorithms considers the technological constraints imposed by IoT units at the edge of a pervasive system. The ToucaNet solution proposed here performs with accuracy comparable to state-of-the-art but with lower memory footprint and computational demand. Furthermore, the BarbNet, an approximated version of ToucaNet, can be deployed on an actual IoT device.

Potential future work encompasses: the approximation of ToucaNet parameters with fixed-point representations or 16-bit floating-point data; introducing the recognition task (directly on the IoT unit or on a second IoT device in a hierarchical classification approach); and tailoring of the proposed solution to local populations and environmental conditions.

CHAPTER 11

Final Remarks

Machine Learning tools and techniques are often equipped with a set of assumptions that may or may not be satisfied in practical applications. Whenever these assumptions do not hold, we have a gap to bridge between theory and practice in order for ML-based tools to be appropriately leveraged in the real world. In the context of this dissertation, starting from the needs of a very specific application, that one of corrosion prediction in pipeline infrastructures, we softened some of those assumptions. Specifically, we have built learning algorithms for corrosion prediction with the final aim to circumvent the need for supervised information coming from the pipeline of interest. While the lack of supervised information is a widely recognized and researched problem, it is currently overlooked in the field of corrosion prediction for pipeline infrastructures. Furthermore, motivated by the time-variability of the corrosion phenomenon, we investigated, in different ML fields, solutions able to weaken the stationarity assumption on data-generating processes. More precisely, in the Reinforcement Learning framework, we proposed solutions able to deal with non-stationarities at two different levels: one in the context of the current task and the other in the context of the distribution underlying the task generating process. In the context of pervasive systems, instead, we proposed a passive-adaptive approach to deal with non-stationary data-generating processes for the Federated Learning framework, and, we concluded with an application of ML-based tools that can be deployed in highly con-

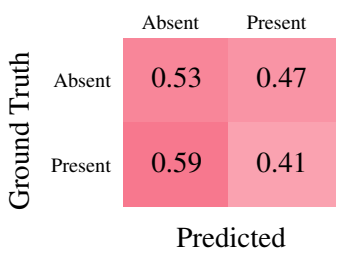
strained devices for in-the-field bird population monitoring.

The interconnections among all the topics we treated in the context of this dissertation are countless and far from being herein exhausted. For instance, being able to take into account time variations in the available historical knowledge of a Transfer Learning (TL) algorithm is not relevant only for the RL framework, but it is a concept that could be exported also in the context of SL algorithms to build better active-adaptive approaches for learning in non-stationary environments. Furthermore, RL techniques could be used in the context of corrosion prediction in order to obtain maintenance policies for the infrastructures of interest given that we manage to learn a model for the evolution of the corrosion phenomenon through time. Finally, the FL paradigm could be used in the context of birdsong detection/recognition to learn a global federated model using data directly sampled from different regions of a given country. All of the above-mentioned connections represent interesting and broad spectrum future perspectives for this dissertation.

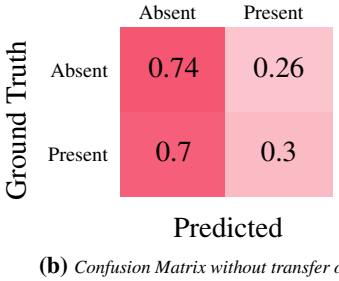
APPENDIX $\mathcal{A}$

Corrosion Detection Experimental Results

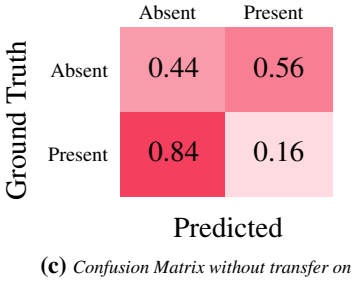
In this section are reported all the binarized versions of the confusion matrices presented in Section 4.3. This is done to check the detection capabilities of the various models.



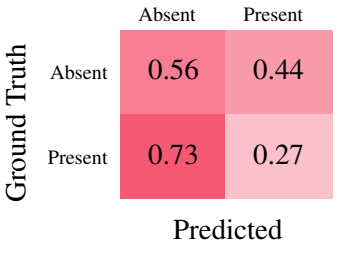
(a) *Confusion Matrix without transfer on P1 using all the sources.*



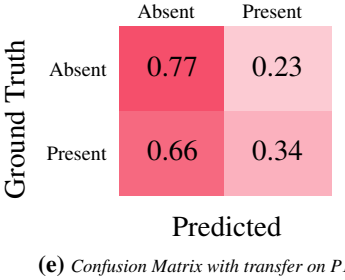
(b) *Confusion Matrix without transfer on P1 using the first two sources according to the IWCV metric.*



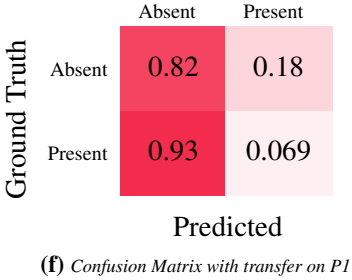
(c) *Confusion Matrix without transfer on P1 using solely the best source according to the IWCV metric.*



(d) *Confusion Matrix with transfer on P1 using all the sources.*

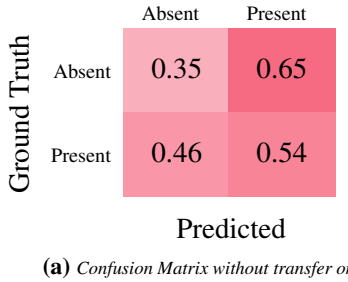


(e) *Confusion Matrix with transfer on P1 using the first two sources according to the IWCV metric.*

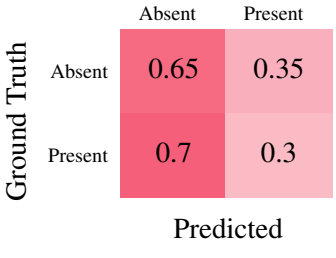


(f) *Confusion Matrix with transfer on P1 using solely the best source according to the IWCV metric.*

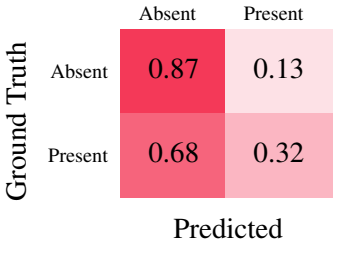
Figure A.1: *Binarized Confusion Matrices on P1.*



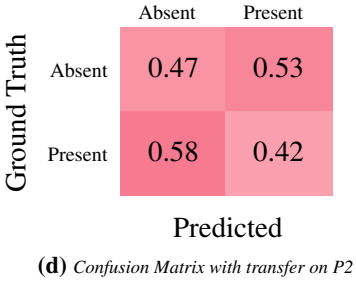
(a) *Confusion Matrix without transfer on P2 using all the sources.*



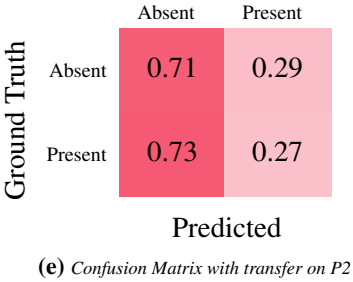
(b) *Confusion Matrix without transfer on P2 using the first two sources according to the IWCV metric.*



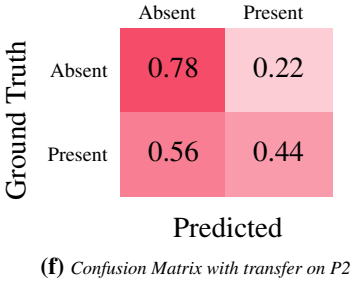
(c) *Confusion Matrix without transfer on P2 using solely the best source according to the IWCV metric.*



(d) *Confusion Matrix with transfer on P2 using all the sources.*

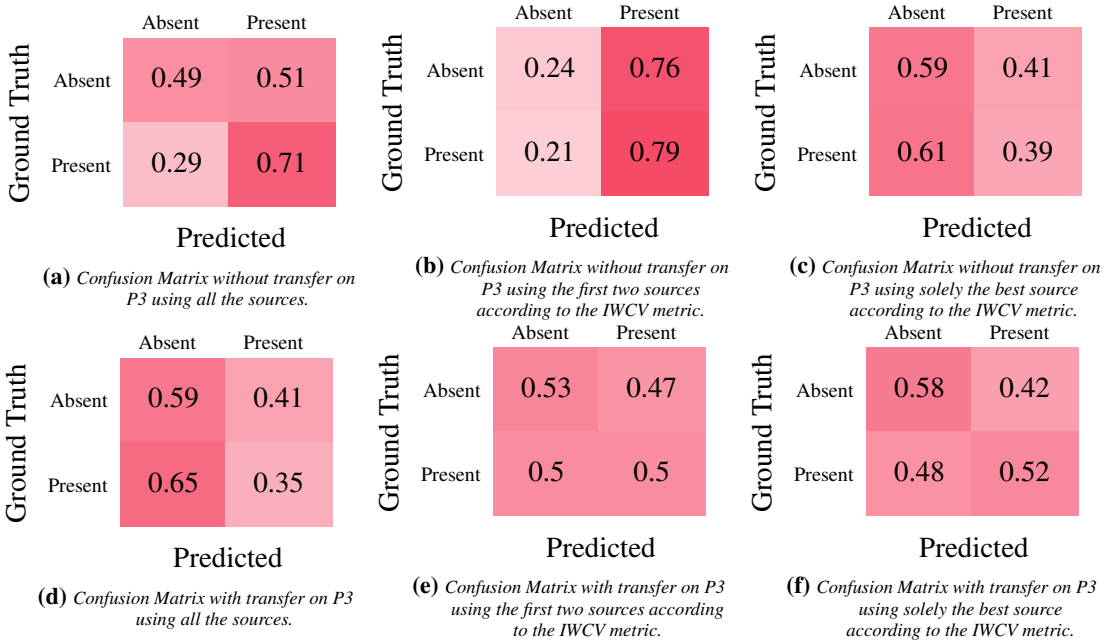


(e) *Confusion Matrix with transfer on P2 using the first two sources according to the IWCV metric.*



(f) *Confusion Matrix with transfer on P2 using solely the best source according to the IWCV metric.*

Figure A.2: *Binarized Confusion Matrices on P2.*



(a) *Confusion Matrix without transfer on P3 using all the sources.*

(b) *Confusion Matrix without transfer on P3 using the first two sources according to the IWCV metric.*

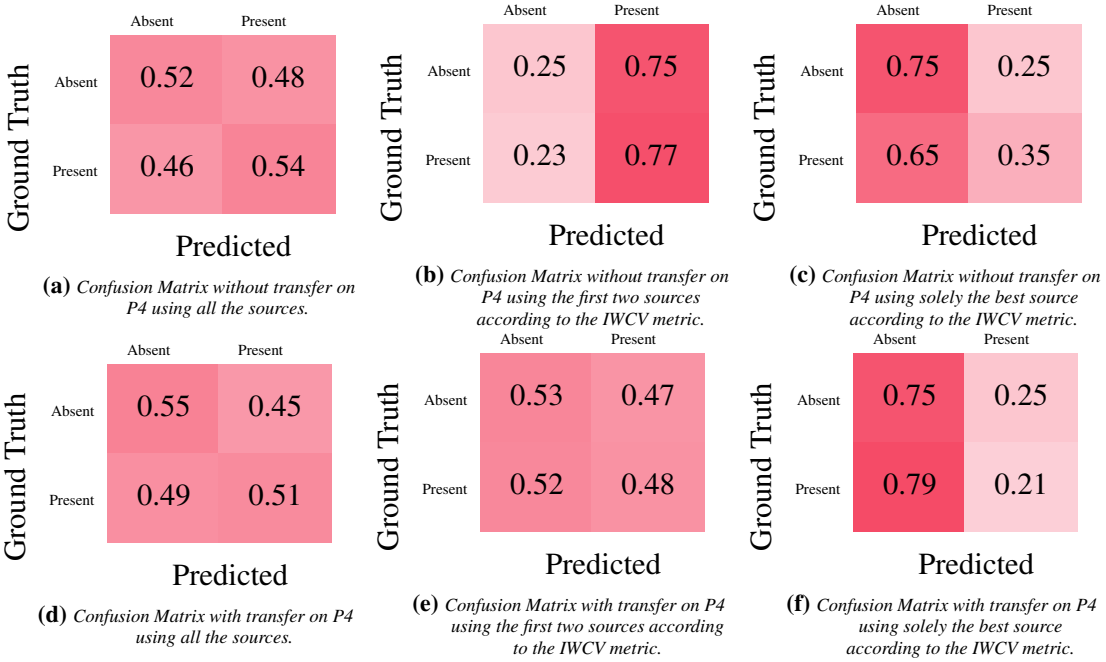
(c) *Confusion Matrix without transfer on P3 using solely the best source according to the IWCV metric.*

(d) *Confusion Matrix with transfer on P3 using all the sources.*

(e) *Confusion Matrix with transfer on P3 using the first two sources according to the IWCV metric.*

(f) *Confusion Matrix with transfer on P3 using solely the best source according to the IWCV metric.*

Figure A.3: *Binarized Confusion Matrices on P3.*



(a) *Confusion Matrix without transfer on P4 using all the sources.*

(b) *Confusion Matrix without transfer on P4 using the first two sources according to the IWCV metric.*

(c) *Confusion Matrix without transfer on P4 using solely the best source according to the IWCV metric.*

(d) *Confusion Matrix with transfer on P4 using all the sources.*

(e) *Confusion Matrix with transfer on P4 using the first two sources according to the IWCV metric.*

(f) *Confusion Matrix with transfer on P4 using solely the best source according to the IWCV metric.*

Figure A.4: *Binarized Confusion Matrices on P4.*

APPENDIX $\mathcal{B}$

Time-Variant Variational Transfer for Value Functions: Proofs

B.1 Proof of Theorem 7.2.6

Definition B.1.1. For a spatial kernel K_S: $\mu_l(K_S) = \int \theta^l K_S(\theta) d\theta$

Definition B.1.2. For a temporal kernel K_T: $a_l(-\rho) = \int_{-\rho}^{1} t^l K_T(t) dt$

Lemma B.1.1 (Estimator consistency on the right boundary)**.** *Let* $t \in B_r = \{\tau : 1 - \lambda \leq \tau \leq 1\}$ *then under assumptions of Theorem 7.2.6:*

$$\mathbb{E}[\hat{p}(\theta, t)|\mathcal{M}] = p(\theta, t) + O(\lambda) + O(tr(H)),$$

where $\mathcal{M}$ *represents all the discrete random variables* M_i *for* $i = 1 \dots n$*.*

Proof.

$$\mathbb{E}[\hat{p}(\theta, t)|\mathcal{M}] =$$

$$= \frac{1}{\bar{N}\lambda |H|^{\frac{1}{2}} a_0(-\rho)} \sum_{i=1}^{n} \int K_T\left(\frac{t-\tau}{\lambda}\right) \sum_{j=1}^{M_i} \int_{-\infty}^{+\infty} K_S\left(H^{-\frac{1}{2}}(\theta - x)\right) p(x, \tau) dx d\tau \tag{B.1}$$

$$= \frac{1}{\bar{N}\lambda \cancel{|H|^{\frac{1}{2}}} a_0(-\rho)} \sum_{i=1}^{n} \int K_T\left(\frac{t-\tau}{\lambda}\right) \sum_{j=1}^{M_i} \int_{+\infty}^{-\infty} -K_S(y) p(\theta - H^{\frac{1}{2}}y, \tau) \cancel{|H|^{\frac{1}{2}}} dy d\tau \tag{B.2}$$

$$= \frac{1}{\bar{N}\lambda a_0(-\rho)} \sum_{i=1}^{n} \int K_T\left(\frac{t-\tau}{\lambda}\right) \sum_{j=1}^{M_i} \int_{-\infty}^{+\infty} K_S(y) \Big(p(\theta,\tau) - (H^{\frac{1}{2}}y)^T \nabla^S p(\theta,\tau) + \frac{1}{2}(H^{\frac{1}{2}}y)^T \mathcal{H}^S p(\theta,\tau)(H^{\frac{1}{2}}y) + o(tr(H)) \Big) dy d\tau \tag{B.3}$$

$$= \frac{1}{\bar{N}\lambda a_0(-\rho)} \sum_{i=1}^{n} \int K_T\left(\frac{t-\tau}{\lambda}\right) M_i \Big(\int_{-\infty}^{+\infty} K_S(y) p(\theta,\tau) dy \cancel{- \int_{-\infty}^{+\infty} K_S(y)(H^{\frac{1}{2}}y)^T \nabla^S p(\theta,\tau) dy} + \int_{-\infty}^{+\infty} \frac{1}{2} K_S(y)(H^{\frac{1}{2}}y)^T \mathcal{H}^S p(\theta,\tau)(H^{\frac{1}{2}}y) dy + o(tr(H)) \Big) d\tau \tag{B.4}$$

$$= \frac{1}{\bar{N}\lambda a_0(-\rho)} \sum_{i=1}^{n} \int K_T\left(\frac{t-\tau}{\lambda}\right) M_i \Big(p(\theta,\tau) + \frac{1}{2}\mu_2(K_S) tr(H \mathcal{H}^s p(\theta,\tau)) + o(tr(H)) \Big) d\tau \tag{B.5}$$

$$= \frac{1}{\lambda a_0(-\rho)} \int_{\frac{t-1}{\lambda}}^{\frac{t}{\lambda}} K_T\left(\frac{t-\tau}{\lambda}\right) \Big(p(\theta,\tau) + O(tr(H)) \Big) d\tau \tag{B.6}$$

$$= \frac{1}{\lambda a_0(-\rho)} \left(\int_{-\rho}^{1} K_T\left(\frac{t-\tau}{\lambda}\right) p(\theta,\tau) d\tau + O(tr(H)) \int_{-\rho}^{1} K_T\left(\frac{t-\tau}{\lambda}\right) d\tau \right) \tag{B.7}$$

$$= \frac{\cancel{\lambda}}{\cancel{\lambda} a_0(-\rho)} \left(- \int_{1}^{-\rho} K_T(v) p(\theta, t - \lambda v) dv - O(tr(H)) \int_{1}^{-\rho} K_T(v) dv \right) \tag{B.8}$$

$$= \frac{1}{a_0(-\rho)} \Big(\int_{-\rho}^{1} K_T(v) \Big(p(\theta,t) - \lambda v p'(\theta,t) + \frac{1}{2}\lambda^2 v^2 p''(\theta,t) + o(\lambda^2) \Big) dv + O(tr(H)) \Big) \tag{B.9}$$

$$= p(\theta,t) - \lambda p'(\theta,t) \frac{a_1(-\rho)}{a_0(-\rho)} + O(\lambda^2) + O(tr(H)) \tag{B.10}$$

$$= p(\theta,t) + O(\lambda) + O(tr(H)), \tag{B.11}$$

where in (B.2) we performed a change of variable, $y = H^{-\frac{1}{2}}(\theta - x)$, in (B.3) we used the following Taylor expansion:

$$p(\theta - H^{\frac{1}{2}}y, \tau) = p(\theta, \tau) - (H^{\frac{1}{2}}y)^T \nabla^S p(\theta, \tau) + \frac{1}{2}(H^{\frac{1}{2}}y)^T \mathcal{H}^S p(\theta, \tau)(H^{\frac{1}{2}}y) + o(tr(H)),$$

in (B.4) we used Assumption 7.2.4,in (B.5) we used Definition B.1.1, in (B.6) we used $\frac{t-\tau}{\lambda} \in [\frac{t-1}{\lambda}, \frac{t}{\lambda})$, in (B.7) we set $t = 1 - \rho\lambda$, which implies $\frac{t-\tau}{\lambda} \in [-\rho, \frac{1}{\lambda} - \rho)$, then we used the support of K_T (assumed to be $[-1, 1]$ without loss of generality) since $\lambda \to 0$. Finally, in (B.8) we used a change of variable, $\frac{t-\tau}{\lambda} = v$, and in (B.9) we used the following Taylor expansion:

$$p(\theta, t - \lambda v) = p(\theta, t) - \lambda v p'(\theta, t) + \frac{1}{2}\lambda^2 v^2 p''(\theta, v) + o(\lambda^2).$$

□

Notice that we reported the consistency proof only on the right boundary because is the one we use in the context of our algorithm. The above procedure can be easily adjusted to prove consistency of the estimator on the left boundary getting the same convergence rate. Moreover, analogously, we can obtain consistency away from the two boundaries with a convergence rate squared w.r.t. λ.

Definition B.1.3. For a spatial kernel K_S: $R(K_S) = \int K_S^2(\theta)d\theta$

Definition B.1.4. For a temporal kernel K_T: $b_{K_T}(-\rho) = \int_{-\rho}^{1} K_T^2(t)dt$

Lemma B.1.2 (Variance of the estimator on the right boundary)**.** *Let $t \in B_r = \{\tau : 1 - \lambda \leq \tau \leq 1\}$ then under assumptions of Theorem 7.2.6:*

$$\mathbb{V}ar[\hat{p}(\theta, t)|\mathcal{M}] \leq \frac{C_1}{\bar{N}|H|^{\frac{1}{2}}\lambda},$$

where $\mathcal{M}$ represents all the discrete random variables M_i for $i = 1 \dots n$.

Proof.

$$\mathbb{V}ar[\hat{p}(\theta, t)|\mathcal{M}] = \frac{1}{\bar{N}a_0^2(-\rho)}\mathbb{V}ar\left[\frac{1}{|H|^{\frac{1}{2}}\lambda}K_T\left(\frac{t - t_i}{\lambda}\right)K_S\left(H^{-\frac{1}{2}}(\theta - x_{ij})\right)\right] \quad \text{(B.12)}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\left(\mathbb{E}\left[\frac{1}{|H|\lambda^2}K_T^2\left(\frac{t - t_i}{\lambda}\right)K_S^2\left(H^{-\frac{1}{2}}(\theta - x_{ij})\right)\right] - \right.$$
$$\left. \mathbb{E}^2\left[\frac{1}{|H|^{\frac{1}{2}}\lambda}K_T\left(\frac{t - t_i}{\lambda}\right)K_S\left(H^{-\frac{1}{2}}(\theta - x_{ij})\right)\right]\right) \quad \text{(B.13)}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int \frac{1}{|H|\lambda^2} K_T^2\left(\frac{t-\tau}{\lambda}\right) \int_{+\infty}^{-\infty} -|H|^{\frac{1}{2}} K_S^2(y) p(\theta - H^{\frac{1}{2}}y, \tau) dy d\tau - \left(\int \frac{1}{|H|^{\frac{1}{2}}\lambda} K_T\left(\frac{t-\tau}{\lambda}\right) \int_{+\infty}^{-\infty} -|H|^{\frac{1}{2}} K_S(y) p(\theta - H^{\frac{1}{2}}y, \tau) dy d\tau \right)^2 \Bigg) \tag{B.14}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int \frac{1}{|H|^{\frac{1}{2}}\lambda^2} K_T^2\left(\frac{t-\tau}{\lambda}\right) \int_{-\infty}^{+\infty} K_S^2(y) \left(p(\theta,\tau) + o(1)\right) dy d\tau - \left(\int \frac{1}{\lambda} K_T\left(\frac{t-\tau}{\lambda}\right) \int_{-\infty}^{+\infty} K_S(y) \left(p(\theta,\tau) + o(1)\right) dy d\tau \right)^2 \Bigg) \tag{B.15}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int \frac{1}{|H|^{\frac{1}{2}}\lambda^2} K_T^2\left(\frac{t-\tau}{\lambda}\right) \left(p(\theta,\tau) + o(1)\right) R(K_S) d\tau - \left(\int \frac{1}{\lambda} K_T\left(\frac{t-\tau}{\lambda}\right) \left(p(\theta,\tau) + o(1)\right) d\tau \right)^2 \Bigg) \tag{B.16}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int_{-\rho}^{1} \frac{1}{|H|^{\frac{1}{2}}\lambda^2} K_T^2\left(\frac{t-\tau}{\lambda}\right) \left(p(\theta,\tau) + o(1)\right) R(K_S) d\tau - \left(\int_{-\rho}^{1} \frac{1}{\lambda} K_T\left(\frac{t-\tau}{\lambda}\right) \left(p(\theta,\tau) + o(1)\right) d\tau \right)^2 \Bigg) \tag{B.17}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int_{1}^{-\rho} -\frac{1}{|H|^{\frac{1}{2}}\lambda} K_T^2(v) \left(p(\theta, t-\lambda v) + o(1)\right) R(K_S) dv - \left(\int_{1}^{-\rho} -K_T(v) \left(p(\theta, t-\lambda v) + o(1)\right) dv \right)^2 \Bigg) \tag{B.18}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)}\Bigg(\int_{-\rho}^{1} \frac{1}{|H|^{\frac{1}{2}}\lambda} K_T^2(v) \left(p(\theta, t) + o(1)\right) R(K_S) dv - \left(\int_{-\rho}^{1} K_T(v) \left(p(\theta, t) + o(1)\right) dv \right)^2 \Bigg) \tag{B.19}$$

$$= \frac{1}{\bar{N}a_0^2(-\rho)} \left(\frac{p(\theta,t) + o(1)}{|H|^{\frac{1}{2}}\lambda} R(K_S) b_{K_T}(-\rho) - \left(a_0(-\rho)(p(\theta,t) + o(1))\right)^2 \right) \tag{B.20}$$

$$= \frac{p(\theta,t) R(K_S) b_{K_T}(-\rho)}{\bar{N}|H|^{\frac{1}{2}}\lambda a_0^2(-\rho)} + O\left(\frac{1}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right) \tag{B.21}$$

$$= O\left(\frac{1}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right) \quad \rightarrow \quad \exists\, C_1 : \mathbb{V}ar[\hat{p}(\theta,t)|\mathcal{M}] \leq \frac{C_1}{\bar{N}|H|^{\frac{1}{2}}\lambda}, \tag{B.22}$$

where in (B.14) we performed a change of variable, $y = H^{-\frac{1}{2}}(\theta - x)$, in (B.15) we used the following Taylor expansion:

$$p(\theta - H^{\frac{1}{2}}y, \tau) = p(\theta, \tau) + o(1),$$

in (B.16) we used Definition B.1.3, in (B.17) we considered the fact that $t \in B_r$ as we have done in (B.6) and (B.7) of the proof of B.1.1, in (B.18) we performed a change of variable, $\frac{t-\tau}{\lambda} = v$, in (B.19) we used the following Taylor expansion:

$$p(\theta, t - \lambda v) = p(\theta, t) + o(1),$$

whereas in (B.20) we have used Definition B.1.4. Finally, in (B.22) we have used the fact that $p(\theta, t)$ has bounded derivatives and is a pdf, therefore it has finite supremum. □

Lemma B.1.3 (Bound on the absolute values)**.** *Let $t \in B_r = \{\tau : 1 - \lambda \leq \tau \leq 1\}$ then under assumptions of Theorem 7.2.6: $\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]$ is the sum of $\bar{N}$ independent random variables, denoted as v_i, with zero mean and absolute values bounded by $\frac{C_2}{\bar{N}|H|^{\frac{1}{2}}\lambda}$. $\mathcal{M}$ represents all the discrete random variables M_i for $i = 1 \dots n$.*

Proof.

$$\begin{aligned} |v_i| = \Bigg| & \frac{1}{\bar{N}\lambda|H|^{\frac{1}{2}}a_0(-\rho)} K_T\left(\frac{t - t_i}{\lambda}\right) K_S(H^{-\frac{1}{2}}(\theta - x_{ij})) - \\ & \frac{p(\theta,t) + O(\lambda) + O(tr(H))}{\bar{N}} \Bigg| \end{aligned} \tag{B.23}$$

$$\leq \left| \frac{M_T M_S}{\bar{N}\lambda|H|^{\frac{1}{2}}a_0(-\rho)} - \frac{p(\theta,t) + O(\lambda) + O(tr(H))}{\bar{N}} \right| \tag{B.24}$$

$$= O\left(\frac{1}{\bar{N}\lambda|H|^{\frac{1}{2}}}\right) \rightarrow \exists C_2 : \ |v_i| \leq \frac{C_2}{\bar{N}|H|^{\frac{1}{2}}\lambda}, \tag{B.25}$$

where in (B.23) we used *lemma* B.1.1 and in (B.24) we used the fact that K_T has a compact support on $\mathbb{R}$ and K_S has a supremum. □

Now the proof of *Theorem* 7.2.6 can follow.

Proof. Let $\xi = C\left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right)^{\frac{1}{2}}$ and $C_3 = \frac{1}{\max\left(C_1, \frac{C_2}{3}\right)}$, using Bernstein's inequality we can write:

$$\mathbb{P}\left(|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > \xi|\mathcal{M}\right) \leq 2\exp\left(-\frac{\frac{1}{2}\xi^2}{\frac{C_1}{\bar{N}|H|^{\frac{1}{2}}\lambda} + \frac{1}{3}\frac{C_2\xi}{\bar{N}|H|^{\frac{1}{2}}\lambda}}\right) \tag{B.26}$$

$$= 2\exp\left(-\frac{\frac{1}{2}C^2\log n}{C_1 + \frac{1}{3}C_2\xi}\right) \leq 2\exp\left(-\frac{C_3C^2\log n}{1+\xi}\right), \forall(\theta,t). \tag{B.27}$$

Therefore, if $C_4 > 0$ is given, and we choose $C^2 > \frac{3C_4}{C_3}$, then we can write:

$$\sup_{(\theta,t)\,\in\,\mathbb{R}^p\times\mathcal{I}} \mathbb{P}\left(|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C\left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right)^{\frac{1}{2}}\right) \leq 2\exp\left(-\frac{3C_4\log n}{1+\xi}\right) \tag{B.28}$$

$$= 2n^{-\frac{3C_4}{1+\xi}}. \tag{B.29}$$

Now restricting to finite subsets $\mathcal{K}_n \subset \mathcal{K} \subset \mathbb{R}^p$ and $\mathcal{I}_n \subset \mathcal{I}$ where $\mathcal{K}_n \times \mathcal{I}_n$ has at most $\lfloor n^{\frac{2C_4}{1+\xi}} \rfloor$ elements, we have:

$$\mathbb{P}\left(\sup_{(\theta,t)\,\in\,\mathcal{K}_n\times\mathcal{I}_n} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C\left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right)^{\frac{1}{2}}\right) \leq 2n^{\frac{-C_4}{1+\xi}}. \tag{B.30}$$

From the H*ölder*-continuity of the estimator (since the two kernels have bounded first derivative):

$$\sup_{(\theta,t)\,\in\,\mathcal{K}\times\mathcal{I}} \{|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]|\} - \sup_{(\theta,t)\,\in\,\mathcal{K}_n\times\mathcal{I}_n} \{|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]|\} =$$

$$\left|\sup_{(\theta,t)\,\in\,\mathcal{K}\times\mathcal{I}} \{|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]|\} - \sup_{(\theta,t)\,\in\,\mathcal{K}_n\times\mathcal{I}_n} \{|\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]|\}\right| \leq$$

$$\left|\sup_{(\theta,t)\,\in\,\mathcal{K}\times\mathcal{I}} \{\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]\} - \sup_{(\theta,t)\,\in\,\mathcal{K}_n\times\mathcal{I}_n} \{\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]\}\right| \leq$$

$$D||v^* - v_n^*||^{\alpha}, \tag{B.31}$$

where

$$v^* = \arg\sup_{(\theta,t)\,\in\,\mathcal{K}\times\mathcal{I}} \{\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]\}$$

$$v_n^* = \arg \sup_{(\theta,t) \in \mathcal{K}_n \times \mathcal{I}_n} \{\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]\},$$

therefore:

$$\mathbb{P}\left(\sup_{(\theta,t) \in \mathcal{K} \times \mathcal{I}} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C \left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda} \right)^{\frac{1}{2}} + D||v^* - v_n^*||^\alpha \right) \leq$$

$$\mathbb{P}\left(\sup_{(\theta,t) \in \mathcal{K}_n \times \mathcal{I}_n} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C \left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda} \right)^{\frac{1}{2}} \right) \tag{B.32}$$

now, for sufficiently large C_4, $||v^* - v_n^*|| \leq \frac{\sqrt{p+1}}{2} \sqrt[p+1]{\frac{(K_{max}-K_{min})^p(I_{max}-I_{min})}{\left\lfloor n^{\frac{2C_4}{1+\xi}} \right\rfloor}}$ and

$D\left(\frac{\sqrt{p+1}}{2} \sqrt[p+1]{\frac{(K_{max}-K_{min})^p(I_{max}-I_{min})}{\left\lfloor n^{\frac{2C_4}{1+\xi}} \right\rfloor}} \right)^\alpha$ is negligible w.r.t. ξ as n tends to infinity, where K_{max} e K_{min} are the endpoints for each dimension of $\mathcal{K}$ (we assume them to be the same in each dimension for the sake of simplicity). Analogously for I_{max} and I_{min} (notice that $\mathcal{I}$ is mono-dimensional). Therefore:

$$\mathbb{P}\Bigg(\sup_{(\theta,t) \in \mathcal{K} \times \mathcal{I}} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C \left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda} \right)^{\frac{1}{2}} +$$

$$D\left(\frac{p+1}{4} \right)^{\frac{\alpha}{2}} \left(\frac{(K_{max} - K_{min})^p (I_{max} - I_{min})}{\left\lfloor n^{\frac{2C_4}{1+\xi}} \right\rfloor} \right)^{\frac{\alpha}{p+1}} \Bigg) \leq$$

$$\mathbb{P}\left(\sup_{(\theta,t) \in \mathcal{K}_n \times \mathcal{I}_n} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| > C \left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda} \right)^{\frac{1}{2}} \right). \tag{B.33}$$

From (B.30) and (B.33), we can write:

$$\mathbb{P}\Bigg(\sup_{(\theta,t) \in \mathcal{K} \times \mathcal{I}} |\hat{p}(\theta,t) - \mathbb{E}[\hat{p}(\theta,t)|\mathcal{M}]| < C \left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda} \right)^{\frac{1}{2}} +$$

$$D\left(\frac{p+1}{4} \right)^{\frac{\alpha}{2}} \left(\frac{(K_{max} - K_{min})^p (I_{max} - I_{min})}{\left\lfloor n^{\frac{2C_4}{1+\xi}} \right\rfloor} \right)^{\frac{\alpha}{p+1}} \Bigg) \geq 1 - 2n^{\frac{-C_4}{1+\xi}} \tag{B.34}$$

Therefore, as $n \to \infty$ with probability 1:

$$|\hat{p}(\theta,t) - p(\theta,t) - O(\lambda) - O(tr(|H|))| = O\Bigg[C\left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right)^{\frac{1}{2}} + D\left(\frac{p+1}{4}\right)^{\frac{\alpha}{2}}\left(\frac{(K_{max}-K_{min})^p(I_{max}-I_{min})}{\left\lfloor n^{\frac{2C_4}{1+\xi}}\right\rfloor}\right)^{\frac{\alpha}{p+1}}\Bigg], \forall\,(\theta,t)\ \in\ \mathcal{K}\times\mathcal{I} \tag{B.35}$$

Finally, we get:

$$\hat{p}(\theta,t) = p(\theta,t) + O\left[\left(\frac{\log n}{\bar{N}|H|^{\frac{1}{2}}\lambda}\right)^{\frac{1}{2}} + \lambda + tr(H)\right], \forall\,(\theta,t)\ \in\ \mathcal{K}\times\mathcal{I} \tag{B.36}$$

□

B.2 Upper Bound on the KL-Divergence Between the Prior and the Posterior

In this section, we report the steps needed to get an upper bound on the KL-Divergence between the posterior q our prior $\hat{p}$. Let us define $S = \frac{1}{a_0(-\rho)\bar{N}\lambda}\sum_{i=1}^{n}\sum_{j=1}^{M_i}K_T(\frac{t-t_i}{\lambda})$, hence:

$$D_{KL}(q||\hat{p}(\cdot,t)) = \int q(\theta)log\frac{q(\theta)}{\hat{p}(\theta,t)}d\theta = \int q(\theta)log\frac{q(\theta)}{\frac{S}{S}\hat{p}(\theta,t)}d\theta \tag{B.37}$$

$$= \int q(\theta)log\frac{q(\theta)}{\frac{1}{Sa_0(-\rho)\bar{N}|H|^{\frac{1}{2}}\lambda}\sum_{i=1}^{n}K_T(\frac{t-t_i}{\lambda})\sum_{j=1}^{M_i}K_S(H^{-\frac{1}{2}}(\theta-\theta_{ij}))}d\theta + \int q(\theta)\log\frac{1}{S}d\theta \tag{B.38}$$

Now the first term in Equation (B.38) is the KL-Divergence between two Mixture of Gaussians, which can be upper bounded using the same procedure as in Hershey and Olsen [2007], and the second term is a constant in the ELBO optimization. Therefore:

$$D_{KL}(q||\hat{p}(\cdot,t)) \leq D_{KL}(\chi^{(2)}||\chi^{(1)}) + \log\frac{1}{S} + \sum_{i,j}\chi^{(2)}_{j,i}D_{KL}(f^q_i||f^{\hat{p}}_j), \tag{B.39}$$

where we are rewriting $q = \sum_i c^q_i f^q_i$ and $\hat{p} = \sum_j c^{\hat{p}}_j f^{\hat{p}}_j$ with c^y_x being a generic weight and $f^y_x = \mathcal{N}(\mu^y_x, \Sigma^y_x)$ being a generic component, $(x,y) \in \{(i,q),(j,\hat{p})\}$. Furthermore, we

have:

$$\chi^{(1)}_{i,j} = \frac{c^{\hat{p}}_j \chi^{(2)}_{j,i}}{\sum_{i'} \chi^{(2)}_{j,i'}}, \qquad \chi^{(2)}_{j,i} = \frac{c^{(q)}_i \chi^{(1)}_{i,j} e^{-D_{KL}(f^q_i || f^{\hat{p}}_j)}}{\sum_{j'} \chi^{(1)}_{i,j'} e^{-D_{KL}(f^q_i || f^{\hat{p}}_{j'})}}. \tag{B.40}$$

Finally, notice that $c^q_i = \frac{1}{C}$ for each i, where C is the number of components for the posterior, whereas $c^{\hat{p}}_j = \frac{1}{Sa_0(-\rho)\bar{N}\lambda} K_T(\frac{t-t_i}{\lambda})$, with a little abuse of notation over the index i and j.

B.3 Proof of Theorem 7.3.1

The proof of Theorem 7.3.1 is straightforward, we just need to follow the same procedure of Tirinzoni et al. [2018a] plugging in the bound on the KL-Divergence of Equation (B.39). In the following we report the proof for completeness.

Proof. We start from Lemma 2 of Tirinzoni et al. [2018a] with variational parameter $\hat{\xi} = (\hat{\mu}_1, \dots, \hat{\mu}_C, \hat{\Sigma}_1, \dots, \hat{\Sigma}_C)$, whereas, for the right-hand side, we set $\mu_i = \theta^*$ and $\Sigma_i = cI$ for each $i = 1, \dots, C$, for some $c > 0$:

$$\begin{aligned}
\mathbb{E}_{q_{\hat{\xi}}}\left[\left|\left|\tilde{B}_\theta\right|\right|^2_\nu\right] &\leq \inf_{\xi \in \Xi}\left\{\mathbb{E}_{q_\xi}\left[\left|\left|\tilde{B}_\theta\right|\right|^2_\nu\right] + \mathbb{E}_{q_{\hat{\xi}}}\left[v(\theta)\right] + 2\frac{\psi}{N} D_{KL}(q_\xi || \hat{p})\right\} + 8\frac{R^2_{max}}{(1-\gamma)^2}\sqrt{\frac{\log\frac{2}{\delta}}{2N}} \\
&\leq \mathbb{E}_{\mathcal{N}(\theta^*, cI)}\left[\left|\left|\tilde{B}_\theta\right|\right|^2_\nu\right] + \mathbb{E}_{\mathcal{N}(\theta^*, cI)}\left[v(\theta)\right] + 2\frac{\psi}{N} D_{KL}(\mathcal{N}(\theta^*, cI) || \hat{p}) + \\
&\quad 8\frac{R^2_{max}}{(1-\gamma)^2}\sqrt{\frac{\log\frac{2}{\delta}}{2N}}.
\end{aligned} \tag{B.41}$$

From Appendix B.2 we have:

$$\begin{aligned}
&D_{KL}(\mathcal{N}(\theta^*, cI) || \hat{p}) \leq \\
&\qquad D_{KL}(\chi^{(2)} || \chi^{(1)}) + \log\frac{1}{S} + \sum_j \chi^{(2)}_j D_{KL}(\mathcal{N}(\theta^*, cI) || \mathcal{N}(\theta_j, \sigma^2 I)),
\end{aligned} \tag{B.42}$$

where

$$\chi^{(1)}_j = c^{\hat{p}}_j, \qquad \chi^{(2)}_j = \frac{c^{\hat{p}}_j e^{-D_{KL}(\mathcal{N}(\theta^*, cI) || \mathcal{N}(\theta_j, \sigma^2 I))}}{\sum_{j'} c^{\hat{p}}_{j'} e^{-D_{KL}(\mathcal{N}(\theta^*, cI) || \mathcal{N}(\theta_{j'}, \sigma^2 I))}} \tag{B.43}$$

obtained noticing that we can remove the index i because we have reduced the posterior to one component. $\chi_j^{(2)}$ can be rewritten:

$$\chi_j^{(2)} = \frac{c_j^{\hat{p}} e^{-\frac{1}{2\sigma^2}||\theta^* - \theta_j||}}{\sum_{j'} c_{j'}^{\hat{p}} e^{-\frac{1}{2\sigma^2}||\theta^* - \theta_{j'}||}} \tag{B.44}$$

if we plug in the closed form expression of the KL-Divergence (B.45) into its definition.

$$D_{KL}(\mathcal{N}(\theta^*, cI)||\mathcal{N}(\theta_j, \sigma^2 I)) = \frac{1}{2}\left(p\log\frac{\sigma^2}{c} + p\frac{c}{\sigma^2} + \frac{||\theta^* - \theta_j||}{\sigma^2} - p\right). \tag{B.45}$$

Now we proceed upper bounding the first and then the third term of (B.42):

$$D_{KL}(\chi^{(2)}||\chi^{(1)}) = \sum_j \chi_j^{(2)} \log\frac{\chi_j^{(2)}}{\chi_j^{(1)}} \tag{B.46}$$

$$= \sum_j \chi_j^{(2)} \log \chi_j^{(2)} - \sum_j \chi_j^{(2)} \log \chi_j^{(1)} \tag{B.47}$$

$$\leq \sum_j \chi_j^{(2)} \log\frac{1}{c_j^{\hat{p}}} \tag{B.48}$$

where we got (B.48) just noticing in (B.47) that the first term is negative. Considering the third term, we have:

$$\begin{aligned}\sum_j \chi_j^{(2)} D_{KL}(\mathcal{N}(\theta^*, cI)||\mathcal{N}(\theta_j, \sigma^2 I)) &= \frac{1}{2}\sum_j \chi_j^{(2)}\left(p\log\frac{\sigma^2}{c} + p\frac{c}{\sigma^2} + \frac{||\theta^* - \theta_j||}{\sigma^2} - p\right) \\ &\leq \frac{1}{2}p\log\frac{\sigma^2}{c} + \frac{1}{2}p\frac{c}{\sigma^2} + \sum_j \chi_j^{(2)}\frac{||\theta^* - \theta_j||}{2\sigma^2}.\end{aligned} \tag{B.49}$$

Therefore:

$$\begin{aligned}&D_{KL}(\mathcal{N}(\theta^*, cI)||\hat{p}) \leq \\ &\qquad \sum_j \chi_j^{(2)} \log\frac{1}{c_j^{\hat{p}}} + \log\frac{1}{S} + \frac{1}{2}p\log\frac{\sigma^2}{c} + \frac{1}{2}p\frac{c}{\sigma^2} + \sum_j \chi_j^{(2)}\frac{||\theta^* - \theta_j||}{2\sigma^2}.\end{aligned} \tag{B.50}$$

Now leveraging the above equation, the following upper bound obtained in the proof of Theorem 3 in Tirinzoni et al. [2018a]:

$$\mathbb{E}_{\mathcal{N}(\theta^*, cI)}\left[\left|\left|\tilde{B}_\theta\right|\right|_\nu^2\right] \leq 2\left|\left|\tilde{B}_{\theta^*}\right|\right|_\nu^2 + \frac{1}{2}\gamma^2\kappa^2 c^2 \phi_{max}^4 + c(\theta_{max}\phi_{max}(1+\gamma))^2, \tag{B.51}$$

and setting c=$\frac{1}{N}$ (since the bound hold for any constant parameter $c > 0$), $c_1 = \frac{8R_{max}^2}{\sqrt{2}(1-\gamma)^2}$, $c_2 = \theta_{max}^2 \phi_{max}^2 (1-\gamma)^2 + \psi p \log \sigma^2 + 2\psi \sum_j \chi_j^{(2)} \log \frac{1}{c_j^p} + 2\psi \log \frac{1}{S}$, $c_3 = \frac{1}{2}\gamma^2 \kappa^2 \phi_{max}^4 + \frac{\psi p}{\sigma^2}$ and $\varphi(\Theta_s) = \frac{1}{\sigma^2} \sum_j \chi_j^{(2)} ||\theta^* - \theta_j||$, we can rewrite Equation (B.41) in the following way:

$$\mathbb{E}_{q_{\hat{\xi}}} \left[\left| \left| \tilde{B}_\theta \right| \right|_\nu^2 \right] \leq 2 \left| \left| \tilde{B}_{\theta^*} \right| \right|_\nu^2 + v(\theta^*) + c_1 \sqrt{\frac{\log \frac{2}{\delta}}{N}} + \frac{c_2 + \psi p \log N + \psi \varphi(\Theta_s)}{N} + \frac{c_3}{N} \tag{B.52}$$

□

B.4 Experimental Details

In this section, we provide some additional experimental details together with further results.

B.4.1 Parametrization

ADAM [Kingma and Ba, 2014] is used in every experiment as optimizer. The source tasks are solved by a direct minimization of the TD error as described in section 3.4 of Tirinzoni et al. [2018a], using a *batch size* of 50 for the rooms environments and of 32 for Mountain Car and the lake Como water system, a *buffer size* of 50000, the projection parameter of the mellow-max TD error gradient set to 0.5, the learning rate $\alpha = 10^{-3}$. The exploration is ϵ-greedy with ϵ linearly decaying from 1 to 0.01 for Mountain Car and to 0.02 for the rooms environments. Both decays happen within 50% of the maximum number of learning iterations. In the lake Como environment we used a soft-max (Gibbs) policy with parameter β linearly increasing from 0.5 to 9.275 through the learning iterations.

In the **rooms** environments, for what concern the two transfer algorithms, c-T2VT, and c-MGVT, we have the following parametrization: *batch size* of 50, *buffer size* of 50000, projection parameter of the mellow-max TD error gradient set to 0.5 (see section 3.4 of Tirinzoni et al. [2018a]), the parameter of Equation (7.2) $\psi = 10^{-6}$, 10 weights to estimate the expected TD error, the learning rates are set to $\alpha_\mu = 10^{-3}$ and $\alpha_L = 0.1$ for the mean and the Cholesky factor L of the posterior (moreover, the minimum eigenvalue reachable by L is set to $\sigma_{min}^2 = 10^{-4}$). Finally, for the prior, we use a diagonal isotropic matrix $H = 10^{-5} I$ and $\lambda = 0.3333$ in the context of c-T2VT, furthermore, we have $\Sigma = 10^{-5} I$ for the prior in the context of c-MGVT.

In the **Mountain Car** environment, c-T2VT and c-MGVT are parametrized in the following way: *batch size* of 500, *buffer size* of 10000, projection parameter of the mellow-max TD error gradient set to 0.5, the parameter of Equation (7.2) $\psi = 10^{-4}$, 10 weights to estimate the expected TD error, the learning rates are set to $\alpha_\mu = 10^{-3}$ and $\alpha_L = 10^{-4}$

for the mean and the Cholesky factor L of the posterior (moreover, the minimum eigenvalue reachable by L is set to $\sigma^2_{min} = 10^{-4}$). Finally, for the prior, we use a diagonal isotropic matrix $H = 10^{-5}I$ and $\lambda = 0.3333$ in the context of c-T2VT, furthermore, we have $\Sigma = 10^{-5}I$ for the prior in the context of c-MGVT.

In the **lake Como** water system, 3-T2VT and 3-MGVT are parametrized in the following way: *batch size* of 32, *buffer size* of 10000, projection parameter of the mellow-max TD error gradient set to 0.5, the parameter of Equation (7.2) $\psi = 10^{-4}$, 4 weights to estimate the expected TD error, the learning rates are set to $\alpha_\mu = 10^{-3}$ and $\alpha_L = 10^{-4}$ for the mean and the Cholesky factor L of the posterior (moreover, the minimum eigenvalue reachable by L is set to $\sigma^2_{min} = 10^{-4}$). Finally, for the prior, we use a diagonal isotropic matrix $H = 10^{-5}I$ and λ was chosen through the maximum-likelihood approach of Section 7.5.5 in the context of 3-T2VT, furthermore, we have $\Sigma = 10^{-5}I$ for the prior in the context of 3-MGVT.

B.4.2 Temporal Dynamics

In this section, we provide the analytical form of the different dynamics employed in our experiments. Notice that these dynamics need to be plugged into the mean of our Gaussian distribution from where we sample the parametrization defining the task (for the rooms environment we will sample the positions of the doors, whereas, for the Mountain Car environment, we will sample the base speed).

- **Linear**: $2t - 1,\ t \in [0, 1]$;
- **Polynomial**: $at^4 + bt^3 + ct^2 + dt + e,\ t \in [0, 1]$ and $a = -15.625$, $b = 39.5833$, $c = -31.875$, $d = 9.91667$ and $e = -1$;

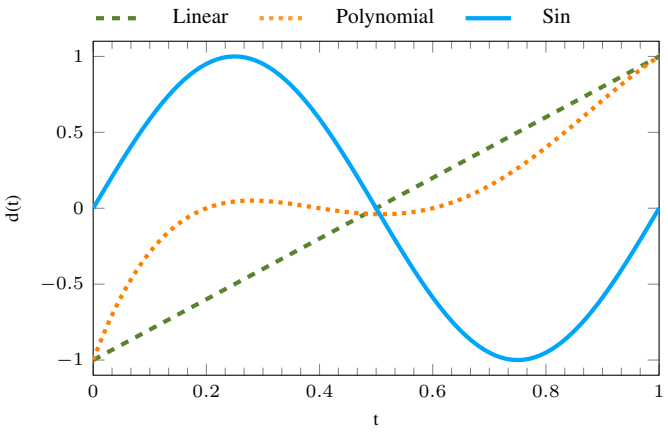


Figure B.1: *Temporal dynamics.*

- **Sinusoidal**: $\sin(2\pi t),\ t \in [0, 1]$.

In Figure B.1, we report the graphical representation of the above analytical functions.

Now, given the range for a parameter $[k_{min}, k_{max}]$, a given dynamic will span over this interval in the following way: $d(t)\frac{(k_{max}-k_{min})}{2} + \frac{(k_{max}+k_{min})}{2}$. Finally, notice that, $[k_{min}, k_{max}] = [0.001, 0.0015]$ for Mountain Car, whereas $[k_{min}, k_{max}] = [0.7+padding, 9.3-padding]$ for the parameters of the rooms environments. The $padding$ variable is 0 for the 2-rooms, whereas is 2 for the 3-rooms environments. This $padding$ variable was necessary in the 3-rooms environments in order for the TD gradient algorithm to be able to solve the source tasks in every configuration of the two doors.

B.4.3 λ-Sensitivity Results

In Figures B.2 and B.3, we report a sensitivity analysis of our algorithm w.r.t. λ in the 2-rooms environment. This analysis is carried out computing the performance of the learning algorithm w.r.t. different values of the previously mentioned parameter (whereas $H = 10^{-5}I$ for every λ). These results are also compared with the performance of the algorithm when λ is chosen according to the likelihood optimization described in Section 7.5.5. In Figures B.4 and B.5, we report the above-described analysis in the context of the 3-rooms environment, whereas, in Figures B.6 and B.7, we have the Mountain Car environment.

In the context of both rooms environments, the performance of the likelihood approach is satisfying, for both 1-T2VT and 3-T2VT, even though in some cases it is not optimal. For what concern, the polynomial dynamic this may be due to its plateau (see Figure B.1) which bias the choice for λ toward bigger values since the likelihood is evaluated in a cross-validation manner. For the same reason, in the $\sin$ dynamic case, the likelihood-based approach tends to select an average λ. Finally, the linear case in the 2-rooms is almost optimal, whereas, in the 3-rooms, the performance decreases. This is due to the fact that, in the 3-rooms environment, we have 2 parameters governing the dynamics (the two doors positions) making the choice of λ harder to make in this setting.

In the context of the Mountain Car environment the likelihood approach always choose the best λ as shown in Figures B.6 and B.7.

Implementation Details: since the $\lambda \in [0, 1]$, we performed a grid search in order to optimize Equation (7.5) .

B.4.4 Further Environmental Settings: Mountain Car and Lake Como Water System

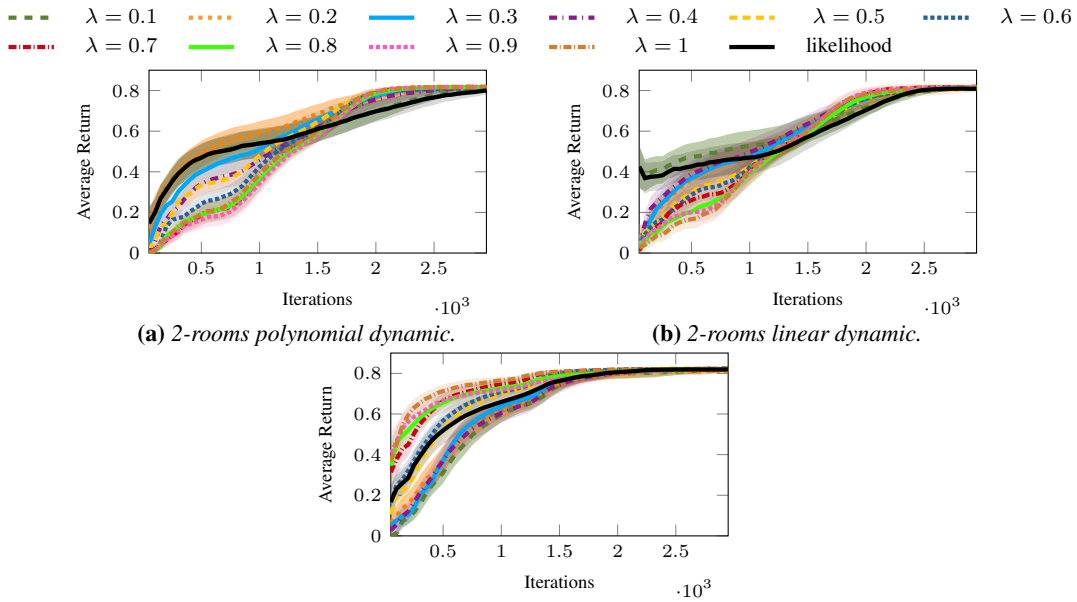
Mountain Car

The state space consists in the *position* and *velocity* of the car. The reward function is always -1 so the agent must reach the goal as soon as possible. The available actions are *backward full throttle*, *zero throttle* and *forward full throttle* encoded as $[-1, 0, 1]$. The

discount factor is $\gamma = 0.99$. The goal position is 0.5. Finally, the transition function is $position_{t+1} = position_t + velocity_{t+1}$, $velocity_{t+1} = velocity_t + a_t * 0.001 - 0.0025 * \cos(3 * position_t)$. The *velocity* is clipped whenever exits the range $[-0.07, 0.07]$ the *position* is bound to lie in $[-1.2, 0.6]$.

Lake Como

The reward function in the lake Como water system is composed of three main costs. The demand cost is a squared function of the discrepancy between actual release and water demand: $-4(\varrho_{t+1} - demand_t)^2$ if t is between may and august, otherwise $-(\varrho_{t+1} - demand_t)^2$. The flooding cost is a constant penalty inflicted to the agent whenever a water level flooding threshold is broken: $-1\ if\ water\ level > 1.24\ else\ 0$. Finally, the unfeasibility penalty is just a discrepancy between the action requested by the agent and the actual release the system was able to accomplish: $-|a_t - \varrho_{t+1}|$. Each component is rescaled in $[-1, 0]$ and contribute uniformly for $\frac{1}{3}$ to the reward function. The actions available to the agent are 8 different amount of water to be released: *[0, 79.39, 88.10, 110.39, 148.39, 200.13, 225.25, 491.61]*. The discount factor is $\gamma = 0.9999$.

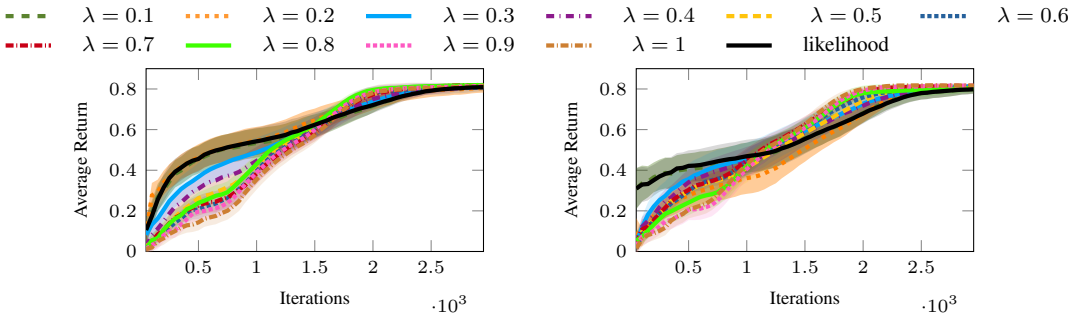


(a) *2-rooms polynomial dynamic.*

(b) *2-rooms linear dynamic.*

(c) *2-rooms* sin *dynamic.*

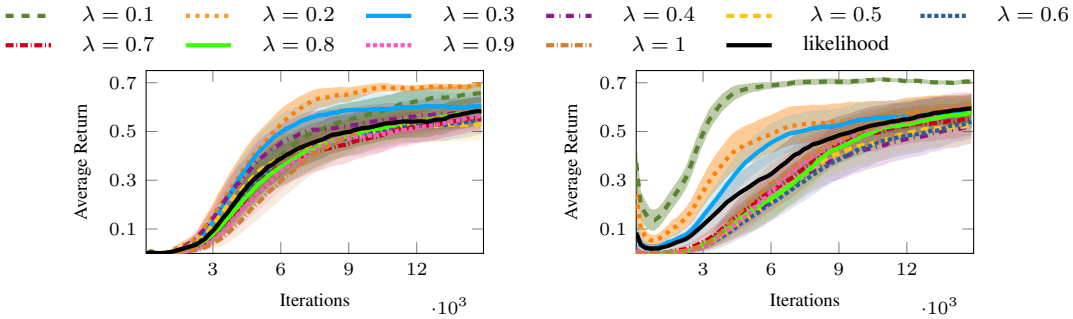
Figure B.2: *Average return achieved by* 1*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*



(a) *2-rooms polynomial dynamic.* **(b)** *2-rooms linear dynamic.*

(c) *2-rooms* sin *dynamic.*

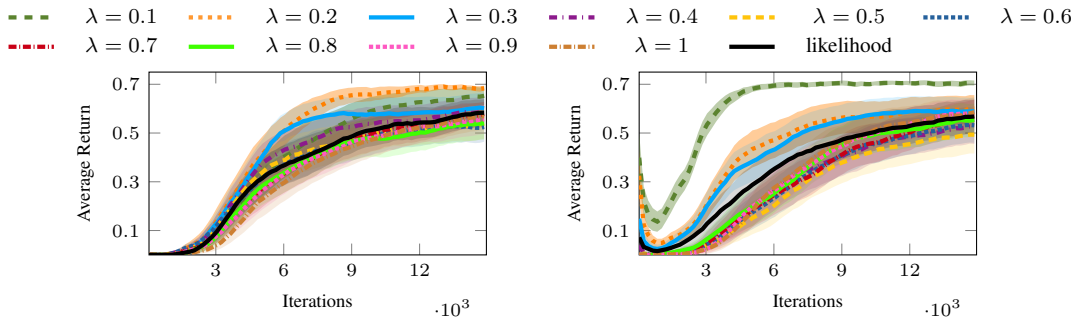
Figure B.3: *Average return achieved by* 3*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*



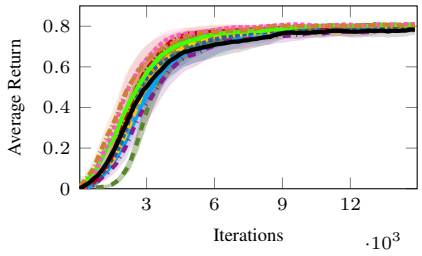
(a) *3-rooms polynomial dynamic.* **(b)** *3-rooms linear dynamic.*

(c) *3-rooms* sin *dynamic.*

Figure B.4: *Average return achieved by* 1*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*

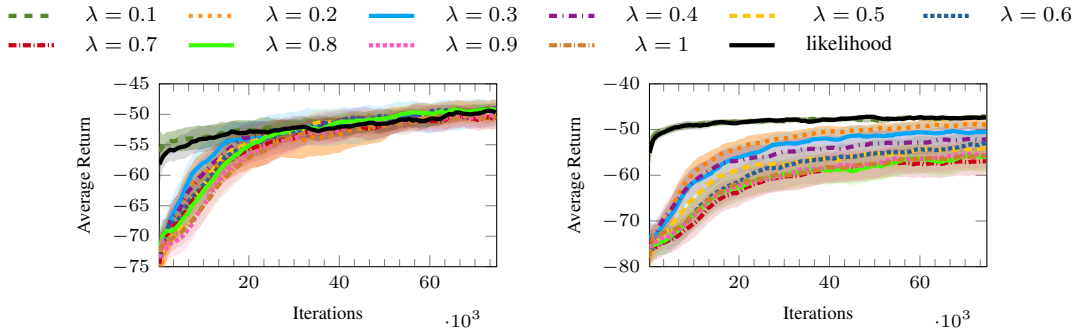


(a) *3-rooms polynomial dynamic.* **(b)** *3-rooms linear dynamic.*

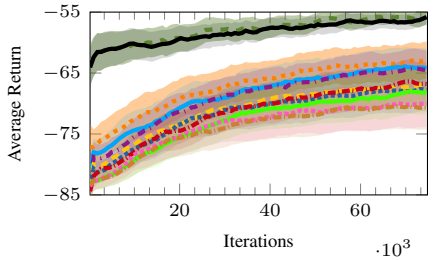


(c) *3-rooms* sin *dynamic.*

Figure B.5: *Average return achieved by* 3*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*

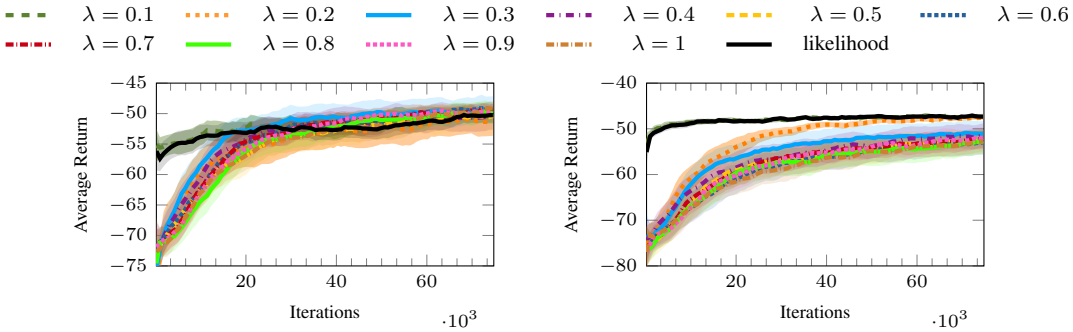


(a) *Mountain Car polynomial dynamic.* **(b)** *Mountain Car linear dynamic.*



(c) *Mountain Car* sin *dynamic.*

Figure B.6: *Average return achieved by* 1*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*



(a) *Mountain Car polynomial dynamic.*

(b) *Mountain Car linear dynamic.*

(c) *Mountain Car* sin *dynamic.*

Figure B.7: *Average return achieved by* 3*-T2VT w.r.t. different choices of* λ *with* 95% *confidence intervals computed using* 50 *independent runs.*

Bibliography

Sherief Abdallah and Michael Kaisers. Addressing environment non-stationarity by repeating q-learning updates. *The Journal of Machine Learning Research*, 17(1):1582–1612, 2016.

T Mitchell Aide, Carlos Corrada-Bravo, Marconi Campos-Cerqueira, Carlos Milan, Giovany Vega, and Rafael Alvarez. Real-time bioacoustics monitoring and automated species identification. *PeerJ*, 1:e103, 2013.

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Lucas N Alegre, Ana LC Bazzan, and Bruno C da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 97–105, 2021.

Cesare Alippi. *Intelligence for embedded systems*. Springer, 2014.

Cesare Alippi, Pietro Braione, Vincenzo Piuri, and Fabio Scotti. A methodological approach to multisensor classification for innovative laser material processing units. In *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188)*, volume 3, pages 1762–1767. IEEE, 2001.

Cesare Alippi, Stavros Ntalampiras, and Manuel Roveri. A cognitive fault diagnosis system for distributed sensor networks. *IEEE transactions on neural networks and learning systems*, 24(8):1213–1226, 2013.

Cesare Alippi, Simone Disabato, and Manuel Roveri. Moving Convolutional Neural Networks to Embedded Systems: The AlexNet and VGG-16 Case. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 212–223, Porto, apr 2018. IEEE. ISBN 978-1-5386-5298-5.

Sven E Anderson, Amish S Dave, and Daniel Margoliash. Template-based automatic recognition of birdsong syllables from continuous recordings. *The Journal of the Acoustical Society of America*, 100(2):1209–1219, 1996.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41–48, 2006.

Bibliography

Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas Navarro, Urmish Thakkar, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul N Whatmough. Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. *arXiv preprint arXiv:2010.11267*, 2020.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

Selin Bastas, Mohammad Wadood Majid, Golrokh Mirzaei, Jeremy Ross, Mohsin M Jamali, Peter V Gorsevski, Joseph Frizado, and Verner P Bingman. A novel feature extraction algorithm for classification of bird flight calls. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1676–1679. IEEE, 2012.

Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Franz Berger, William Freillinger, Paul Primus, and Wolfgang Reisinger. Bird audio detection - dcase 2018. Technical report, DCASE2018 Challenge, September 2018.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Forrest Briggs, Raviv Raich, and Xiaoli Z Fern. Audio classification of bird species: A statistical manifold approach. In *2009 Ninth IEEE International Conference on Data Mining*, pages 51–60. IEEE, 2009.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Stuart A Brooker, Philip A Stephens, Mark J Whittingham, and Stephen G Willis. Automated detection and classification of birdsong: An ensemble approach. *Ecological Indicators*, 117:106609, 2020.

Jacob Burbea. The convexity with respect to gaussian distributions of divergences of order α. *Utilitas Mathematica*, 26: 171–192, 1984.

Giuseppe Canonaco, Marcello Restelli, and Manuel Roveri. Model-free non-stationarity detection and adaptation in reinforcement learning. In *European Conference on Artificial Intelligence*, pages 1047–1054. IOS Press, 2020a.

Giuseppe Canonaco, Manuel Roveri, Cesare Alippi, Fabrizio Podenzani, Antonio Bennardo, Marco Conti, and Nicola Mancini. Corrosion prediction in oil and gas pipelines: a machine learning approach. In *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE, 2020b.

Giuseppe Canonaco, Alex Bergamasco, Alessio Mongelluzzo, and Manuel Roveri. Adaptive federated learning in presence of concept drift. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2021a.

Giuseppe Canonaco, Manuel Roveri, Cesare Alippi, Fabrizio Podenzani, Antonio Bennardo, Marco Conti, and Nicola Mancini. A machine-learning approach for the prediction of internal corrosion in pipeline infrastructures. In *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE, 2021b.

Giuseppe Canonaco, Manuel Roveri, Cesare Alippi, Fabrizio Podenzani, Antonio Bennardo, Marco Conti, and Nicola Mancini. A transfer-learning approach for corrosion prediction in pipeline infrastructures. *Applied Intelligence*, pages 1–16, 2021c.

Giuseppe Canonaco, Andrea Soprani, Matteo Giuliani, Andrea Castelletti, Manuel Roveri, and Marcello Restelli. Time-variant variational transfer for value functions. In *Uncertainty in Artificial Intelligence*, pages 876–886. PMLR, 2021d.

A Castelletti, Stefano Galelli, Marcello Restelli, and Rodolfo Soncini-Sessa. Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(9), 2010.

Olivier Catoni. Pac-bayesian supervised classification: the thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*, 2007.

Yash Chandak, Scott Jordan, Georgios Theocharous, Martha White, and Philip S Thomas. Towards safe policy improvement for non-stationary mdps. *Advances in Neural Information Processing Systems*, 33, 2020a.

Yash Chandak, Georgios Theocharous, Shiv Shankar, Sridhar Mahadevan, Martha White, and Philip S Thomas. Optimizing for the future in non-stationary mdps. *arXiv preprint arXiv:2005.08158*, 2020b.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *arXiv preprint arXiv:1901.08755*, 2019.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism. In *International Conference on Machine Learning*, pages 1843–1854. PMLR, 2020.

Samuel PM Choi, Dit-Yan Yeung, and Nevin Lianwen Zhang. An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems*, pages 987–993, 2000.

Lior Cohen, Gil Avrahami, Mark Last, and Abraham Kandel. Info-fuzzy algorithms for mining dynamic data streams. *Applied Soft Computing*, 8(4):1283–1294, 2008.

Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pages 217–224. ACM, 2006.

Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200, 2007.

G De Masi, M Gentile, R Vichi, R Bruschi, and G Gabetta. Multiscale processing of loss of metal: a machine learning approach. In *Journal of Physics: Conference Series*, volume 869, page 012023. IOP Publishing, 2017.

Giulia De Masi, Roberta Vichi, Manuela Gentile, Roberto Bruschi, and Giovanna Gabetta. A neural network predictive model of pipeline internal corrosion profile. In *Proceeding of First International Conference on Systems Informatics, Modeling and Simulation. IEEE Computer Society Washington. DC, USA*, volume 29, pages 01–05, 2014.

C De Waard and U_ Lotz. Prediction of co˜ 2 corrosion of carbon steel. In *CORROSION-NATIONAL ASSOCIATION OF CORROSION ENGINEERS ANNUAL CONFERENCE-*. NACE, 1993.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Simone Disabato and Manuel Roveri. Incremental on-device tiny machine learning. In *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, pages 7–13, 2020.

Simone Disabato, Giuseppe Canonaco, Paul G Flikkema, Manuel Roveri, and Cesare Alippi. Birdsong detection at the edge with deep learning. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 9–16. IEEE, 2021.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. A kernel-based approach to non-stationary reinforcement learning in metric spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 3538–3546. PMLR, 2021.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access, 2016.

Honghui Du, Leandro L Minku, and Huiyu Zhou. Multi-source transfer learning for non-stationary environments. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

Yilun Du and Karthic Narasimhan. Task-agnostic dynamics priors for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1696–1705, 2019.

Lixin Duan, Dong Xu, and Shih-Fu Chang. Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1338–1345. IEEE, 2012.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Olivier Dufour, Thierry Artieres, Hervé Glotin, and Pascale Giraudet. Clusterized mel filter cepstral coefficients and support vector machines for bird song identification. *Proceedings of the 1st workshop on Machine Learning for Bioacoustics joint to the 30th ICML*, pages 89–93, 2013.

Bradley Efron and Robert J Tibshirani. An introduction to the bootstrap, volume 57 of. *Monographs on Statistics and applied probability*, page 17, 1993.

Vassiliy A Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

André C Ferreira, Liliana R Silva, Francesco Renna, Hanja B Brandl, Julien P Renoult, Damien R Farine, Rita Covas, and Claire Doutrelant. Deep learning-based methods for individual recognition in small birds. *Methods in Ecology and Evolution*, 11(9):1072–1085, 2020.

George Fishman. *Monte Carlo: concepts, algorithms, and applications*. Springer Science & Business Media, 2013.

Elizabeth JS Fox, J Dale Roberts, and Mohammed Bennamoun. Text-independent speaker identification in birds. In *Ninth International Conference on Spoken Language Processing*, 2006.

Karl-Heinz Frommolt and Klaus-Henry Tauchert. Applying bioacoustic methods for long-term monitoring of a nocturnal wetland bird. *Ecological Informatics*, 21:4–12, 2014.

Haotian Fu, Hongyao Tang, Jianye Hao, Chen Chen, Xidong Feng, Dong Li, and Wulong Liu. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. *arXiv preprint arXiv:2009.13891*, 2020.

Pratik Gajane, Ronald Ortner, and Peter Auer. A sliding-window algorithm for markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*, 2018.

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

Matteo Giuliani, Yu Li, Andrea Castelletti, and C Gandolfi. A coupled human-natural systems analysis of irrigated agriculture under changing climate. *Water Resources Research*, 52(9):6928–6947, 2016.

Martin Graciarena, Michelle Delplanche, Elizabeth Shriberg, Andreas Stolcke, and Luciana Ferrer. Acoustic front-end optimization for bird species recognition. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 293–296. IEEE, 2010.

Thomas Grill and Jan Schlüter. Two convolutional neural networks for bird detection in audio signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1764–1768. IEEE, 2017.

Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1): 17–61, 1988.

Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential decision-making under non-stationary environments via sequential change-point detection. In *Learning over Multiple Contexts (LMCE)*, 2014.

Eric C Hall and Rebecca M Willett. Online convex optimization in dynamic environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):647–662, 2015.

Peter Hall, Hans-Georg Müller, and Ping-Shi Wu. Real-time density and mode estimation with application to time-dynamic mode tracking. *Journal of Computational and Graphical Statistics*, 15(1):82–100, 2006.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

Patrick J Hart, Esther Sebastián-González, Ann Tanimoto, Alia Thompson, Tawn Speetjens, Madolyn Hopkins, and Michael Atencio-Picado. Birdsong characteristics are related to fragment size in a neotropical forest. *Animal Behaviour*, 137:45–52, 2018.

Douglas M Hawkins, Peihua Qiu, and Chang Wook Kang. The changepoint model for statistical process control. *Journal of quality technology*, 35(4):355–366, 2003.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE, 2007.

Timothy Classen Hesterberg. *Advances in importance sampling*. PhD thesis, Stanford University, 1988.

Ivan Himawan, Michael Towsey, and Paul Roe. 3d convolution recurrent neural networks for bird sound detection. In *Proceedings of the 3rd Workshop on Detection and Classification of Acoustic Scenes and Events*, pages 1–4, 2018.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

Allison S Injaian, Lauren Y Poon, and Gail L Patricelli. Effects of experimental anthropogenic noise on avian settlement patterns and reproductive success. *Behavioral Ecology*, 29(5):1181–1189, 2018.

M Chris Jones. Simple boundary correction for kernel density estimation. *Statistics and computing*, 3(3):135–146, 1993.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5915–5926, 2019.

Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in neural information processing systems*, pages 6250–6261, 2017.

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

Joseph A Kogan and Daniel Margoliash. Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden markov models: A comparative study. *The Journal of the Acoustical Society of America*, 103(4):2185–2196, 1998.

Chih-Yuan Koh, Jaw-Yuan Chang, Chiang-Lin Tai, Da-Yo Huang, Han-Hsing Hsieh, and Yi-Wen Liu. Bird sound classification using convolutional neural networks. In *CLEF (Working Notes)*, 2019.

Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86, 1951.

Mario Lasseck. Bird song classification in field recordings: winning solution for nips4b 2013 competition. In *Proc. of int. symp. Neural Information Scaled for Bioacoustics, sabiod. org/nips4b, joint to NIPS, Nevada*, pages 176–181, 2013.

Mario Lasseck. Acoustic bird detection with deep convolutional neural networks. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pages 143–147, 2018.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 599–606, 2010.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551, 2008.

Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 32:7216–7225, 2019.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Chang-Hsing Lee, Sheng-Bin Hsu, Jau-Ling Shih, and Chih-Hsun Chou. Continuous birdsong recognition using gaussian mixture modeling of image shape features. *IEEE Transactions on Multimedia*, 15(2):454–464, 2012.

Lucas Lehnert and Michael L Littman. Transfer with model features in reinforcement learning. *arXiv preprint arXiv:1807.01736*, 2018.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 410–419, 2012.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Xiaogang Li, Dawei Zhang, Zhiyong Liu, Zhong Li, Cuiwei Du, and Chaofang Dong. Materials science: Share corrosion data. *Nature News*, 527(7579):441, 2015.

Kexi Liao, Quanke Yao, Xia Wu, and Wenlong Jia. A numerical corrosion rate prediction method for direct assessment of wet gas gathering pipelines internal corrosion. *Energies*, 5(10):3892–3907, 2012.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Sinno Jialin Pan, and S Yu Philip. Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1076–1089, 2013.

Odalric-Ambrym Maillard, Rémi Munos, Alessandro Lazaric, and Mohammad Ghavamzadeh. Finite-sample analysis of bellman residual minimization. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 299–314, 2010.

Peter Marler and Donald Isaac. Physical analysis of a simple bird song as exemplified by the chipping sparrow. *The Condor*, 62(2):124–135, 1960.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Russell Mendonca, Xinyang Geng, Chelsea Finn, and Sergey Levine. Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling. *arXiv preprint arXiv:2006.07178*, 2020.

Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, pages 5442–5454, 2018.

Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising markov logic networks for transfer learning. In *Aaai*, volume 7, pages 608–614, 2007.

Leandro L Minku. Transfer learning in non-stationary environments. In *Learning from Data Streams in Evolving Environments*, pages 13–37. Springer, 2019.

Leandro L Minku and Xin Yao. How to make best use of cross-company data in software effort estimation? In *Proceedings of the 36th International Conference on Software Engineering*, pages 446–456, 2014.

Rajdeep Mukherjee, Dipyaman Banerjee, Kuntal Dey, and Niloy Ganguly. Convolutional recurrent neural network based bird audio detection. Technical report, DCASE2018 Challenge, September 2018.

Rafael Hernández Murcia and Vıctor Suárez Paniagua. Bird identification from continuous audio recordings. *Proceedings of the 1st workshop on Machine Learning for Bioacoustics joint to the 30th ICML*, pages 96–97, 2013.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Lawrence Neal, Forrest Briggs, Raviv Raich, and Xiaoli Z Fern. Time-frequency segmentation of bird song in noisy acoustic environments. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2012–2015. IEEE, 2011.

Srdjan Nešić. Key issues related to modelling of internal corrosion of oil and gas pipelines–a review. *Corrosion science*, 49(12):4308–4338, 2007.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019. URL `https://arxiv.org/abs/1912.06680`.

Ronald Ortner, Pratik Gajane, and Peter Auer. Variational regret bounds for reinforcement learning. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*, 2019.

Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

Sindhu Padakandla, KJ Prabuchandran, and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11):3590–3606, 2020.

Kuldip K Paliwal, James G Lyons, and Kamil K Wójcicki. Preference for 20-40 ms window duration in speech analysis. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–4. IEEE, 2010.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Sinno Jialin Pan, James T Kwok, Qiang Yang, et al. Transfer learning via dimensionality reduction. In *AAAI*, volume 8, pages 677–682, 2008.

Jan Peter Patist. Optimal window change detection. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 557–562. IEEE, 2007.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

Christian F Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps transferable model-based rl in a handful of trials. *arXiv preprint arXiv:2002.03072*, 2020.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4): 682–697, 2008.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Ilyas Potamitis. Automatic classification of a taxon-rich community recorded in the wild. *PloS one*, 9(5), 2014.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.

Nirosha Priyadarshani, Stephen Marsland, Isabel Castro, and Amal Punchihewa. Birdsong denoising using wavelets. *PloS one*, 11(1), 2016.

Nirosha Priyadarshani, Stephen Marsland, and Isabel Castro. Automated birdsong recognition in complex acoustic environments: a review. *Journal of Avian Biology*, 49(5):jav–01447, 2018.

Nirosha Priyadarshani, Stephen Marsland, Julius Juodakis, Isabel Castro, and Virginia Listanti. Wavelet filters for automated recognition of birdsong in long-time field recordings. *Methods in Ecology and Evolution*, 11(3):403–417, 2020.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007.

Alfréd Rényi. On measures of information and entropy. In *Proceedings of the 4th Berkeley symposium on mathematics, statistics and probability*, volume 1, 1961.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Kenneth V Rosenberg, Adriaan M Dokter, Peter J Blancher, John R Sauer, Adam C Smith, Paul A Smith, Jessica C Stanton, Arvind Panjabi, Laura Helft, Michael Parr, et al. Decline of the north american avifauna. *Science*, 366 (6461):120–124, 2019.

Manuel Roveri. Learning discrete-time markov chains under concept drift. *IEEE transactions on neural networks and learning systems*, 2019.

Zachary J Ruff, Damon B Lesmeister, Leila S Duchac, Bharath K Padmaraju, and Christopher M Sullivan. Automated identification of avian vocalizations with deep convolutional neural networks. *Remote Sensing in Ecology and Conservation*, 2019.

Schlumberger. Olga dynamic multiphase flow simulator. URL `https://www.software.slb.com/sitecore/content/software/home/software/products/olga`.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series, 2018.

Masayuki Senzaki, Jesse R Barber, Jennifer N Phillips, Neil H Carter, Caren B Cooper, Mark A Ditmer, Kurt M Fristrup, Christopher JW McClure, Daniel J Mennitt, Luke P Tyrrell, et al. Sensory pollutants alter bird phenology and fitness across a continent. *Nature*, 587(7835):605–609, 2020.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

R. Soncini-Sessa, A. Castelletti, and E. Weber. *Integrated and participatory water resources management: Theory*. Elsevier, Amsterdam, NL, 2007.

Norsork Standard. Co2 corrosion rate calculation model. *Majorstural, Norway: Norwegian Technological Standards Institute Oscarsgt*, 20, 2005.

Dan Stowell, Michael D Wood, Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *Methods in Ecology and Evolution*, 10(3): 368–380, 2019.

W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382, 2001.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Richard S Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878. ACM, 2007.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009. URL `http://jmlr.org/papers/v10/taylor09a.html`.

Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep):2125–2167, 2007.

Matthew E Taylor, Nicholas K Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 488–505. Springer, 2008.

Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148, 2016.

Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

Andrea Tirinzoni, Rafael Rodriguez Sanchez, and Marcello Restelli. Transfer of value functions via variational methods. *Advances in Neural Information Processing Systems*, 31:6179–6189, 2018a.

Andrea Tirinzoni, Andrea Sessa, Matteo Pirotta, and Marcello Restelli. Importance weighted transfer of samples in reinforcement learning. In *International Conference on Machine Learning*, pages 4936–4945. PMLR, 2018b.

Andrea Tirinzoni, Mattia Salvini, and Marcello Restelli. Transfer of samples in policy search via multiple importance sampling. In *International Conference on Machine Learning*, pages 6264–6274, 2019.

Michael Towsey, Birgit Planitz, Alfredo Nantes, Jason Wimmer, and Paul Roe. A toolbox for animal call recognition. *Bioacoustics*, 21(2):107–125, 2012.

Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *ACM Computing Surveys (CSUR)*, 53(2):1–33, 2020.

Fabio Vesperini, Leonardo Gabrielli, Emanuele Principi, and Stefano Squartini. A capsule neural networks based approach for bird audio detection. Technical report, DCASE2018 Challenge, September 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 16(2):117–186, 1945.

Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

Pengcheng Wu and Thomas G Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the twenty-first international conference on Machine learning*, page 110, 2004.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.

Jiachen Yang, Brenden Petersen, Hongyuan Zha, and Daniel Faissol. Single episode policy transfer in reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJeQoCNYDS`.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.