Titre: Title:	Contributions to the Trustworthy Machine Learning Pipeline: Data Selection, Training, and Post-training Verification through Convexity and the Wasserstein Distance
Auteur: Author:	Julien Pallage
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pallage, J. (2025). Contributions to the Trustworthy Machine Learning Pipeline: Data Selection, Training, and Post-training Verification through Convexity and the Wasserstein Distance [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/64736/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/64736/
Directeurs de recherche: Advisors:	Antoine Lesage-Landry
Programme: Program:	Génie électrique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Contributions to the Trustworthy Machine Learning Pipeline: Data Selection, Training, and Post-training Verification through Convexity and the Wasserstein Distance

JULIEN PALLAGE
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Avril 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Contributions to the Trustworthy Machine Learning Pipeline: Data Selection, Training, and Post-training Verification through Convexity and the Wasserstein Distance

présenté par **Julien PALLAGE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Shuang GAO, président
Antoine LESAGE-LANDRY, membre et directeur de recherche
Bowen YI, membre

DEDICATION

À Georgette qui a su me transmettre sa curiosité.

ACKNOWLEDGEMENTS

This work would not have been possible without the support of so many! I want to extend my gratitude for the thoughtful and sincere guidance of Prof. Antoine Lesage-Landry; the day-to-day help in the realm of data science from Dr. Salma Naccache and Dr. Bertrand Scherrer; the endorsement of Odile Noël, Ahmed Abdellatif, and Steve Boursiquot from Hilo; the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de Recherche du Québec (FRQ), MITACS, and Hilo; as well as the fun discussions and good time spent with friends and colleagues, Olivier, Jean-Luc, Matthias, Abraham, Loreley, William, Samuel, Xavier, Étienne, Christina, Julien, and Tudor. Finally, I want to thank my partner, Nohaila, for always being there for me, it would have been a lot harder without you.

On another note, the works of Prof. Spyros Chatzivasileiadis and Prof. Peyman Mohajerin Esfahani were instrumental in motivating this thesis. I heartfully invite interested readers to consult their respective literature.

RÉSUMÉ

Dans ce travail, nous étudions les défis et risques liés au déploiement de modèles complexes d'apprentissage machine (*machine learning*, ML) dans des applications critiques. Nous débutons par un survol de différentes méthodes de la litérature utilisées pour renforcer la fiabilité de ces modèles afin d'introduire le concept de pipeline d'apprentissage machine fiable (*trustworhty*). Nous étudions spécifiquement les besoins et les contraintes associées aux centrales virtuelles étant donné leur besoin de meilleurs algorithmes de prédiction et la sévérité potentielle d'erreurs opérationnelles. Nous proposons plusieurs contributions visant à améliorer la sûreté des modèles d'apprentissage, et ce tout au long de leur cycle de vie, sans compromettre leur performance.

Comme première contribution, nous présentons une nouvelle méthode d'entraînement robuste en distribution, sous la distance de Wasserstein, pour les réseaux de neurones convexes peu profond (*shallow convex neural networks*, SCNNs) soumis à des jeux de données corrompues. Notre approche repose sur un nouveau problème d'optimisation d'entraînement convexe permettant de faire le pont entre les réseaux de neurones ReLU convexes optimaux et les réseaux de neurones ReLU non-convexes. Nous adaptons ce programme d'entraînement sous sa formulation robuste en distribution avec la distance de Wasserstein de premier ordre. Cette méthode est conservatrice, a une basse stochasticité attribuable à la convexité, est résoluble avec des solveurs libres accès, et peut être facilement déployées à grande échelle. Nous obtenons des garanties de performance théoriques hors échantillon, nous démontrons comment adapter l'entraînement pour inclure des contraintes physiques convexes, et nous proposons un problème de vérification post-entraînement pour évaluer la stabilité des réseaux de neurones convexes peu profond. Finalement, nous évaluons numériquement notre méthode sur des jeux de données synthétiques; une application réelle de la centrale virtuelle québécoise, soit la prédiction horaire de consommation d'énergie d'immeubles non-résidentiels; et testons la stabilité sur des jeux de données de références en apprentissage machine.

Dans un deuxième temps, nous proposons une nouvelle méthode non-supervisée de détection d'anomalies avec la distance de Wasserstein tronquée (*sliced-Wasserstein*). Cette méthode de filtrage de données est conceptuellement intéresseantes pour pré-traiter les données d'entraînement de modèles de ML étant donné son conservatisme et son interprétation basée sur le transport optimal. Pour limiter le temps de calcul, nous proposons également deux approximations. La première approximation consiste à traiter en parallèle des représentations de cardinalité plus petite du jeu de données initial, tandis que la deuxième réduit le problème à la distance euclidienne. Nous proposons également un premier jeu de données ouvertes qui permet d'observer des mécanismes de gestion

de demande locale par rémunération de pointe (*localized critical peak rebates*). Nous présentons une analyse qualitative et numérique de notre méthode de filtrage pour la détection d'anomalies et la sélection de données d'entraînement et l'utilisons pour obtenir un premier *benchmark* sur notre jeu de données ouvertes.

Enfin, nous explorons les implications de nos propositions pour la commande de systèmes dynamiques. Nous proposons une formulation de commande prédictive non linéaire exploitant notre représentation convexe en nombres entiers des SCNNs. Nous montrons également un exemple de comment la vérification post-entraînement peut améliorer la satisfaction des contraintes critiques en matière de sécurité lors de la commande.

ABSTRACT

In this work, we study some of the issues and risks of deploying complex machine learning models in safety-critical applications. By looking into the literature, we first present a global portrait of the different initiatives to make these models more reliable. We then introduce the general concept of *trustworthy machine learning pipelines* from pre-deployment to model exploitation. We specifically study the needs and constraints associated with virtual power plants as they are perfect examples of large-scale critical applications benefiting from the integration of complex machine learning. We then propose different contributions aiming at enhancing model safety with minimal compromise.

We first propose Wasserstein distributionally robust shallow convex neural networks (SCNNs) to provide reliable nonlinear predictions when subject to adverse and corrupted datasets. Our approach is based on a new convex training program for ReLU-based shallow neural networks which allows us to cast the problem as an exact, tractable reformulation of its order-1 Wasserstein distributionally robust counterpart. Our training procedure is conservative, has low stochasticity, is solvable with open-source solvers, and is scalable to large industrial deployments. We provide out-of-sample performance guarantees, show that hard convex physical constraints can be enforced in the training program, and propose a mixed-integer convex post-training verification program to evaluate model stability. Finally, we numerically demonstrate the performance of our model on a synthetic experiment, a real-world power system application, viz., the prediction of nonresidential buildings' hourly energy consumption in the context of virtual power plants, and on benchmark datasets.

We then propose a new unsupervised anomaly detection method using the sliced-Wasserstein distance. This filtering technique is conceptually interesting for ML pipelines deploying machine learning models in critical sectors as it offers a conservative data selection and an optimal transport interpretation. To ensure the scalability of the method, we provide two approximations. The first approximation relies on filtering reduced-cardinality representations of the datasets in parallel, it is viable when multiple computational threads are available. The second approximation makes use of a fast-to-compute Euclidian distance approximation. Additionally, we open the first dataset showcasing localized critical peak rebate demand response in a northern climate. We present the filtering patterns of our method on synthetic datasets and numerically benchmark our method for anomaly detection and for training data selection. We then use our method as part of a first benchmark model for our open-source dataset.

Finally, we explore some of the implications of our previous propositions in control applications. We propose a nonlinear data-driven model predictive control program by reutilizing our mixed-integer convex representation of SCNNs. We then hint at how post-training verification can be leveraged to enhance safety-critical constraint satisfaction during control.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AD	Anomaly Detection
ARIMA	Autoregressive Integrated Moving Average
BBO	Blackbox Optimization
CNN	Convolutional Neural Network
CPR	Critical Peak Rebates
DER	Distributed Energy Resource
DR	Demand Response
DRO	Distributionally Robust Optimization
FNN	Feedforward Neural Network
GP	Gaussian Process
GPU	Graphics Processing Unit
IoT	Internet of Things
KNN	k-Nearest Neighbours
LCPR	Localized Critical Peak Rebates
LOF	Local Outlier Factor
MAE	Mean Absolute Error
MICP	Mixed-Integer Convex Programming
MIP	Mixed-Integer Programming
ML	Machine Learning
MPC	Model Predictive Control
NN	Neural Network
OT	Optimal Transport
PCSCNN	Physics-constrained Shallow Convex Neural Network
PDE	Partial Differential Equation
PTV	Post-training Verification
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
SGD	Stochastic Gradient Descent
SNN	Shallow Neural Network
SCNN	Shallow Convex Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
SW	Sliced-Wasserstein

SWD	Sliced-Wasserstein Distance
TPE	Tree-structured Parzen Estimator
UCI	University of California Irvine
VPP	Virtual Power Plant
WaDiRo	Wasserstein Distributionally Robust
WD	Wasserstein Distance

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

What makes machine learning models trustworthy? I want to reassure the reader that we will not dive into the philosophical implications of this question, as it is purely rhetorical. The short and blunt answer is: most of the time, they are not. Using machine learning models should always be approached with a healthy dose of skepticism. However, this does not mean they should be disregarded. Machine learning is a powerful tool with immense potential across a myriad of applications, and skepticism serves as a valuable driver for developing methods that enhance its reliability.

With the rapid advances in computing and the now broader accessibility to try out different machine learning models, enthusiasm for these technologies is at an all-time high. But, as users, it is essential to understand the limitations of the tools we rely on. The key questions we should ask ourselves are:

Question 1. *When can we reasonably trust the machine learning models that we are deploying for a specific application?*

Question 2. *What can we do to maximize the reliability of our machine learning models for a specific application?*

In this work, we are interested in critical applications, i.e., applications in which failure to meet the expected performance can lead to costly consequences. In this line of work, the requirements needed for machine learning models to be considered trustworthy are quite high, especially for models with complex architecture. By complex ML, we refer to any ML process that is of higher complexity than simple classical models, e.g., shallow or deep neural networks, transformers, tree-based models, and Gaussian processes (Sarker 2021). Our primary focus is on virtual power plants.

This chapter begins by introducing the concept of a *trustworthy machine learning pipeline*. We then follow by defining our motivating safety-critical application, virtual power plants, before presenting the general outline of the work, the associated references, and the notation used throughout.

1.1 Trustworthy machine learning

Consider a general feedforward neural network (NN) training problem for a network of $h \in \mathbb{N}$ hidden layers under a loss function $\mathcal{L} : \mathbb{R}^{N\times m_{h+1}} \times \mathbb{R}^{N\times m_{h+1}} \to \mathbb{R}$ which penalizes the difference between the $N \in \mathbb{N}$ training labels and NN predictions of dimension m_{h+1}, the number of hidden neurons in the output layer. Let $\mathbf{x}_j \in \mathbb{R}^{m_0}, \mathbf{y}_j \in \mathbb{R}^{m_{h+1}},$ and $\hat{\mathbf{y}}_j \in \mathbb{R}^{m_{h+1}}$ define training features, training labels, and training predictions, respectively, $\forall j \in [\![N]\!]$. Matrices $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}^{N\times m_{h+1}}$ denote

the collection of all labels and predictions. The problem is as follows:

$$\min_{\mathbf{W}_1,\mathbf{W}_2,\ldots,\mathbf{W}_h,\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_h} \mathcal{L}\left(\hat{\mathbf{Y}},\mathbf{Y}\right) \tag{NNT}$$
$$\text{s.t.} \quad \hat{\mathbf{y}}_j = \phi^h(\mathbf{W}_h^\top(\ldots\phi^2(\mathbf{W}_2^\top\phi^1(\mathbf{W}_1^\top\mathbf{x}_j+\mathbf{b}_1)+\mathbf{b}_2)\ldots)+\mathbf{b}_h) \quad \forall j\in[\![N]\!],$$

where $\phi^i:\mathbb{R}^{m_i}\to\mathbb{R}^{m_{i+1}}$ is the activation function of the hidden layer i, $m_i\in\mathbb{N}^*$ is the number of neurons in layer i, $\mathbf{W}_i\in\mathbb{R}^{m_{i-1}\times m_i}$ are the weights in layer i, and $\mathbf{b}_i\in\mathbb{R}^{m_i}$ are the bias weights of layer i for all $i\in\{1,2,3,\ldots,h\}=:[\![h]\!]$. In our notation, we assume that NN inputs and outputs are vectors. A visual representation of a single-layer (shallow) feedforward NN is presented in Figure 1.1.

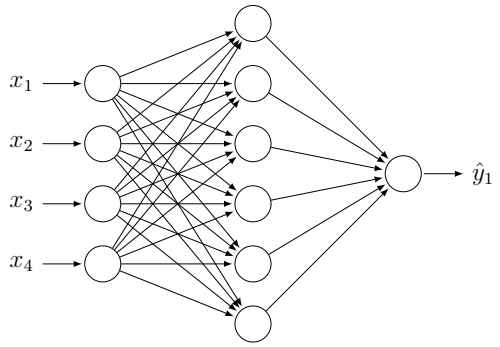


Figure 1.1 Architecture of a feedforward NN with a single hidden layer

A common activation function ϕ^i for regression is the element-wise rectified linear unit (ReLU) operator:

$$\mathrm{ReLU}(\cdot)=\max\{\mathbf{0},\cdot\},$$

as it is accurate when modelling nonlinear patterns and biomimics the required excitation threshold for brain neurons to *activate* (Householder 1941). The acquisition of features is done through the input layer and the output of labels through the output layer such that a neural network of h hidden layers has effectively $h+2$ layers. To solve (NNT), the gradient of the loss is propagated iteratively, one layer at a time, from the output neurons to the input neurons, by computing gradient estimators of the NN architecture at each layer and updating weights with methods derived from non-convex optimization, e.g., stochastic gradient descent (Amari 1993). This procedure is referred to as backpropagation (Bishop and Bishop 2023).

In part because (NNT) is typically non-convex and only solvable through stochastic heuristics with many hyperparameters, NNs have been traditionally disregarded in critical sectors, e.g., energy (Chatzivasileiadis et al. 2022), healthcare (Rasheed et al. 2022), and finance (Giudici and Raffinetti 2023). Even though they tend to be successful nonlinear predictors, their lack of interpretability,

limited out-of-the-box performance guarantees, significant data requirements, as well as high susceptibility to data corruption and other adversarial attacks, have labelled them as unreliable for such applications.

In the past decade, many solutions have been proposed to guide and certify NN training as much as possible to mitigate these inherent limitations before deployment. For example, blackbox optimization (BBO) algorithms with provable convergence properties can add some local convergence guarantees in the hyperparameter tuning phase (Audet et al. 2022, Kawaguchi and Sun 2021) because NNs' loss landscape, with respect to their hyperparameters, can be fractal (Sohl-Dickstein 2024). However, when paired with highly stochastic training programs, their relevance is diminished.

Because their high complexity and opacity make them unpredictable, complete performance assessment is hard to obtain during training. As such, post-training verification frameworks have emerged as a way to certify formal guarantees on trained machine learning (ML) models (Huang et al. 2017). These procedures are conceptually similar to quality control tests in factories. By using a model's trained weights as parameters and representing the model's internal mechanisms in an optimization problem, these frameworks aim to find theoretical worst-case scenarios, e.g., worst-case constraint violation in a specific critical application (Venzke et al. 2020), worst-case instability (Huang et al. 2021), and more (Albarghouthi 2021). A general feedforward neural network post-training verification (PTV) problem takes the form:

$$
\begin{aligned}
&\max_{\mathbf{x},\hat{\mathbf{y}},\boldsymbol{\sigma}} \ \mathcal{V}\left(\hat{\mathbf{y}}, \mathbf{x}, \boldsymbol{\sigma}\right) && && \text{(PTV)}\\
&\text{s.t.} \quad \hat{\mathbf{y}} = \phi^h(\mathbf{W}_h^\top(\ldots \phi^2(\mathbf{W}_2^\top \phi^1(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)\ldots) + \mathbf{b}_h) && &&\\
&\qquad f_i(\hat{\mathbf{y}}, \mathbf{x}, \boldsymbol{\sigma}) \leq 0 && && \forall i \in [\![n]\!]\\
&\qquad g_i(\hat{\mathbf{y}}, \mathbf{x}, \boldsymbol{\sigma}) = 0 && && \forall i \in [\![l]\!]\\
&\qquad \mathbf{x} \in \mathcal{X}, && &&
\end{aligned}
$$

where $\mathbf{x} \in \mathcal{X}$ is any possible input from the feature space $\mathcal{X} \subseteq \mathbb{R}^{m_0}$, $\mathcal{V} : \mathbb{R} \times \mathbb{R}^{m_{h+1}} \times \mathbb{R}^{\text{card}(\boldsymbol{\sigma})} \to \mathbb{R}$ is an objective function which measures a specific guarantee, $\boldsymbol{\sigma}$ represents all of the problem's other variables used in the verification process, and $f_1, f_2, \ldots, f_n, g_1, g_2, \ldots, g_l$ are the family of inequality and equality constraints needed for the certification. Note that a convex representation of the internal mechanisms with respect to the inputs is extremely desirable for such problems because the convergence guarantees to global optimality of convex optimization eliminate any doubt on the verification's validity.

When a neural network is used in a dynamical setting, i.e., a system that evolves with time over a state-space and follows a set of physical rules (Wiggins 2003), the introduction of physical knowl-

edge from the setting into its training procedure can increase its overall reliability and greatly reduce the data requirements to obtain good empirical performances (Karniadakis et al. 2021, Xu et al. 2023). This type of training is often referred to as physics-informed, physics-guided, or physics-constrained depending on how physical knowledge is embedded in the training process. We refer interested readers to Cuomo et al. (2022) for an overview of different techniques. We will cover the most common one, based on weighted multi-objective optimization, in Section 2.1.

Typically, as it is hard to guarantee NN out-of-sample performance over time and their performance may degrade (drift), the aforementioned techniques are combined in pre-deployment and post-deployment loops (Stiasny et al. 2022) with tight MLOps (contraction of machine learning, software development, and operations) procedures for lifecycle management (Kreuzberger et al. 2023). Good MLOps practices help monitor and assert, theoretically and empirically, the quality of ML models from training to deployment but require great computational infrastructure to be well implemented. Such infrastructures are not democratized in every sector, especially energy and healthcare. We present an example of a safe pre-deployment ML pipeline inspired by Stiasny et al. (2022) in Figure 1.2.

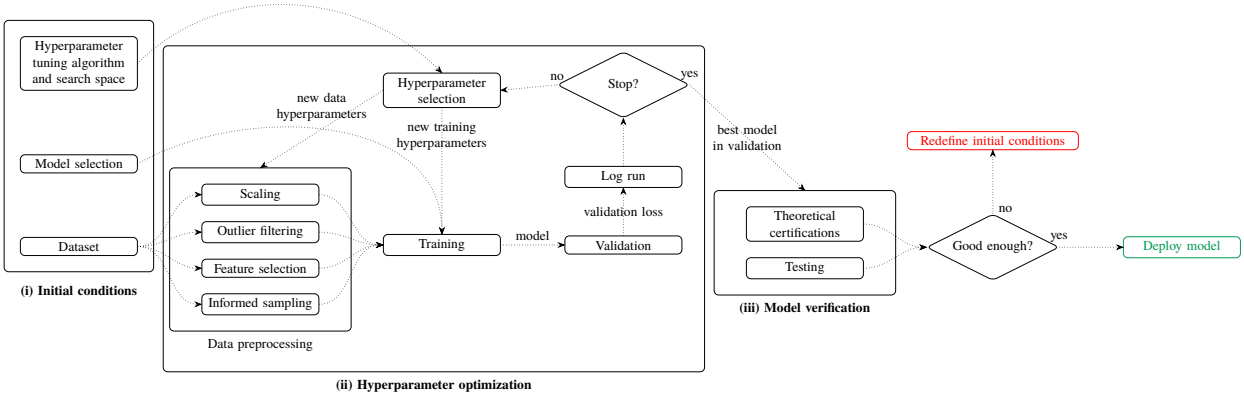


Figure 1.2 Example of safe pre-deployment ML pipeline

A safe pre-deployment pipeline can be divided into three parts: (i) definition of initial conditions, (ii) hyperparameter optimization, and (iii) model verification. Initial conditions englobe the choice of dataset, ML model, hyperparameter tuning algorithm, and the definition of the hyperparameter search space. Hyperparameter optimization covers the typical training and validation phases: hyperparameters are chosen for each run to pre-process the training dataset and train the ML model, the trained model is then evaluated on the validation dataset, and different metrics are logged. This part ends when a stopping criterion is met, e.g., the maximum number of runs is reached. Finally, the best models in the validation phase are verified through post-training certification methods and evaluated on the testing dataset. If the performance expectations are not met by any model, initial

conditions must be redefined. Otherwise, the best ML model can be deployed.

To ensure continued safety during model exploitation, monitoring must continue post-deployment. Figure 1.3 presents an example of a post-deployment pipeline in a dynamical environment subject to safety constraints where the deployed ML models are utilized for forecasting and data-driven control. The loop has two main sections: (i) exploitation of the deployed models for forecasting and data-driven optimal control and (ii) feedback collection and monitoring. In the first section, we use some (or all) models to obtain forecasts with respect to the present environment and the history. These forecasts can serve as parameters in the optimal control problem. Additionally, similarly to post-training verification frameworks, the weights of some (or all) trained models can be utilized in an optimization problem to obtain data-driven optimal controls, e.g., data-driven model predictive control (Garcia et al. 1989) and internal model control (Garcia and Morari 1982). In the second section, offline feedback is collected to monitor constraints satisfaction and prediction errors from all models. If the performance of some models is not good enough, they must be updated following the pre-deployment procedure akin to Figure 1.2.

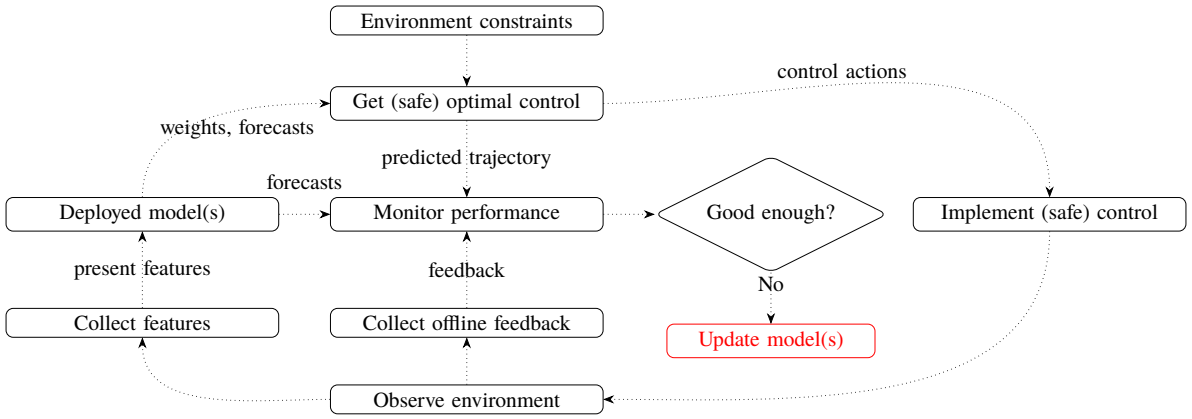


Figure 1.3 Example of safe post-deployment ML pipeline with control

We thus define the trustworthy ML pipeline as follows.

Definition 1 (Trustworthy ML pipeline)**.** *A trustworthy machine learning pipeline is an automated procedure that aims to maximize the reliability of ML models throughout their whole lifecycle, from training to deployment. By combining safety mechanisms with constant monitoring, it deploys ML models that can be consistently trusted with respect to the stakes of the application of deployment.*

The trustworthy ML pipeline promotes ML models that are precise, predictable, explainable, robust, and respectful of application constraints while also having good performance guarantees, theoretical certifications, and limited bias. We refer interested readers to Varshney (2022) for an overview of trustworthy ML.

1.2 Virtual power plants

In this work, our general goal is to make improvements to the safe ML pipeline for critical applications, but, as stated earlier, we find motivation specifically in virtual power plant (VPP) applications.

VPPs, first defined by Pudjianto et al. (2007), are decentralized networks of distributed energy resources (DERs), e.g., photovoltaic panels, wind farms, electric vehicle batteries, thermostats, and other controllable grid-edge energy resources, that are collectively managed and operated as a unified entity through a central control system. Instead of generating electricity from a single location like typical power plants, VPPs mobilize and incentivize geographically dispersed generation, storage, and consumption to offer a vast range of grid services. For example, a VPP can be used for peak shaving, frequency regulation, load tracking, and phase balancing by coordinating its fleet accordingly. It relies on the forecast, the optimization, and the control of these distributed assets to exploit flexibility and meet grid needs with limited infrastructure change as well as minimal investment (Zhang et al. 2019). The concept of VPP stands at the centerpiece of the grid decarbonization paradigm as it allows the integration of more intermittent renewable energy sources into the grid by addressing their inherent instability with added flexibility.

Demand response (DR) is one of the many tools utilized by VPP operators to gain flexibility. DR encourages customer power consumption pattern shifts by leveraging financial incentives (Albadi and El-Saadany 2007). It is either enforced directly through load control or indirectly by promoting specific human behaviours. Some notable DR programs include real-time pricing (Lijesen 2007), time-of-use pricing (Yang et al. 2013), critical peak pricing (Herter 2007), and critical peak rebates (Zhang and Li 2012). To be effective, DR programs can leverage historical data and machine learning (ML) to fine-tune and optimize their services for current and future needs. For example, in real-time pricing programs, electricity rates are modified on a small timescale to incentivize immediate or near-immediate consumption changes. Because there is no trivial mapping between aggregated power consumption at a specific time and electricity rates, historical data and trained ML models can be used to find a price that answers present grid needs (Ruan et al. 2024). A sudden increase in solar radiance and wind speed could lead to lower electricity rates as a way to encourage users to meet the current renewable production.

VPPs are essential for a sustainable, reliable, and resilient grid and algorithms are fundamental in its making. Unfortunately, there is a general trade-off. More precise ML models in VPP applications should lead to better decision-making, facilitate renewable integration, and thus help towards grid decarbonization. Yet more precise ML models often come with an added complexity, which makes them riskier for deployment. Finally, implementing a complete safe ML pipeline can sometimes be impossible for some large-scale VPP applications where computational resources is limited.

1.3 Research questions

Having introduced the main ideas behind trustworthy ML pipelines and virtual power plants, we can now synthesize our main research questions.

Question 3. *How can we increase the trustworthiness of neural networks in large-scale critical applications?*

Question 4. *How can we keep the safe ML pipeline lightweight and user-friendly for real-world applications and constraints?*

Question 5. *What are the main challenges of deploying complex ML models in virtual power plant applications, and how can we solve them?*

Most of the computational burden from the pre-deployment pipeline stems from hyperparameter optimization. As such, there is a growing interest in designing complex predictive models with low-stochasticity training procedures, e.g., few hyperparameters and convex with respect to the weights, as well as lower data pre-processing requirements, e.g., by being distributionally robust. These models simplify the hyperparameter optimization phase and have more chance than standard models to meet the requirements of the verification phase.

In the first part of our work, our motivation is to design a nonlinear predictive ML model with a simple low-stochasticity training program that can be guided by physical constraints and that has inherent performance guarantees. We also aim to design this model with an architecture that can be easily included in post-training verification frameworks or data-driven control applications, if needs be. Shallow convex neural networks, as we will show, are perfect candidates for the task. In doing so, we hope to lower computational infrastructure needs for safe deployment in large-scale critical sectors where a lack of expertise and budget constraints do not permit state-of-the-art MLOps pipelines at all stages, and yet accurate nonlinear predictions are required. We specifically target VPP applications where reliable forecasts of distributed assets are crucial for the safe operations of the electrical power grid. As an example of a standard VPP application, we utilize our model to predict the hourly energy consumption of individual non-residential buildings enrolled in demand response programs (Siano 2014). Because of their immense impact on distribution grids and their diverse activities, non-residential buildings have complex nonlinear consumption patterns that must be predicted accurately to enhance grid operation (Massana et al. 2015). As each building needs its model and thus possibly thousands of models must be deployed, scalability issues arise when utilizing complex stochastic ML training procedures in complete, thorough, safe MLOps pipelines. Finally, the sensitivity of the application, and the risk associated with poor predictions, promote models with theoretical guarantees, mechanisms to respect physical constraints, and robustness to

possible anomalies in the training data (Zhang et al. 2021). We refer readers to Chapter 3 for the complete treatment of our proposed model.

Moreover, dataset quality can highly influence the performance of ML models. Similarly to other internet of things (IoT) applications, virtual power plants may suffer from heavy data corruption that arises from their reliance on sensors and telecommunication systems. The adversarial properties stemming from the amalgam of numerical errors, noise in sensor readings, telemetry issues, meter outages, and unusual extreme events can disrupt the prediction quality of ML models. Outlier filtering is thus primordial in a reliable ML pipeline, especially when limited regularization mechanisms are implemented during training.

In the second part of our work, we propose a new outlier filtering method for training data selection based on optimal transport. On the side, we also open-source a new dataset that showcases a unique DR mechanism. We refer readers to Chapter 4 for the complete proposition.

In the conclusion, we discuss some open perspectives in control stemming from our previous work. These perspectives are treated in Section 5.2.

1.4 Papers and talks derived from this Master's thesis

We compile a list of references associated with this thesis.

Pre-prints and workshops.

- Pallage J, Lesage-Landry A (2025) Sliced-wasserstein distance-based data selection. *arXiv preprint arXiv:2504.12918*
- Pallage J, Scherrer B, Naccache S, Bélanger C, Lesage-Landry A (2024) Sliced-Wasserstein-based Anomaly Detection and Open Dataset for Localized Critical Peak Rebates. *NeurIPS 2024 Workshop on Tackling Climate Change with Machine Learning*
- Pallage J, Lesage-Landry A (2024b) Wasserstein Distributionally Robust Shallow Convex Neural Networks. *arXiv preprint arXiv:2407.16800*

Talks, posters, and seminars.

- Pallage J (2025) The Trustworthy Machine Learning Pipeline for Virtual Power Plants, 5th GERAD's Student Day, Montréal, Québec, Canada
- Pallage J (2024) Advances in the Trustworthy Machine Learning Pipeline, Seminars at the Automatic Control Lab, École Polytechnique Fédérale de Lausanne, Online.

- Pallage J, Lesage-Landry A (2024c) Wasserstein Distributionally Robust Shallow Convex Neural Networks, 25th International Symposium on Mathematical Programming, Montréal, Québec, Canada.
- Pallage J, Lesage-Landry A (2024d) Wasserstein Distributionally Robust Shallow Convex Neural Networks, Optimization Days, Montréal, Québec, Canada.
- Pallage J, Lesage-Landry A (2024a) (Trustworthy) AI for Québec's Virtual Power Plant. *Cahiers du GERAD,* IVADO Digital Futures 2024, Montréal, Québec, Canada

1.5 Notation

We use the following notation throughout this Master's thesis:

- Bold lowercase letters, e.g., $\mathbf{a}$, refer to vectors, bold uppercase letters, e.g., $\mathbf{A}$, refer to matrices, and normal lowercase or uppercase letters refer to scalars, e.g., $a,\ A$. Bold Greek letters may refer to matrices depending on context.
- The sets of real, natural, and integer numbers are denoted by $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{Z}$, respectively.
- Calligraphic uppercase letters refer to sets, e.g., $\mathcal{X}$ and $\mathcal{Z}$, or to functions depending on context, e.g., $\mathcal{L}(\cdot)$ and $\mathcal{V}(\cdot)$.
- Letters $\mathbf{x},\ \mathbf{y},$ and $\mathbf{z}$ are reserved for features, labels, and samples, respectively, while $\hat{\mathbf{y}}$ refers to model predictions.
- Let $\mathbf{a} \in \mathbb{R}^n$ be a general vector of dimension $n \in \mathbb{N}^*$, column vectors are compactly written $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and row vectors as $\mathbf{a}^\top = (a_1, a_2, \ldots, a_n)^\top = (a_1\ a_2\ \ldots\ a_n)$.
- Bracket notation $[\![\cdot]\!]$ is used as a concise alternative to $\{1, 2, \ldots, \cdot\}$.
- $\{\cdot\} \succeq 0$ refers to semidefinite positiveness if $\{\cdot\}$ is a matrix and is an element-wise ≥ 0 if $\{\cdot\}$ is a vector.
- $\mathbb{1}(\cdot) : \mathbb{R}^n \to \{0, 1\}^n$ is an element-wise indicator function of arbitrary dimension n which returns 1 or 0 element-wise if the condition is true or false, respectively, for each separate element.
- $\mathrm{vec}(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$ denotes the vectorization operator which flattens matrices. If $\mathbf{A} = (\mathbf{a}_1\ \mathbf{a_2}\ \ldots\ \mathbf{a_n})$ where $\mathbf{a}_i \in \mathbb{R}^m,\ n, m \in \mathbb{N}^*,\ \forall i \in [\![n]\!]$, then $\mathrm{vec}(\mathbf{A}) = (\mathbf{a}_1^\top\ \mathbf{a}_2^\top\ \ldots\ \mathbf{a}_n^\top)^\top$.
- $\mathrm{card}(\{\cdot\})$ refers to the cardinality of the set $\{\cdot\}$.
- $\mathbf{0}_n \in \mathbb{R}^n$ refers to the null vector of size n.
- $\|\cdot\|_1$ refers to the ℓ_1-norm , $\|\cdot\|_2$ refers to the ℓ_2-norm, the ℓ_1-loss refers to $\|\mathbf{Y} - \hat{\mathbf{Y}}\|_1$, and the ℓ_2-loss refers to $\|\mathbf{Y} - \hat{\mathbf{Y}}\|_2^2$ for $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ the matrices of all training labels and respective model predictions, respectively.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we complete a literature review of key concepts from each chapter.

2.1 Wasserstein distributionally robust shallow convex neural networks

We now review the bodies of literature closest to Chapter 3. We refer interested readers to Bishop and Nasrabadi (2006) and Bishop and Bishop (2023) for a full horizon of classical ML and modern deep learning as we will not cover concepts such as backpropagation and iterative methods to optimize non-convex neural networks, e.g., stochastic gradient descent (Amari 1993).

Convex training of neural networks. Convex optimization for machine learning (ML) training procedures is desirable as it guarantees convergence to the minimal loss. In recent years, Pilanci and Ergen (2020) have shown that training a feedforward shallow neural network (SNN), i.e., a one-hidden layer NN, with ReLU activation functions is equivalent to a convex training procedure. They also extended their results to convolutional neural networks (CNNs). More importantly, they fundamentally linked the ReLU-SNN training problem to a regularized convex program. Following this work, Mishkin et al. (2022) have obtained an efficient convex training procedure of SNNs with graphics processing unit (GPU) acceleration, and Kuelbs et al. (2024) have designed an adversarial training procedure by controlling the worst-case output of the neural network in a neighbourhood around each training samples. In this work, we generalize the regularization procedure of feedforward SNNs in regression tasks through the lens of distributionally robust optimization with the Wasserstein metric. The metric is introduced in Section 3.2.1.

Distributionally robust neural networks. Distributionally robust optimization (DRO), in the ML setting, aims to minimize the expected training loss of an ML model over the most adverse data distribution within a distance of the training set, i.e., the distribution inside a specified ambiguity set, constructed from the available training data, that generates the largest expected loss. As demonstrated by Kuhn et al. (2019), DRO minimizes the worst-case optimal risk associated with the training of the model. Different authors have worked on distributionally robust neural networks. Bai et al. (2023a) propose to train deep neural networks using the first-order approximation of the Wasserstein DRO loss function in the backpropagation but leave the numerical evaluation for future work. Wasserstein DRO uses the definition of the Wasserstein distance to construct the ambiguity set. We cover it in Section 3.2.1. The authors of Levy et al. (2020) have developed a DRO gradient estimator that integrates into neural networks' non-convex optimization pipeline similarly to

the widely used stochastic gradient descent (SGD) optimizer. In Sagawa et al. (2020), the authors propose a method to train deep NNs in a DRO fashion by defining the ambiguity set with groups of the training data and increasing regularization. These methods integrate distributional robustness in the training procedure of deep NNs but do not directly tackle the Wasserstein DRO problem, or an equivalent reformulation. They also all suffer from the lack of global optimality guarantees making them harder to deploy in large-scale critical applications. Our proposition, while only applying to shallow networks, works with the tractable reformulation of the Wasserstein DRO problem proposed by Mohajerin Esfahani and Kuhn (2018) and converges to a global optimum because of convexity.

Physics-informed neural networks. The training approach for physics-informed neural networks (PINNs) primarily involves a weighted multi-objective function designed to narrow the gap between labels and model predictions while penalizing physical constraint violations. This is often achieved by simultaneously minimizing the residuals of a partial differential equation (PDE) and its boundary conditions (Cuomo et al. 2022). Suppose all NN training weights are collected in $\mathbf{\Theta}$, a typical PINN training program takes the form:

$$\min_{\mathbf{\Theta}} \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}} \left(\Phi(\mathbf{X}, \mathbf{\Theta})\right) + \lambda_{\mathcal{B}} \mathcal{L}_{\mathcal{B}} \left(\Phi(\mathbf{X}, \mathbf{\Theta})\right) + \lambda_{\text{data}} \mathcal{L} \left(\Phi(\mathbf{X}, \mathbf{\Theta}), \mathbf{Y}\right)$$

where $\mathcal{L}_{\mathcal{F}}$ and $\mathcal{L}_{\mathcal{B}}$ are the functions enforcing the differential equations and boundary conditions, respectively, of the governing physical system, $\mathcal{L}$ is the regression objective between labeled data and model predictions, and $\hat{\mathbf{Y}} = \Phi(\mathbf{X}, \mathbf{\Theta})$ (Cuomo et al. 2022). The scalar weights $\lambda_{\mathcal{F}}$, $\lambda_{\mathcal{B}}$, and λ_{data} must be tuned empirically. We note that the constraints are softly enforced by penalizing their violation in the objective. It is similar to considering the Lagrangian relaxation of the constrained training problem. The loss terms related to the PDE are derived through the automatic differentiation of the neural network's outputs (Baydin et al. 2017). This complex loss function is then iteratively optimized through backpropagation within the network. While Cuomo et al. (2022) note that boundary conditions can be strictly imposed during training by forcing a part of the network to satisfy these conditions, see Zhu et al. (2021), they are typically integrated as penalty terms within the loss function. By using the augmented Lagrangian in the objective function and increasing the penalty terms related to constraint violations iteratively, it is theoretically possible to converge to a PINN which imposes hard inequality constraints during training (Lu et al. 2021). Physics-informed neural networks are characterized by their extensive modelling capabilities, facilitated by automatic differentiation, that allows for the approximation of continuous PDEs across various neural network architectures, e.g., CNNs (Gao et al. 2021), long short-term memory networks (Zhang et al. 2020), Bayesian NNs (Yang et al. 2021), and generative adversarial networks (Yang et al. 2020). Although limited to convex constraints, in Chapter 3, we demonstrate that our training methodology can en-

force hard constraints on the shallow network's output during training without complexifying its architecture or objective function. These constraints may include, but are not limited to, boundary constraints, ramping constraints, discrete PDEs, and constraints derived from discrete-time dynamical systems.

Application to non-residential building energy consumption. As stated previously, prediction algorithms used in large-scale VPP applications must respect some inherent constraints. They need to be computationally scalable, they must be able to model nonlinearities, and the sensibility of the application requires trustworthiness in their predictions, e.g., robustness, interpretability, and performance guarantees. Regarding the forecast of non-residential buildings' energy consumption, because grid operators do not generally have access to information on building activity, occupancy, and control, the system is approached as a blackbox and data-driven approaches tend to be the default option (Amasyali and El-Gohary 2018). Authors have used support vector regression (SVR) (Dong et al. 2005) and other hybrid methods leveraging both classification and linear regression to satisfy the aforementioned constraints (Amasyali and El-Gohary 2018). These methods are linear by parts and can approximate nonlinear distributions adequately. Statistical methods such as autoregressive integrated moving average (ARIMA) (Newsham and Birt 2010) and Gaussian processes have also proved to be good contenders as they even offer empirical uncertainty bounds (Weng and Rajagopal 2015). Many authors have proposed deep-learning methods, yet these propositions do not have much industrial acceptability as they fail to meet the industrial constraints and bloat the requirements for a safe ML pipeline. In Chapter 3, we show that our proposition is interesting for virtual power plant applications where the forecast of many distributed assets is used for critical decision-making as we satisfy all industrial constraints better than other comparable models.

Outlier generation. Distributionally robust learning, in general, aims at protecting machine learning models from out-of-distribution outliers in the training set. These outliers tend to be adversarial to the prediction quality of ML models in validation and testing sets, and being able to ignore them should increase models' out-of-sample performance. As such, to correctly benchmark distributionally robust machine learning models, one must be able to generate outliers that are indeed out-of-distribution with respect to the original distribution and be able to do it regardless of the form of the original distribution. In the literature, we found that methodologies to achieve these two objectives in regression tasks are limited. For example, Chen and Paschalidis (2018) create a training set by randomly drawing samples from two distributions, the inlying and outlying one, under probability $p \in (0, 1)$ and $1-p$, respectively. They used Gaussian distributions for both. In the work of Qi et al. (2022), authors task their model with the forecast of a real-world dataset. To control the adversity of the setting, they discard the real labels and create a synthetic version of them by using the predic-

tion of a linear regression trained on the original dataset. They then tune an additive noise over the labels to simulate different levels of outlier corruption. In (Liu et al. 2022), they notably propose generating corrupted synthetic data by integrating attack mechanisms into the generation to bias their dataset, i.e., selection bias and anti-causal bias. While their methodology is interesting, they must rely on multiple Gaussian distributions to design their attacks as intended. By leveraging the sliced-Wasserstein distance (Bonnotte 2013), a tractable approximation of the Wasserstein distance, our methodology generates outliers, does not rely on any assumptions of the original distribution, making it quite flexible, and ensures that outliers do not create tangible patterns.

2.2 Sliced-Wasserstein-based anomaly detection

We now continue by reviewing the literature associated to Chapter 4 and by presenting some context on localized critical peak rebates.

Anomaly detection. As hinted in Chapter 1 anomaly detection (AD) and outlier filtering are primordial for training data selection in a reliable ML pipeline. Unsupervised AD methods are preferable as they do not need human-made labels and their hyperparameters can be tuned simultaneously with other ML models' included in the loop. Popular unsupervised AD methods (Goldstein and Uchida 2016) include local outlier factor (LOF) (Breunig et al. 2000), isolation forest (Liu et al. 2008), k-nearest neighbours (KNN) (Angiulli and Pizzuti 2002), connectivity-based outlier factor (Tang et al. 2002), and one-class support vector machine (SVM) (Amer et al. 2013). These methods either use clustering or local density to assign an outlier score.

We are interested in optimal transport-based (OT) metrics for AD. Ducoffe et al. (2019) proposed a Wasserstein generative adversarial network for AD tasks and Wang et al. (2024) designed a differentiable training loss function using the Wasserstein metric for deep classifiers.

To the best of our knowledge, no unsupervised OT method has been proposed yet for AD.

Localized critical peak rebates. Québec, Canada, stands as an outlier in the electrical grid decarbonization paradigm. As there is a global tendency to invest in intermittent renewable energy sources to decarbonize grids worldwide, Québec, while not totally stranger to this trend, is generally mostly carbon-free thanks to its impressive hydroelectric power capacity. One of its main challenges comes from its northern climate, its reliance on electric baseboard heating systems for residential heating, and its unrestrictive home insulation policies (Gerbet 2023). During glacial winter days, as electric heaters are all running at once, and peak consumption hours hit, Québec's hydropower capacity can be reached (Whitmore and Pineau 2024). To accommodate winter peak

power needs, Hydro-Québec, the state-owned company in charge of electric power generation, transmission, and distribution, must operate its only on-grid thermal power plant and import electricity from neighbouring provinces and states. These imports are usually expensive and produce much more greenhouse gas emissions in comparison to local energy sources (Hydro-Québec 2023).

To remediate this issue without solely relying on the deployment of new generation and transmission infrastructures, demand response (DR) initiatives have flourished in the province (Pelletier and Faruqui 2022). With Québec's long tradition of fixed electricity rates, one of its main DR mechanisms is critical peak rebates (CPR). Residential customers enrolled in a CPR program receive financial compensation during pre-specified time periods, referred to as challenges, for reducing their energy consumption with respect to their expected baseload (Mercado et al. 2014). CPR programs are purely voluntary and virtually penalty-free. They thus depend on consumers' goodwill, motivation, and sensitivity. Yet, as we have seen in our work, CPR events can be powerful tools for shifting power consumption before and after each event.

We work with a variation of CPR, viz., localized CPR (LCPR), in which the events are called for localized relief in the grid instead of being cast for the whole system. LCPR can diversify the types of services offered by typical CPR programs, e.g., they can alleviate stress on local equipment like substation transformers (Li et al. 2024) or they can be paired with generation forecasts of distributed energy resources (DERs), e.g., roof solar panels and private windfarms, to balance demand and generation during peak hours. LCPR is highly underexplored in the literature and is a valuable application to benchmark trustworthy machine learning models. Indeed, the higher spatial granularity, the critical aspect of the task, the dependence on behavioural tendencies, and the lower margin for error require the deployment of forecasting models that offer performance guarantees (Venzke and Chatzivasileiadis 2021), robustness to noise (Chen and Paschalidis 2020), interpretability (Gilpin et al. 2018), physical constraints satisfaction (Misyris et al. 2020), a sense of prediction confidence (Jospin et al. 2022, Wilson and Izmailov 2020), or a combination of them (Pallage and Lesage-Landry 2024b). Being able to predict and utilize localized peak-shaving potential in the electrical grid, through programs similar to LCPR, could accelerate the integration of DERs and, specifically for Québec, phase out its dependence on fossil energy-based imports. There are thus concrete incentives for VPPs to develop and deploy this service. To the best of our knowledge, no published open-source datasets showcase either LCPR or CPR schemes in a northern climate.

CHAPTER 3 WASSERSTEIN DISTRIBUTIONALLY ROBUST SHALLOW CONVEX NEURAL NETWORKS

In this chapter, we present the first part of our work. The results and general structure are adapted from Pallage and Lesage-Landry (2024b).

Conceptually, our propositions contribute to the literature on trustworthy training and post-training certification. As a reminder, they are highlighted in Figure 3.1.

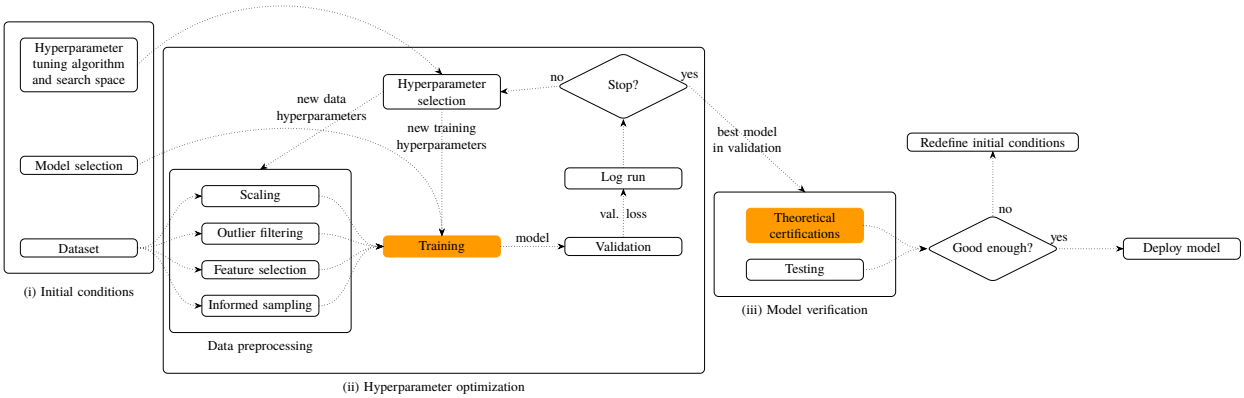


Figure 3.1 Concepts from the trustworthy ML pipeline covered in Chapter 3

3.1 Contributions

In this chapter, we propose a new distributionally robust convex training procedure for ReLU shallow neural networks under the Wasserstein metric. Under some conditions, we show that the standard ReLU shallow convex neural network (SCNN) training can be decoupled and formulated as a constrained linear regression over a modified sample space. SCNN training requires a set of *sampling* vectors, which are known a priori, to model possible activation patterns. We condition over this set to construct a new independent and identically distributed conditional empirical distribution based on the modified sample space. Decoupling the training and working with this new empirical distribution allows us to easily adapt the SCNN training to the tractable formulation of the Wasserstein-DRO problem proposed by Mohajerin Esfahani and Kuhn (2018), derive out-of-sample performance guarantees stemming from its Rademacher complexity, and enforce hard convex physical constraints during training.

Additionally, we propose a first post-training verification program specifically tailored to SCNNs. We model the inner workings of SCNNs as a mixed-integer convex optimization problem to cer-

tify the worst-case local instability in the feature space of a trained SCNN. This post-verification framework enables us to evaluate how different regularization paradigms, such as distributionally robust learning, influence the stability of trained SCNNs without having to map their weights to their non-convex equivalents. This program also opens research opportunities for bilevel programs where SCNN certification and training are tackled simultaneously.

We propose a new methodology to benchmark distributionally robust (and regularized) machine learning models in a controlled synthetic environment. We design a method to introduce out-of-sample outliers under the sliced-Wasserstein distance (Bonnotte 2013), a computationally tractable approximation of the Wasserstein distance. We combine this method with noise to corrupt machine learning datasets generated with standard optimization benchmark functions typically reserved for non-convex solver benchmarking. We show that our Wasserstein distributionally robust SCNN (WaDiRo-SCNN) training performs well empirically in highly corrupted environments: it is robust and conservative while being able to model non-convex functions. Finally, we test our model on a real-world VPP application, viz., the hourly energy consumption prediction of non-residential buildings from Montréal, Canada. For this application, we enforce the physical constraint that energy consumption is non-negative. We observe that this constraining greatly reduces the number of constraint violations during testing while having no significant precision drop with the original non-physics constrained WaDiRo-SCNN.

To the best of our knowledge, the specific contributions of this chapter are as follows:

- We demonstrate that the SCNN with ℓ_1-loss training problem is similar to a constrained linear regression over a modified sample space.
- We formulate an exact, tractable, and low-stochasticity Wasserstein distributionally robust training program for shallow convex neural networks under both ℓ_1-norm and ℓ_2-norm order-1 Wasserstein distances by exploiting this constrained linear regression formulation and linking it with past literature.
- We derive out-of-sample guarantees for WaDiRo-SCNNs with ℓ_1-loss training via the Rademacher complexity under its original sample space and its modified one.
- We show that hard convex physical constraints can be easily included in the training procedure of SCNNs with ℓ_1-norm loss and their WaDiRo counterpart.
- We design a mixed-integer convex program (MICP) for the post-training verification of SCNNs. We utilize it to show how distributionally robust and regularized training methods increase model stability empirically.

- We propose a new benchmarking procedure with synthetic data to empirically evaluate the robustness of ML models regarding out-of-distribution outliers.
- We illustrate the conservatism of WaDiRo-SCNNs in predicting synthetic benchmark functions with controlled data corruption and its performance in predicting real-world non-residential buildings' consumption patterns.

The rest of this section is organized as follows. In Section 3.2, we introduce Wasserstein distributionally robust optimization and its tractable formulation for convex problems (Section 3.2.1) as well as shallow convex neural networks (Section 3.2.2). In Section 3.3 we present the main results of this chapter, viz., WaDiRo-SCNNs. In Sections 3.4, 3.5, and 3.6 we address the theoretical out-of-sample performance of our proposition, physics-constrained SCNNs, and SCNN post-training verification, respectively. Finally, in Section 3.7 we present a methodology to generate outliers and in Section 3.8, we present all numerical experiments.

3.2 Preliminaries

We start by covering the fundamentals of Wasserstein distributionally robust optimization before introducing shallow convex neural networks.

3.2.1 Distributionally robust optimization

Distributionally robust optimization (DRO) minimizes a worst-case expected loss function over an ambiguity set, i.e., an uncertainty set in the space of probability distributions (Kuhn et al. 2019). The problem can be formulated as:

$$\inf_{\boldsymbol{\beta}} \sup_{\mathbb{Q}\in\Omega} \mathbb{E}^{\mathbb{Q}}[h_{\boldsymbol{\beta}}(\mathbf{z})], \tag{DRO}$$

where $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^d, d \in \mathbb{N}$, is a random vector, $\mathcal{Z}$ is the set of all possible values of $\mathbf{z}$, $\mathbb{Q}$ is a distribution from the ambiguity set Ω that characterize the distribution of the random vector $\mathbf{z}$, $\boldsymbol{\beta} \in \mathbb{R}^p$ is the p-dimension decision vector, and $h_{\boldsymbol{\beta}}(\mathbf{z}) : \mathcal{Z} \times \mathbb{R}^p \to \mathbb{R}$ is the objective function when applying $\boldsymbol{\beta}$ on a sample (Chen and Paschalidis 2020, Mohajerin Esfahani and Kuhn 2018).

We are interested in a data-driven branch of DRO which defines the ambiguity set as a Wasserstein ball, a ball which contains all distributions within a Wasserstein distance from a reference distribution (Yue et al. 2022), centered on an empirical distribution containing N samples: $\hat{\mathbb{P}}_N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\mathbf{z}_i}$, where $\delta_{\mathbf{z}_i}$ is a Dirac delta function assigning a probability mass of one at each

known sample $\mathbf{z}_i$. The empirical distribution is constructed with historical realizations of the random vector $\mathbf{z}$. Note that as the number of samples goes to infinity the empirical distribution converges to the true distribution $\mathbb{P}$ (van der Vaart 2000, Chapter 19.2). The Wasserstein distance, also called earth mover's distance and closely related to the optimal transport plan (Kuhn et al. 2019), can be pictured as the minimal effort that it would take to displace a given pile of a weighted resource, e.g., dirt, to shape a second specific pile. The effort of a trip is given by the distance traveled and the weight of the carried resource. Specifically, it provides a sense of similarity between the two distributions. Formally, the probabilistic definition of the order-t Wasserstein distance between probability measures $\mathbb{U}$ and $\mathbb{V}$ on $\mathcal{Z}$ is:

$$W_{\|\cdot\|,t}(\mathbb{U},\mathbb{V}) = \inf_{\substack{U\sim\mathbb{U},\\ V\sim\mathbb{V}}} \left(\mathbb{E}[\|U-V\|^t]\right)^{\frac{1}{t}}, \quad t \geq 1,$$

where the infimum is taken over all possible couplings of random vectors U and V which are marginally distributed as $\mathbb{U}$ and $\mathbb{V}$, respectively (Panaretos and Zemel 2019). As such, the Wasserstein ball defines a space containing all possible distributions close to the distribution in its center. We refer interested readers to the open-access book of Panaretos and Zemel (2020) which covers the Wasserstein distance thoroughly while offering interesting statistical insights.

In this context, (DRO) is expressed as:

$$\begin{aligned} &\inf_{\beta} \sup_{\mathbb{Q}\in\Omega} \quad \mathbb{E}^{\mathbb{Q}}[h_\beta(\mathbf{z})] \qquad \text{(WDRO)}\\ &\text{with } \Omega = \{\mathbb{Q}\in\mathcal{P}(\mathcal{Z}) : W_{\|\cdot\|,t}(\mathbb{Q},\hat{\mathbb{P}}_N) \leq \epsilon\}, \end{aligned}$$

where Ω is the order-t Wasserstein ball of radius $\epsilon > 0$ centered on $\hat{\mathbb{P}}_N$, and $\mathcal{P}(\mathcal{Z})$ is the set of all probability distributions with support on $\mathcal{Z}$. In a way, ϵ can be interpreted as a hyperparameter that controls our conservatism towards $\hat{\mathbb{P}}_N$ since a greater radius admits distributions farther from $\hat{\mathbb{P}}_N$ into the ambiguity set. This problem is not tractable (Mohajerin Esfahani and Kuhn 2018).

Using the strong duality theorem and the order-1 Wasserstein metric, Mohajerin Esfahani and Kuhn (2018) show that an exact tractable reformulation of (WDRO) is obtainable if $h_{\boldsymbol{\beta}}(\mathbf{z})$ is convex in $\mathbf{z}\in\mathcal{Z}$ and if $\mathcal{Z} = \mathbb{R}^d$, where d is the dimension of the samples. The problem to consider then becomes:

$$\begin{aligned} \inf_{\boldsymbol{\beta}} \sup_{\mathbb{Q}\in\Omega} \mathbb{E}^{\mathbb{Q}}[h_{\boldsymbol{\beta}}(\mathbf{z})] = &\inf_{\boldsymbol{\beta}} \kappa\epsilon + \frac{1}{N}\sum_{i=1}^{N} h_{\boldsymbol{\beta}}(\mathbf{z}_i) \qquad \text{(TWDRO)}\\ &\text{s.t. } \kappa = \sup\{\|\boldsymbol{\theta}\|_* : h^*_{\boldsymbol{\beta}}(\boldsymbol{\theta}) < +\infty\}, \end{aligned}$$

where $\|\cdot\|_*$ is the dual norm $\|\boldsymbol{\theta}\|_* \equiv \sup_{\|\mathbf{z}\|\leq 1} \boldsymbol{\theta}^\top\mathbf{z}$ and $h^*_{\boldsymbol{\beta}}(\boldsymbol{\theta})$ is the convex conjugate of $h_{\boldsymbol{\beta}}(\mathbf{z})$,

$h^*_{\boldsymbol{\beta}}(\boldsymbol{\theta}) = \sup_{\mathbf{z}}\{\boldsymbol{\theta}^\top \mathbf{z} - h_{\boldsymbol{\beta}}(\mathbf{z})\}$ (Boyd and Vandenberghe 2004).

Training machine learning models with (TWDRO) has been proved to be a generalization of regularization (Shafieezadeh-Abadeh et al. 2019). By training over the worst-case expected distribution in a Wasserstein ball centered on the training dataset, we can expect to avoid overfitting on the empirical distribution and yielding increased robustness against outliers, noise, and some adversarial attacks.

3.2.2 Shallow convex neural networks

Let $\mathbf{X} \in \mathbb{R}^{N\times d}$ be a data matrix of N feature vectors $\mathbf{x}_j \in \mathbb{R}^d \; \forall j \in [\![N]\!]$, and $\mathbf{y} \in \mathbb{R}^N$ be the corresponding one-dimensional labels. Define data samples as a feature-label pair: $\mathbf{z}_j = (\mathbf{x}_j, y_j) \; \forall j \in [\![N]\!]$. The non-convex training problem of a single-output feedforward shallow neural network (SNN) with ReLU activation functions and without bias weights is as follows:

$$p^\star = \min_{\mathbf{W}_1, \mathbf{w}_2} \mathcal{L}\left(\sum_{i=1}^{m} \{(\mathbf{X}\mathbf{W}_{1i})_+ w_{2i}\}, \mathbf{y}\right), \tag{SNNT}$$

where m is the number of hidden neurons, $\mathbf{W}_1 \in \mathbb{R}^{d\times m}$, and $\mathbf{w}_2 \in \mathbb{R}^m$ are the weights of the first and second layers, $\mathcal{L} : \mathbb{R}^N \to \mathbb{R}$ is a convex loss function, and $(\cdot)_+$ is an element-wise $\max\{0, \cdot\}$ operator (ReLU).

The convex reformulation of the SNN training problem (SNNT) is based on an enumeration process of all the possible ReLU activation patterns in the hidden layer for a fixed training dataset $\mathbf{X}$ (Pilanci and Ergen 2020). The enumeration is equivalent to generating all possible hyperplane arrangements passing through the origin and clustering the data with them. The set of all possible activation patterns a ReLU SNN can take for a training set $\mathbf{X}$ is given by:

$$\mathcal{D}_{\mathbf{X}} = \left\{\mathbf{D} = \mathrm{diag}(\mathbb{1}(\mathbf{X}\mathbf{s} \succeq \mathbf{0})) : \mathbf{s} \in \mathbb{R}^d\right\}, \tag{3.1}$$

where $\mathbf{D}$ is a diagonal matrix with a possible activation pattern encoded on the diagonal.

By introducing the cone-constrained variables $\boldsymbol{\nu}, \boldsymbol{\omega} \in \mathbb{R}^{P\times d}$ and the diagonal matrices from $\mathcal{D}_{\mathbf{X}}$, Pilanci and Ergen (2020) have obtained a convex training procedure from which we can retrieve the previous non-convex weights under some conditions. The optimization problem is now defined as

$$\begin{aligned} p^\star = \min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & \mathcal{L}\left(\sum_{i=1}^{P} \mathbf{D}_i \mathbf{X}(\boldsymbol{\nu}_i - \boldsymbol{\omega}_i), \mathbf{y}\right) \\ \text{s.t.} \quad & \boldsymbol{\nu}_i, \boldsymbol{\omega}_i \in \mathcal{K}_i \quad \forall i \in [\![P]\!], \end{aligned} \tag{SCNNT}$$

where $\mathcal{K}_i = \{\mathbf{u} \in \mathbb{R}^d : (2\mathbf{D}_i - \mathbf{I})\mathbf{X}\mathbf{u} \succeq \mathbf{0}\}$ and $P = \text{card}(\mathcal{D}_\mathbf{X})$. For $m \geq m^\star = \sum_{i:\boldsymbol{\nu}_i \neq \mathbf{0}}^{P} 1 + \sum_{i:,\boldsymbol{\omega}_i \neq \mathbf{0}}^{P} 1$, with $1 \leq m^\star \leq N + 1$, problems (SNNT) and (SCNNT) have identical optimal values (Pilanci and Ergen 2020). At least one optimal non-convex NN from (SNNT) can be assembled, using $m^\star$ neurons, with the optimal solution of (SCNNT) and the following mapping between the optimal weights of each problem:

$$(\mathbf{W}_{1i}^\star, \mathbf{w}_{2i}^\star) = \left(\frac{\boldsymbol{\nu}_i^\star}{\sqrt{\|\boldsymbol{\nu}_i^\star\|_2}}, \sqrt{\|\boldsymbol{\nu}_i^\star\|_2} \right), \qquad \text{if} \quad \boldsymbol{\nu}_i^\star \neq \mathbf{0}, \forall i \in [\![P]\!] \tag{3.2}$$

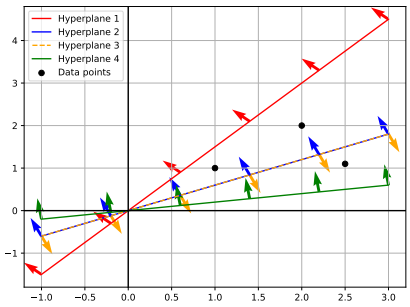
$$\left(\mathbf{W}_{1(i+P)}^\star, \mathbf{w}_{2(i+P)}^\star\right) = \left(\frac{\boldsymbol{\omega}_i^\star}{\sqrt{\|\boldsymbol{\omega}_i^\star\|_2}}, -\sqrt{\|\boldsymbol{\omega}_i^\star\|_2} \right), \quad \text{if} \quad \boldsymbol{\omega}_i^\star \neq \mathbf{0}, \forall i \in [\![P]\!]. \tag{3.3}$$

We note that Wang et al. (2021) demonstrate how to recover the full optimal set of non-convex networks. To better understand the mechanism proposed by Pilanci and Ergen (2020), we propose the following example inspired by Pilanci (2022) for a dataset $\mathbf{X}$ of three samples with two colinear samples.

Example 1. *Consider the following dataset:*

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} = \begin{bmatrix} (1.0, 1.0) \\ (2.0, 2.0) \\ (1.1, 2.5) \end{bmatrix}.$$

We can generate only four clustering schemes from this dataset using (3.1). The hyperplane arrangements and their diagonal matrices are presented in Figure 3.2.



(a) Hyperplane arrangements

$$\mathbf{D}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{D}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{D}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{D}_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) Corresponding diagonal matrices

Figure 3.2 Supporting figure for Example 1

For this specific dataset, by expanding each $\mathbf{D}_i\mathbf{X}$ *product, we obtain the following (*SCNNT*):*

$$p^\star = \min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad \mathcal{L}\left(\begin{bmatrix}\mathbf{x}_1^\top\\ \mathbf{x}_2^\top\\ 0\end{bmatrix}(\boldsymbol{\nu}_2-\boldsymbol{\omega}_2)+\begin{bmatrix}0\\ 0\\ \mathbf{x}_3^\top\end{bmatrix}(\boldsymbol{\nu}_3-\boldsymbol{\omega}_3)+\begin{bmatrix}\mathbf{x}_1^\top\\ \mathbf{x}_2^\top\\ \mathbf{x}_3^\top\end{bmatrix}(\boldsymbol{\nu}_4-\boldsymbol{\omega}_4),\mathbf{y}\right)$$
$$\text{s.t.} \quad \boldsymbol{\nu}_i,\boldsymbol{\omega}_i \in \mathcal{K}_i \quad \forall i \in [\![P]\!],$$

where $\mathcal{K}_i = \{\mathbf{u}\in\mathbb{R}^d : (2\mathbf{D}_i-\mathbf{I})\mathbf{X}\mathbf{u}\succeq 0\}$ *and* $P=4$*, This problem is equivalent to (*SNNT*) with respect to* $\mathbf{X}$*.*

To make the problem computationally tractable, we can approximate the full hyperplane arrangements by sampling random vectors (Mishkin et al. 2022). For example, using a Gaussian sampling, we substitute $\mathcal{D}_\mathbf{X}$ by $\tilde{\mathcal{D}}_\mathbf{X} = \{\mathbf{D}_i = \text{diag}(\mathbb{1}(\mathbf{X}\mathbf{S}_i \succeq \mathbf{0})) : \mathbf{S}_i \sim \mathcal{N}_d(\boldsymbol{\mu},\boldsymbol{\Sigma}), \mathbf{D}_i \neq \mathbf{D}_{j\neq i}, i \in [\![P]\!]\}$ where $\mathbf{S}_i$ is a random Gaussian vector of mean $\boldsymbol{\mu}\in\mathbb{R}^d$ and of covariance $\boldsymbol{\Sigma}\in\mathbb{R}^{d\times d}$, and such that $\mathbf{s}_i\in\mathbb{R}^d$ is its realization, $\forall i\in[\![P]\!]$. This approximation has the same minima as (SCNNT) if

$$m \geq b \coloneqq \sum_{\mathbf{D}_i\in\tilde{\mathcal{D}}_\mathbf{X}} \text{card}\left(\{\boldsymbol{\nu}_i^\star : \boldsymbol{\nu}_i^\star \neq \mathbf{0}\}\cup\{\boldsymbol{\omega}_i^\star : \boldsymbol{\omega}_i^\star \neq \mathbf{0}\}\right),$$

where in this case $\boldsymbol{\nu}^\star, \boldsymbol{\omega}^\star$ are global optima of the sub-sampled convex problem and if the optimal activations are in the convex model, i.e., $\{\text{diag}(\mathbf{X}\mathbf{W}_{1i}^\star \geq 0) : i\in[\![m]\!]\} \subseteq \tilde{\mathcal{D}}_X$ (Mishkin et al. 2022). Interested readers are referred to Pilanci and Ergen (2020) and Mishkin et al. (2022) for the detailed coverage of SCNNs.

We remark that if these conditions are met, for a feature sample $\mathbf{x}$ not included in the training dataset $\mathbf{X}$, then the non-convex formulation of the neural network gives the same output as the convex formulation when using the same set of realized sampling vectors $\mathcal{S} = \{\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_P\}$. This notion will be useful in the next section.

If these conditions are not met, i.e., the SNN is not wide enough, we still obtain an approximately optimal network that can outperform conventional training methods as highlighted by Mishkin et al. (2022). By approximate optimality, we mean that it is optimized by a convex program that guarantees optimality while not having an exact mapping to the optimal non-convex neural network formulation. Thus, choosing the maximal width of the SCNN, which depends on P, is a way to control the tractability of the convex training with respect to the available computational power and the size of the training dataset.

3.3 Main proposition

We now introduce WaDiRo-SCNNs and provide tractable procedures to train them. First, we reexpress the training objective function of the SCNN in a form compatible with the tractable formulation of the Wasserstein DRO problem from Mohajerin Esfahani and Kuhn (2018) to obtain performance guarantees for critical applications.

We make the following assumption on samples from $\mathbb{P}$.

Assumption 1. *Samples* $\mathbf{z} = (\mathbf{x}, y) \in \mathcal{Z} = \mathbb{R}^{d+1}$ *are independent and identically distributed ().*

This is a common assumption in statistical machine learning which is also made by, e.g., Chen and Paschalidis (2020) and Mohajerin Esfahani and Kuhn (2018). In practice, in learning settings, we only have access to a finite amount of training samples from which we can indirectly observe $\mathbb{P}$. This motivates the use of a finite empirical distribution $\hat{\mathbb{P}}_N$ that approximates $\mathbb{P}$ (Kuhn et al. 2019). As such, we assume that our empirical samples are drawn independently from a consistent unknown distribution which is When formulating (`TWDRO`), only the independence of the samples is necessary to build $\hat{\mathbb{P}}_N$. The stronger identically distributed assumption is then only required to bound the out-of-sample performance, see Section 3.4. As a result, the later part of Assumption 1 implies that any distribution drift is neglected from the analysis. While Assumption 1 is not always preserved in practice, e.g., time-series forecasting, it is required for the theoretical analysis. Recall (3.1) and let $d_{ji} = \mathbf{D}_i(j,j) = \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_i \geq \mathbf{0}),\ \forall j \in [\![N]\!],\ i \in [\![P]\!]$ such that $\mathbf{d}_j \in \{0,1\}^P$ is the vector encoding the activation state of $\mathbf{x}_j$ for each vector in $\mathcal{S}$. For the remainder of this section, we consider the modified features $\hat{\mathbf{x}}_j = \left(\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top), \mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top)\right)\ \forall j \in [\![N]\!]$, which are a higher order basis expansion of the original features with their respective possible activation patterns. This basis expansion is justified in the proof of the following lemma.

We now proceed to decouple the training procedure for distinct samples.

Lemma 1. *Suppose that the training loss function* $\mathcal{L}$ *is the* ℓ_1*-norm, and let* $\hat{\mathbf{z}}_j = (\hat{\mathbf{x}}_j, y_j)\ j \in [\![N]\!]$ *be modified samples. Then, the convex training procedure (`SCNNT`) is equivalent to a constrained linear regression problem.*

Proof: Define $\mathbf{u}_i = \boldsymbol{\nu}_i - \boldsymbol{\omega}_i$ and let $\mathbf{U} \in \mathbb{R}^{P\times d}$ be the matrix collecting each $\mathbf{u}_i^\top, \forall j \in [\![N]\!], \forall i \in$

$[\![P]\!]$. Considering the ℓ_1-norm loss function, we have:

$$\begin{aligned}
\mathcal{L}\left(\sum_{i=1}^{P}\mathbf{D}_i\mathbf{X}(\boldsymbol{\nu}_i-\boldsymbol{\omega}_i),\mathbf{y}\right) &= \left\|\left(\sum_{i=1}^{P}d_{1i}\mathbf{x}_1^\top\mathbf{u}_i,\sum_{i=1}^{P}d_{2i}\mathbf{x}_2^\top\mathbf{u}_i,\ldots,\sum_{i=1}^{P}d_{Ni}\mathbf{x}_N^\top\mathbf{u}_i\right)-\mathbf{y}\right\|_1 \\
&= \sum_{j=1}^{N}\left|\mathbf{d}_j^\top\mathbf{U}\mathbf{x}_j-y_j\right| \\
&= \sum_{j=1}^{N}|x_{j1}(d_{j1}u_{11}+\ldots+d_{jP}u_{P1})+\ldots+ \\
&\qquad\qquad x_{jd}(d_{j1}u_{1d}+\ldots+d_{jP}u_{Pd})-y_j| \\
&= \sum_{j=1}^{N}\left|(\mathrm{vec}(\mathbf{U}),-1)^\top\left(\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top),y_j\right)\right| \\
&= \sum_{j=1}^{N}\left|(\mathrm{vec}(\boldsymbol{\nu}),\mathrm{vec}(-\boldsymbol{\omega}),-1)^\top\left(\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top),\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top),y_j\right)\right| \\
&= \sum_{j=1}^{N}\left|\boldsymbol{\beta}^\top\hat{\mathbf{z}}_j\right|,
\end{aligned}$$

where $\boldsymbol{\beta} = (\mathrm{vec}(\boldsymbol{\nu}),\mathrm{vec}(-\boldsymbol{\omega}),-1)$ represent the weights of the linear regression and $\hat{\mathbf{z}}_j = \left(\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top),\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top),y_j\right) = (\hat{\mathbf{x}}_j,y_j)\ \forall j\in[\![N]\!]$ are the modified samples. The training problem (SCNNT) can be written as:

$$\begin{aligned}
p_{\ell_1}^\star &= \min_{\boldsymbol{\nu},\boldsymbol{\omega}:\boldsymbol{\nu}_i,\boldsymbol{\omega}_i\in\mathcal{K}_i\ \forall i\in[\![P]\!]}\left\|\mathbf{y}-\sum_{i=1}^{P}\mathbf{D}_i\mathbf{X}(\boldsymbol{\nu}_i-\boldsymbol{\omega}_i)\right\|_1 \\
&= \min_{\boldsymbol{\nu},\boldsymbol{\omega}:\boldsymbol{\nu}_i,\boldsymbol{\omega}_i\in\mathcal{K}_i\ \forall i\in[\![P]\!]}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^\top\hat{\mathbf{z}}_j\right| \\
&\qquad\qquad \text{s.t.} \qquad \boldsymbol{\beta} = (\mathrm{vec}(\boldsymbol{\nu}),\mathrm{vec}(-\boldsymbol{\omega}),-1)\,,
\end{aligned}$$

and is equivalent to a constrained linear regression problem with respect to the modified samples. □

Additionally, we note that the LASSO and Ridge regularized SCNN training problems are exactly equivalent to their respective linear regression formulation.

$$\begin{aligned}
p_{\ell_1,r}^\star &= \min_{\boldsymbol{\nu},\boldsymbol{\omega}:\boldsymbol{\nu}_i,\boldsymbol{\omega}_i\in\mathcal{K}_i\ \forall i\in[\![P]\!]}\left\|\mathbf{y}-\sum_{\mathbf{D}_i\in\mathcal{D}_\mathbf{X}}\mathbf{D}_i\mathbf{X}(\boldsymbol{\nu}_i-\boldsymbol{\omega}_i)\right\|_1+\sum_{i=1}^{P}\left(r(\boldsymbol{\nu}_i)+r(\boldsymbol{\omega}_i)\right) \\
&= \min_{\boldsymbol{\nu},\boldsymbol{\omega}:\boldsymbol{\nu}_i,\boldsymbol{\omega}_i\in\mathcal{K}_i\ \forall i\in[\![P]\!]}\sum_{j=1}^{N}\left|y_j-(\mathrm{vec}(\boldsymbol{\nu}),\mathrm{vec}(-\boldsymbol{\omega}))^\top\hat{\mathbf{x}}_j\right|+r\left([\mathrm{vec}(\boldsymbol{\nu}),\mathrm{vec}(-\boldsymbol{\omega})]\right),
\end{aligned}$$

where $r(\cdot) = \|\cdot\|_1$ for LASSO regularization and $r(\cdot) = \|\cdot\|_2^2$ for Ridge regularization.

The ℓ_1-norm loss function tends to be more robust against outliers than other regression objectives when training regression models (Chen and Paschalidis 2020). It is then a preferable choice in an application where distributional robustness is desired and will be used for the remainder of this chapter as it is also convenient in the WaDiRo reformulation. Nonetheless, we remark that Lemma 1 can be adapted to other separable loss functions, i.e., functions f such that $f(\phi_1(x_1) + \phi_2(x_2) + \ldots \phi_N(x_N)) = f(\phi_1(x_1)) + f(\phi_2(x_2)) + \ldots + f(\phi_N(x_N))$ where $x_j\ j \in [\![N]\!]$ are variables and $\phi_j\ j \in [\![N]\!]$ are arbitrary functions.

We remark that the transformed feature space preserves the independence of individual samples when conditioned on $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_P\}$ because the mapping from the original feature space to the modified one depends only on individual samples and $\mathcal{S}$. Whether $\mathcal{S}$ is constructed by random sampling or a by deterministic hyperplane arrangement has no importance as long as these vectors are saved prior to training and are used throughout the training and evaluation. From this observation, we now introduce our second lemma.

Lemma 2. *Suppose Assumption 1 holds, i.e., samples $\mathbf{z} \in \mathcal{Z}$ are i.i.d., then modified samples given the set $\mathcal{S}$, $\hat{\mathbf{z}}|\mathcal{S} \in \hat{\mathcal{Z}}$, are also i.i.d.*

Proof: We observe that:

$$\mathbf{x}_j\mathbf{d}_j^\top = \mathbf{x}_j \begin{bmatrix} \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_1 \geq 0) & \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_2 \geq 0) & \ldots & \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_P \geq 0) \end{bmatrix} \quad \forall j \in [\![N]\!].$$

Using Lemma 1, we obtain:

$$\begin{aligned} g_{\boldsymbol{\beta}}(\hat{\mathbf{z}}_j) &= \left|\boldsymbol{\beta}^\top \left(\mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top), \mathrm{vec}(\mathbf{x}_j\mathbf{d}_j^\top), y_j\right)\right| && \forall j \in [\![N]\!] \\ &= \left|\boldsymbol{\beta}^\top \left(f(\mathbf{x}_j, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_P), y_j\right)\right| && \forall j \in [\![N]\!] \\ &= \left|\boldsymbol{\beta}^\top \left(\hat{\mathbf{x}}_j, y_j\right)\right| && \forall j \in [\![N]\!], \end{aligned}$$

where $\hat{\mathbf{x}}_j = f(\mathbf{x}_j, \mathcal{S})$ is a basis expansion of the original feature space using the definition of the hyperplane generation (or approximation) and $g_{\boldsymbol{\beta}}(\hat{\mathbf{z}}) : \hat{\mathcal{Z}} \times \mathbb{R}^{2Pd+1} \to \mathbb{R}$ is the modified objective function when applying $\boldsymbol{\beta}$ on a modified sample. We have showed that each $\hat{\mathbf{x}}_j$ depends solely on $\mathbf{x}_j$ and $\mathcal{S}$. Because $\mathcal{S}$ is obtained prior to training, either by random sampling or by a deterministic hyperplane arrangement which depends on the whole training set $\mathbf{X}$, and the mapping f is the same on every sample, $\hat{\mathbf{z}}|\mathcal{S}$ is also i.i.d. □

Following Lemmas 1 and 2, we consider the new empirical distribution of the modified samples conditioned on $\mathcal{S}$: $\hat{\mathbb{F}}_N^{\mathcal{S}} = \frac{1}{N}\sum_{j=1}^N \delta_{\hat{\mathbf{z}}_j|\mathcal{S}}$, where $\hat{\mathbf{z}}_j = (\hat{\mathbf{x}}_j, y_j)^\top$. As the number of modified samples grows, $\hat{\mathbb{F}}_N^{\mathcal{S}}$ converges to the true underlying distribution $\mathbb{F}^{\mathcal{S}}$.

To make our case for the similarity between samples of $\hat{\mathbb{P}}_N$ and $\hat{\mathbb{F}}_N^{\mathcal{S}}$, we provide a detailed example of the mapping between the samples of the two distributions.

Example 2. *Let* $\mathbf{z}_j = (\mathbf{x}_j, y_j)$ *be any sample from* $\hat{\mathbb{P}}_N$ *and* $\hat{\mathbf{z}}_j = (\hat{\mathbf{x}}_j, y_j)$ *be the equivalent sample from* $\hat{\mathbb{F}}_N^{\mathcal{S}}$. *Suppose that* $\mathrm{card}(\mathcal{S}) = 3$ *and that* $\mathbf{d}_j^\top = \begin{bmatrix} \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_1 \geq 0) & \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_2 \geq 0) & \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_3 \geq 0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$, *then:*

$$\mathrm{vec}(\mathbf{x}_j \mathbf{d}_j^\top) = \mathrm{vec}\left(\mathbf{x}_j \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}\right) = \mathrm{vec}\left(\begin{bmatrix} \mathbf{x}_j & \mathbf{0}_d & \mathbf{x}_j \end{bmatrix}\right) = \mathrm{vec}\left(\begin{bmatrix} x_{j1} & 0 & x_{j1} \\ x_{j2} & 0 & x_{j2} \\ \vdots & \vdots & \vdots \\ x_{jd} & 0 & x_{jd} \end{bmatrix}\right)$$

$$= (\mathbf{x}_j,\ \mathbf{0}_d,\ \mathbf{x}_j),$$

and accordingly:

$$\hat{\mathbf{z}}_j = (\hat{\mathbf{x}}_j, y_j) = \left(\mathrm{vec}(\mathbf{x}_j \mathbf{d}_j^\top), \mathrm{vec}(\mathbf{x}_j \mathbf{d}_j^\top), y_j\right) = ((\mathbf{x}_j,\ \mathbf{0}_d,\ \mathbf{x}_j), (\mathbf{x}_j,\ \mathbf{0}_d,\ \mathbf{x}_j), y_j)\,.$$

Now, recall that $\mathcal{S}$ *is known a priori and can be generated randomly to approximate the hyperplane arrangement of* $\hat{\mathbb{P}}_N$. *As such,* $\mathcal{S}$ *do not bring additional information to* $\mathbf{x}_j$*; indeed* $\hat{\mathbf{x}}_j$ *is purely an expansion of* $\mathbf{x}_j$ *with a possible* ReLU *activation pattern. Effectively, we map the original samples to a higher dimension and apply the Wasserstein distributionally robust regularization on this higher representation which is closely related to the original feature space.*

We formulate the non-tractable Wasserstein distributionally robust training problem of the SCNN by centering the Wasserstein ball on $\hat{\mathbb{F}}_N^{\mathcal{S}}$:

$$\begin{aligned} \inf_{\boldsymbol{\beta}} \sup_{\mathbb{Q} \in \Omega} \ & \mathbb{E}^{\mathbb{Q}}[g_{\boldsymbol{\beta}}(\hat{\mathbf{z}}) | \mathcal{S}] && \texttt{(WDR-SCNNT)} \\ \text{s.t.} \quad & \boldsymbol{\nu}_i, \boldsymbol{\omega}_i \in \mathcal{K}_i \quad \forall i \in [\![P]\!] \\ & \boldsymbol{\beta} = (\mathrm{vec}(\boldsymbol{\nu}), \mathrm{vec}(-\boldsymbol{\omega}), -1) \\ & \Omega = \{\mathbb{Q} \in \mathcal{P}(\hat{\mathcal{Z}}) : W_{\|\cdot\|,t}(\mathbb{Q}, \hat{\mathbb{F}}_N^{\mathcal{S}}) \leq \epsilon\}, \end{aligned}$$

where $\hat{\mathbf{z}} \in \hat{\mathcal{Z}}$ and $\mathcal{P}(\hat{\mathcal{Z}})$ is the space of all possible distributions with support on $\hat{\mathcal{Z}} = \mathbb{R}^{2Pd+1}$. Note that $g_{\boldsymbol{\beta}}(\hat{\mathbf{z}}) = \left|\boldsymbol{\beta}^\top \left(\mathrm{vec}(\mathbf{x}_j \mathbf{d}_j^\top), \mathrm{vec}(\mathbf{x}_j \mathbf{d}_j^\top), y_j\right)\right|$ as defined in Lemma 2.

Proposition 1. *Consider the* ℓ_1*-norm training loss function and suppose Assumption 1 holds, then the convex reformulation of (*`WDR-SCNNT`*) is given by the following:*

(i) If the ℓ_1-norm is used in the definition of the order-1 Wasserstein distance $\left(W_{\|\cdot\|_1,1}\right)$, we have:

$$\begin{array}{lll} \min\limits_{\boldsymbol{\nu},\boldsymbol{\omega},a,\mathbf{c}} & \epsilon a + \dfrac{1}{N}\displaystyle\sum_{j=1}^{N} c_j & \qquad \texttt{(WDR1-SCNNT)} \\ \text{s.t.} & \boldsymbol{\beta}_k \leq a & \forall k \in [\![2Pd+1]\!] \\ & -\boldsymbol{\beta}_k \leq a & \forall k \in [\![2Pd+1]\!] \\ & \boldsymbol{\beta}^\top \hat{\mathbf{z}}_j \leq c_j & \forall j \in [\![N]\!] \\ & -\boldsymbol{\beta}^\top \hat{\mathbf{z}}_j \leq c_j & \forall j \in [\![N]\!] \\ & \boldsymbol{\beta} = (\operatorname{vec}(\boldsymbol{\nu}), \operatorname{vec}(-\boldsymbol{\omega}), -1) & \\ & \boldsymbol{\nu}_i, \boldsymbol{\omega}_i \in \mathcal{K}_i & \forall i \in [\![P]\!]. \end{array}$$

(ii) If the ℓ_2-norm is used in the definition of the order-1 Wasserstein distance $\left(W_{\|\cdot\|_2,1}\right)$, we obtain:

$$\begin{array}{lll} \min\limits_{\boldsymbol{\nu},\boldsymbol{\omega},a,\mathbf{c}} & \epsilon a + \dfrac{1}{N}\displaystyle\sum_{j=1}^{N} c_j & \qquad \texttt{(WDR2-SCNNT)} \\ \text{s.t.} & \|\boldsymbol{\beta}\|_2^2 \leq a^2 & \\ & a \geq 0 & \\ & \boldsymbol{\beta}^\top \hat{\mathbf{z}}_j \leq c_j & \forall j \in [\![N]\!] \\ & -\boldsymbol{\beta}^\top \hat{\mathbf{z}}_j \leq c_j & \forall j \in [\![N]\!] \\ & \boldsymbol{\beta} = (\operatorname{vec}(\boldsymbol{\nu}), \operatorname{vec}(-\boldsymbol{\omega}), -1) & \\ & \boldsymbol{\nu}_i, \boldsymbol{\omega}_i \in \mathcal{K}_i & \forall i \in [\![P]\!]. \end{array}$$

Proof: Lemma 1 states that the training problem, with respect to the modified samples $\hat{\mathbf{z}}$, is similar to a constrained linear regression over a basis expansion of the original feature space. This problem is convex in its objective and its constraints. Lemma 2 shows that the distribution of the modified samples is when conditioned on $\mathcal{S}$. Because (WDR-SCNNT) centers the Wasserstein ball on $\hat{\mathbb{F}}_N^{\mathcal{S}}$, by invoking Lemmas 1 and 2 regarding the modified sample space, we can reexpress (WDR-SCNNT) to the tractable DRO formulation (TWDRO). The final forms then follow from Theorem 6.3 of Mohajerin Esfahani and Kuhn (2018) and are inherited from WaDiRo linear regressions (Chen and Paschalidis 2020, Chapter 4). The detailed proof, adapted to shallow convex neural networks, is provided in Appendix A.1 for completeness. □

We remark that (WDR1-SCNNT) and (WDR2-SCNNT) can be easily modified to take into account the bias weights of the hidden and output layers of the SNN. The process is as follows: (i) extend $\mathbf{X}$ with a column of ones which is equivalent to adding bias weights in the hidden layer; (ii) let $\boldsymbol{\beta}$

be defined as $(\text{vec}(\boldsymbol{\nu}), \text{vec}(-\boldsymbol{\omega}), b, -1)$ and $\hat{\mathbf{z}}_j$ as $\left(\text{vec}(\mathbf{x}_j\mathbf{d}_j^\top), \text{vec}(\mathbf{x}_j\mathbf{d}_j^\top), 1, y_j\right)\ \forall j \in [\![N]\!]$ where $b \in \mathbb{R}$ is the bias of the sole output neuron. We also remark that both formulations of Proposition 1 are fairly fast to solve with open-source convex solvers, e.g., Clarabel (Goulart and Chen 2024), if the maximal number of neurons dictated by $\tilde{\mathcal{D}}_X$ and the size of the training set are *reasonable* in regards to the available computational power, e.g., in our case less than 300 neurons and 3000 samples were required. We provide time comparisons in Section A.2.3 and identify acceleration methods from the literature that could be adapted to our formulation in Section 3.9. Moreover, with only two hyperparameters, i.e., the maximal number of neurons and the Wasserstein ball radius, the hyperparameter search space is significantly reduced compared to standard non-convex training programs. As such, they contribute to a lighter MLOps pipeline by diminishing the computational effort required during hyperparameter tuning. Recall that the weights from the standard non-convex SNN training problem can be retrieved via (3.2) and (3.3) even though they are not required to obtain accurate predictions as the SCNN representation is sufficient.

3.4 Out-of-sample performance

Next, we obtain formal out-of-sample performance guarantees for WaDiRo-SCNNs. By first computing the Rademacher complexity (Bartlett and Mendelson 2002), it is then possible to bound the expected out-of-sample error of our model. We make the following additional assumptions similarly to Chen and Paschalidis (2020).

Assumption 2. *The norm of modified samples from $\mathbb{F}^{\mathcal{S}}$ are bounded above: $\|\hat{\mathbf{z}}\| \leq \hat{R} < +\infty$.*

Assumption 3. *The dual norm of the weights is bounded above: $\sup_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_* = B_* < +\infty$.*

These assumptions are reasonable as we aim for real-world applications which, by nature, generally satisfy them.

As shown previously, the SCNN training problem is similar to a linear regression. We can follow the procedure used in Chen and Paschalidis (2020) for linear regressions to bound the performance of WaDiRo-SCNNs.

Proposition 2. *Under Assumptions 1, 2, and 3, and considering an ℓ_1-loss function, with probability at least $1-\delta$ with respect to the sampling, the expected out-of-sample error of WaDiRo-SCNNs on $\mathbb{F}^{\mathcal{S}}$ is:*

$$\mathbb{E}^{\mathbb{F}^{\mathcal{S}}}\left[\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}\right| \middle| \mathcal{S}\right] \leq \frac{1}{N}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}_j\right| + \frac{2B_*\hat{R}}{\sqrt{N}} + B_*\hat{R}\sqrt{\frac{8\ln(2/\delta)}{N}},$$

for a modified sample $\hat{\mathbf{z}}$ from the true distribution, that was not available for training, where $\boldsymbol{\beta}^\star$ are trained weights obtained from either (`WDR1-SCNNT`) or (`WDR2-SCNNT`).

Moreover, for any $\zeta > \frac{2B_*\hat{R}}{\sqrt{N}} + B_*\hat{R}\sqrt{\frac{8\ln(2/\delta)}{N}}$, we have:

$$\Pr\left(\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}\right| \geq \frac{1}{N}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}_j\right| + \zeta \middle| \mathcal{S}\right) \leq \frac{\frac{1}{N}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}_j\right| + \frac{2B_*\hat{R}}{\sqrt{N}} + B_*\hat{R}\sqrt{\frac{8\ln(2/\delta)}{N}}}{\frac{1}{N}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}_j\right| + \zeta}.$$

Proof: By Lemma 1, the WaDiRo-SCNN's training loss function with respect to $\hat{\mathbb{F}}_N^{\mathcal{S}}$ belongs to the following family:

$$\mathcal{H} = \left\{(\mathbf{x}, y) \to h_{\boldsymbol{\beta}}(\mathbf{x}, y) : h_{\boldsymbol{\beta}}(\mathbf{x}, y) = \left|y - \mathbf{x}^\top\boldsymbol{\beta}\right|\right\}.$$

Thus, the bounds follow from Theorem 4.3.3 of Chen and Paschalidis (2020). □

Interestingly, we can obtain similar performance bounds with respect to the original distribution $\mathbb{P}$ instead of the modified one. We now make further assumptions.

Assumption 4. *The norm of the original samples, features, and labels are bounded above: $\|\mathbf{z}\| \leq R < +\infty$, $\|\mathbf{x}\| \leq S < +\infty$, and $\|\mathbf{y}\| \leq T < +\infty$.*

Consider the *general* family of loss functions for arbitrary weights $\boldsymbol{\beta}$:

$$\mathcal{G} = \left\{(\mathbf{x}, y) \mapsto g_{\boldsymbol{\beta}}(\mathbf{x}, y) : g_{\boldsymbol{\beta}}(\mathbf{x}, y) = \left|y - \boldsymbol{\beta}^\top\hat{\mathbf{x}}\right|,\ \hat{\mathbf{x}} = \left(\mathrm{vec}(\mathbf{x}\mathbf{d}^\top)^\top, \mathrm{vec}(\mathbf{x}\mathbf{d}^\top)^\top\right)\right\}, \tag{3.4}$$

where $\mathbf{d}^\top = \begin{bmatrix}\mathbb{1}(\mathbf{x}^\top\mathbf{s}_1 \geq 0) & \mathbb{1}(\mathbf{x}^\top\mathbf{s}_2 \geq 0) & \dots & \mathbb{1}(\mathbf{x}^\top\mathbf{s}_P \geq 0)\end{bmatrix}$ and $\mathbf{s}_i \in \mathcal{S}\ \forall i \in [\![P]\!]$ are *general* sampling vectors.

Consider $\psi_1 = B_*2PR$ for (`WDR1-SCNNT`) and $\psi_2 = B_*\sqrt{2P}R$ for (`WDR2-SCNNT`), upper bounds on the prediction error of the WaDiRo-SCNN for any sample from the sample space. Then, we can rewrite the out-of-sample performance of trained WaDiRo-SCNNs as follows.

Proposition 3. *Under Assumptions 1, 3, and 4, and considering an ℓ_1-loss function, with probability at least $1-\delta$ with respect to the sampling, the expected out-of-sample error of WaDiRo-SCNNs on $\mathbb{P}^{\mathcal{S}}$ is:*

$$\mathbb{E}^{\mathbb{P}^{\mathcal{S}}}\left[\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}\right| \middle| \mathcal{S}\right] \leq \frac{1}{N}\sum_{j=1}^{N}\left|\boldsymbol{\beta}^{\star\top}\hat{\mathbf{z}}_j\right| + \frac{2\psi_l}{\sqrt{N}} + B_*\hat{R}\sqrt{\frac{8\ln(2/\delta)}{N}}, \tag{3.5}$$

for a modified sample $\hat{\mathbf{z}}$ from the true distribution given $\mathcal{S}$, that was not available for training, where $\boldsymbol{\beta}^\star$ are the optimal weights obtained from either (`WDR1-SCNNT`) or (`WDR2-SCNNT`), $\mathbb{P}^{\mathcal{S}}$ is the conditioned underlying distribution, and where $l \in \{1, 2\}$.

Proof: We first bound the value of the loss function for a single arbitrary sample. For (WDR2-SCNNT) the dual norm of the weights is given by $\|\cdot\|_2$, thus:

$$
\begin{aligned}
|\boldsymbol{\beta}^\top \hat{\mathbf{z}}| &\leq \|\boldsymbol{\beta}\|_2 \|\hat{\mathbf{z}}\|_2 && \text{(Hölder's inequality)} \\
&\leq B_* \sqrt{\sum_{i=1}^{P} \sum_{k=1}^{d} 2(x_k \mathbb{1}(\mathbf{x}^\top \mathbf{s}_i \geq 0))^2 + y^2} \\
&\leq B_* \sqrt{\sum_{i=1}^{P} \sum_{k=1}^{d} 2(x_k)^2 + y^2} && \text{(Full activation)} \\
&= B_* \sqrt{2P\|\mathbf{x}\|_2^2 + y^2} \\
&\leq B_* \sqrt{2PS^2 + T^2} \\
&\leq B_* \sqrt{2P} R \\
&= \psi_2. && \text{(Definition)}
\end{aligned}
$$

For (WDR1-SCNNT), the dual norm is the $\|\cdot\|_\infty$-norm. Hölder's inequality similarly yields:

$$
\begin{aligned}
|\boldsymbol{\beta}^\top \hat{\mathbf{z}}| &\leq \|\boldsymbol{\beta}\|_\infty \|\hat{\mathbf{z}}\|_1 && \text{(Hölder's inequality)} \\
&\leq B_* \sum_{i=1}^{P} \sum_{k=1}^{d} 2|(x_k \mathbb{1}(\mathbf{x}^\top \mathbf{s}_i \geq 0))| + |y| \\
&\leq B_*(2P\|\mathbf{x}\|_1 + |y|) \\
&\leq B_*(2PS + T) \\
&\leq B_* 2PR \\
&= \psi_1. && \text{(Definition)}
\end{aligned}
$$

We now bound the Rademacher complexity $\mathcal{R}_N$ over N samples for the family of functions $\mathcal{G}$ using

the definition provided by Bartlett and Mendelson (2002):

$$\begin{aligned}
\mathcal{R}_N(\mathcal{G}) &= \mathbb{E}\left[\mathbb{E}\left[\sup_{g\in\mathcal{G}}\frac{2}{N}\left|\sum_{j=1}^{N}\sigma_j g_\beta(\mathbf{x}_j, y_j)\right| \left|(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\right.\right]\right] \\
&\leq \mathbb{E}\left[\frac{2}{N}\left|\sum_{j=1}^{N}\sigma_j\psi_l\right|\right] && \text{(Upper bound)} \\
&= \frac{2\psi_l}{N}\mathbb{E}\left[\left|\sum_{j=1}^{N}\sigma_j\right|\right] \\
&\leq \frac{2\psi_l}{N}\mathbb{E}\left[\sqrt{\sum_{j=1}^{N}\sigma_j^2}\right] && (\sqrt{N}\geq 0) \\
&= \frac{2\psi_l}{\sqrt{N}},
\end{aligned}$$

where $\sigma_j\ j \in [\![N]\!]$ are independent random variables sampled from the set $\{-1, 1\}$, and $l \in \{1, 2\}$ accordingly to the training procedure.

Thus, by adapting Theorem 8 of Bartlett and Mendelson (2002) to loss functions mapping between $[0, B_*\hat{R}]$ and using the fact that $\mathcal{R}_N(\mathcal{G})$ is greater than the Rademacher complexity term defined in Theorem 8, we obtain the expression from (3.5). □

We remark that the expected out-of-sample error reduces to the training error as the number of training samples grows. We also remark that the tighter the bounds are on the sample space, i.e., $\hat{R},\ R,\ S,$ and T, the tighter the guarantees are. Proof of Proposition 3 also leads to a probabilistic guarantee similar to the one in Proposition 2.

The theoretical guarantees provided in this section contain a probabilistic term and may be difficult to leverage empirically. Nonetheless, there is value in knowing that they exist when deploying complex ML models such as ours in critical applications because of the general lack of trust towards them. They also indicate how one can increase reliability by making these bounds tighter.

3.5 Physics-constrained shallow convex neural networks

Compared to traditional physics-informed neural networks (Cuomo et al. 2022), because the training problem is a convex program, knowledge of the system can be introduced as hard, convex constraints in the training phase. Let $\hat{y}_j = (\mathrm{vec}(\boldsymbol{\nu}), \mathrm{vec}(-\boldsymbol{\omega}))^\top \hat{\mathbf{x}}_j,\ j \in [\![N]\!]$ be the prediction of the SCNN for a specific modified input vector. We formulate the physics-constrained SCNN (PCSCNN) training

problem as:

$$\begin{aligned} \min_{\boldsymbol{\nu},\boldsymbol{\omega}} \sum_{j=1}^{N} \mathcal{L}\left(\hat{y}_j, y_j\right) & \qquad \text{(PCSCNNT)} \\ \text{s.t. } \hat{y}_j \in \mathcal{Y}_j & \quad \forall j \in [\![N]\!] \\ \boldsymbol{\nu}_i, \boldsymbol{\omega}_i \in \mathcal{K}_i & \quad \forall i \in [\![P]\!], \end{aligned}$$

where $\mathcal{Y}_j \subseteq \mathbb{R}$ is the convex intersection of all physical constraints for sample $j \in [\![N]\!]$. The constraints are guaranteed to be respected during training, which is generally not true in common physics-informed NNs. We illustrate some convex constraints with examples:

$$l_j \leq \hat{y}_j \leq u_j \qquad \forall j \in [\![N]\!] \tag{3.6}$$

$$|\hat{y}_j - \hat{y}_{j-1}| \leq r_j \qquad \forall j \in [\![N-1]\!] \tag{3.7}$$

$$\underline{\xi}_j \leq \sum_{i=1}^{j} \boldsymbol{\eta}_i^\top \mathbf{x}_i + \sum_{i=1}^{j} \gamma_i \hat{y}_i \leq \bar{\xi}_j \quad \forall j \in [\![N]\!], \tag{3.8}$$

where in (3.6), (3.7), and (3.8), with respect to sample j and $\forall i \in [\![j]\!]$, $u_j \in \mathbb{R}$ is an upper bound, $l_j \in \mathbb{R}$ is a lower bound, and $r_j \in \mathbb{R}$ is a ramping bound. Additionally, (3.8) is a bounded first-order relation between input and output history with $\underline{\xi}_j \in \mathbb{R}, \bar{\xi}_j \in \mathbb{R}, \boldsymbol{\eta}_i \in \mathbb{R}^d, \gamma_i \in \mathbb{R}$, e.g., a discrete-time dynamical equation or PDE. These examples are linear or easily linearizable, but we stress that any convex constraint can be included. In the case of non-differentiable convex functions, the numerical solver must be chosen accordingly, or a subgradient method be implemented Boyd and Park (2014). We remark that the WaDiRo-SCNN training problem can be similarly modified to consider physical constraints by adding $\hat{y}_j \in \mathcal{Y}_j \; \forall j \in [\![N]\!]$ to the constraints of (`WDR1-SCNNT`) and (`WDR2-SCNNT`).

Proposition 4. *Assume that mappings (3.2) and (3.3) are exact, i.e., $m \geq m^\star$, then (`PCSCNNT`) is equivalent to adding convex physical constraints to (`SNNT`) for any additive convex loss function $\mathcal{L}$, e.g., $\|\cdot\|_1$ and $\|\cdot\|_2^2$.*

Proof: To prove Proposition 4 we utilize the exact mappings of the original problem and show that the integration of physical constraints into (`SNNT`) yields to (`PCSCNNT`). We start by stating the physics-constrained non-convex problem:

$$\begin{aligned} p_{\mathcal{Y}}^\star = \min_{\mathbf{W}_1, \mathbf{w}_2} & \sum_{j=1}^{N} \mathcal{L}\left(\sum_{i=1}^{m} \max\left\{\mathbf{x}_j^\top \mathbf{W}_{1i}, 0\right\} w_{2i}, y_j\right) \\ \text{s.t.} & \sum_{i=1}^{m} \max\left\{\mathbf{x}_j^\top \mathbf{W}_{1i}, 0\right\} w_{2i} \in \mathcal{Y}_j \quad \forall j \in [\![N]\!]. \end{aligned}$$

Substituting mappings (3.2) and (3.3), and adding the cone constraints, we have:

$$
\begin{aligned}
p^\star_{\mathcal{Y}} = \min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & \sum_{j=1}^{N} \mathcal{L}\left(\sum_{i=1}^{m^\star/2} \max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\nu_i}}{\sqrt{\|\boldsymbol{\nu_i}\|_2}}, 0 \right\} \sqrt{\|\boldsymbol{\nu_i}\|_2} + \max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\omega_i}}{\sqrt{\|\boldsymbol{\omega_i}\|_2}}, 0 \right\} (-\sqrt{\|\boldsymbol{\omega_i}\|_2}), y_j \right) \\
\text{s.t.} \quad & \sum_{i=1}^{m^\star/2} \max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\nu_i}}{\sqrt{\|\boldsymbol{\nu_i}\|_2}}, 0 \right\} \sqrt{\|\boldsymbol{\nu_i}\|_2} + \\
& \qquad \max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\omega_i}}{\sqrt{\|\boldsymbol{\omega_i}\|_2}}, 0 \right\} (-\sqrt{\|\boldsymbol{\omega_i}\|_2}) \in \mathcal{Y}_j \quad \forall j \in [\![N]\!] \\
& (2\mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_i \geq 0) - 1)\mathbf{x}_j^\top \boldsymbol{\nu_i} \geq 0 \quad \forall j \in [\![N]\!], \forall i \in \left[\!\left[\frac{m^\star}{2} \right]\!\right] \\
& (2\mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_i \geq 0) - 1)\mathbf{x}_j^\top \boldsymbol{\omega_i} \geq 0 \quad \forall j \in [\![N]\!], \forall i \in \left[\!\left[\frac{m^\star}{2} \right]\!\right].
\end{aligned}
$$

Using the hyperplane arrangement from which we obtained the possible activation patterns, we can show that if there is an activation for a specific ij, then $\max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\nu_i}}{\sqrt{\|\boldsymbol{\nu_i}\|_2}}, 0 \right\} \sqrt{\|\boldsymbol{\nu_i}\|_2} = \mathbf{x}_j^\top \boldsymbol{\nu_i}$, and $\max\left\{ \mathbf{x}_j^\top \frac{\boldsymbol{\omega_i}}{\sqrt{\|\boldsymbol{\omega_i}\|_2}}, 0 \right\} (-\sqrt{\|\boldsymbol{\omega_i}\|_2}) = -\mathbf{x}_j^\top \boldsymbol{\omega_i}$, and otherwise both equals 0. Thus, we can completely substitute the ReLU operator by $d_{ji} = \mathbb{1}(\mathbf{x}_j^\top \mathbf{s}_i \geq 0)$, leading to:

$$
\begin{aligned}
p^\star_{\mathcal{Y}} = \min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & \sum_{j=1}^{N} \mathcal{L}\left(\sum_{i=1}^{m^\star/2} d_{ji} \mathbf{x}_j^\top (\boldsymbol{\nu}_i - \boldsymbol{\omega}_i), y_j \right) \\
\text{s.t.} \quad & \sum_{i=1}^{m^\star/2} d_{ji} \mathbf{x}_j^\top (\boldsymbol{\nu}_i - \boldsymbol{\omega}_i) \in \mathcal{Y}_j \; \forall j \in [\![N]\!] \\
& (2d_{ji} - 1)\mathbf{x}_j^\top \boldsymbol{\nu_i} \geq 0 \quad \forall j \in [\![N]\!], \forall i \in \left[\!\left[\frac{m^\star}{2} \right]\!\right] \\
& (2d_{ji} - 1)\mathbf{x}_j^\top \boldsymbol{\omega_i} \geq 0 \quad \forall j \in [\![N]\!], \forall i \in \left[\!\left[\frac{m^\star}{2} \right]\!\right],
\end{aligned}
$$

from which we find (PCSCNNT). □

We remark that a similar approach can be used for the regularized problem and that the converse of the proposition can be obtained following the same idea.

3.6 Post-training verification

As mentioned previously, there is an appeal in having ML model architectures that are easily integrable in post-training verification frameworks. We now showcase how to integrate SCNNs in such

frameworks by proposing a local-Lipschitz stability certification program evaluating the stability induced by different SCNN training programs, viz., (`WDR1-SCNNT`), (`WDR2-SCNNT`), and (`SCNNT`). We certify the worst-case output deviation of SCNNs when subject to a small perturbation $\varepsilon \in \mathcal{E}$ in their feature space. This methodology is inspired by Huang et al. (2021) on ReLU feedforward deep neural networks, but tailored to SCNNs thanks to the decoupled formulation. Recall the definition of the Lipschitz constant for a function $f : \mathcal{X} \to \mathbb{R}$. A function is K-Lipschitz if there exists a constant $0 < K < +\infty$ such that :

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq K\|\mathbf{x}_1 - \mathbf{x}_2\|,$$

for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. In our case, we consider only neighbouring points by defining $\mathbf{x}_2 = \{\mathbf{x}_1 + \boldsymbol{\varepsilon} | \boldsymbol{\varepsilon} \in \mathcal{E}, \mathbf{x}_1 \in \mathcal{X}, \mathcal{E} \text{ is a bounded convex set}\}$, thus the term local-Lipschitz. We define the ε-stability of a function f by L_ε such that:

$$L_\varepsilon = \max_{\mathbf{x}_1, \boldsymbol{\varepsilon}: \mathbf{x}_1 \in \mathcal{X}, \boldsymbol{\varepsilon} \in \mathcal{E}} |f(\mathbf{x}_1) - f(\mathbf{x}_1 + \boldsymbol{\varepsilon})|,$$

which is visibly linked to the local-Lipschitz bound.

Proposition 5. *Let $\boldsymbol{\beta}^\star$ be the optimal weights obtained by an SCNN training program and a set of sampling vectors $\mathcal{S}$ obtained before training. Assuming an SCNN makes a finite mapping from a bounded feature space to the label space, the worst possible output deviation $0 \leq L_\varepsilon < +\infty$ of the SCNN for any perturbation vector $\boldsymbol{\varepsilon}$, from a bounded convex set $\mathcal{E}$, on any input from a bounded feature space $\mathcal{X}$ can be obtained by a mixed-integer convex program (MICP).*

Proof: Let $M \in \mathbb{R}_{>0}$ a large strictly positive scalar, $\mathbf{x} \in \mathcal{X}$ any possible input from feature space $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathbf{x}_\varepsilon \in \mathcal{X}$ a perturbed input close to $\mathbf{x}$. The post-training verification problem can be

formulated as:

$$
\begin{aligned}
L_\varepsilon = \max_{\hat{y},\hat{y}_\varepsilon,\mathbf{x},\boldsymbol{\varepsilon},\mathbf{A},\mathbf{B},\boldsymbol{\gamma},\boldsymbol{\eta}} & |\hat{y} - \hat{y}_\varepsilon| && && (3.9)\\
\text{s.t.} \quad \hat{y} & = \boldsymbol{\beta}^{\star\top}\hat{\mathbf{x}} && && (3.10)\\
\hat{\mathbf{x}} & = (\mathrm{vec}(\mathbf{A}), \mathrm{vec}(\mathbf{A})) && && (3.11)\\
\mathbf{x}^\top \mathbf{s}_i & \leq M\gamma_i && \forall i \in [\![P]\!] && (3.12)\\
\mathbf{x}^\top \mathbf{s}_i & \geq -M(1-\gamma_i) && \forall i \in [\![P]\!] && (3.13)\\
x_k & \leq a_{ki} + M(1-\gamma_i) && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.14)\\
x_k & \geq a_{ki} - M(1-\gamma_i) && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.15)\\
a_{ki} & \leq M\gamma_i && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.16)\\
a_{ki} & \geq -M\gamma_i && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.17)\\
\boldsymbol{\gamma} & \in \{0,1\}^P && && (3.18)\\
\hat{y}_\varepsilon & = \boldsymbol{\beta}^{\star\top}\hat{\mathbf{x}}_\varepsilon && && (3.19)\\
\hat{\mathbf{x}}_\varepsilon & = (\mathrm{vec}(\mathbf{B}), \mathrm{vec}(\mathbf{B})) && && (3.20)\\
\mathbf{x}_\varepsilon & = (\mathbf{x} + \boldsymbol{\varepsilon}) && && (3.21)\\
\mathbf{x}_\varepsilon^\top \mathbf{s}_i & \leq M\eta && \forall i \in [\![P]\!] && (3.22)\\
\mathbf{x}_\varepsilon^\top \mathbf{s}_i & \geq -M(1-\eta_i) && \forall i \in [\![P]\!] && (3.23)\\
x_{\varepsilon,k} & \leq b_{ki} + M(1-\eta_i) && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.24)\\
x_{\varepsilon,k} & \geq b_{ki} - M(1-\eta_i) && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.25)\\
b_{ki} & \leq M\eta_i && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.26)\\
b_{ki} & \geq -M\eta_i && \forall i \in [\![P]\!],\ \forall k \in [\![d]\!] && (3.27)\\
\boldsymbol{\eta} & \in \{0,1\}^P && && (3.28)\\
\mathbf{s}_i & \in \mathcal{S} && \forall i \in [\![P]\!] && (3.29)\\
\boldsymbol{\varepsilon} & \in \mathcal{E} && && (3.30)\\
\mathbf{x} & \in \Xi, && && (3.31)\\
\mathbf{x}, \mathbf{x}_\varepsilon & \in \mathcal{X} && && (3.32)
\end{aligned}
$$

where $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_P)$ and $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_P)$ are intermediate variables. $\mathcal{E}, \Xi \subset \mathbb{R}^d$ are convex sets that ensure that $\boldsymbol{\varepsilon}$ is finite and *small* and that $\mathbf{x}$ is finite, respectively, e.g., box constraints on each element or bounded Euclidian norms. The size of $\mathcal{E}$ should be chosen with respect to the application. Constraints (3.10)$-$(3.18) and (3.20)$-$(3.28) are used to represent the ReLU-like activation patterns of the SCNN for the unperturbed and perturbed

output, respectively. Specifically, matrices $\mathbf{A}$ and $\mathbf{B}$ model $\mathbf{x}\mathbf{d}^\top$ and $\mathbf{x}_\varepsilon{\mathbf{d}_\varepsilon}^\top$ with constraints (3.14)−(3.17) and (3.24)−(3.27), respectively. Constraints (3.12), (3.13), (3.22), and (3.23) embed the following definitions: $\mathbf{d}^\top = \begin{bmatrix}\mathbb{1}(\mathbf{x}^\top\mathbf{s}_1 \geq 0) & \mathbb{1}(\mathbf{x}^\top\mathbf{s}_2 \geq 0) & \dots & \mathbb{1}(\mathbf{x}^\top\mathbf{s}_P \geq 0)\end{bmatrix}$ and $\mathbf{d}_\varepsilon^\top = \begin{bmatrix}\mathbb{1}(\mathbf{x}_\varepsilon^\top\mathbf{s}_1 \geq 0) & \mathbb{1}(\mathbf{x}_\varepsilon^\top\mathbf{s}_2 \geq 0) & \dots & \mathbb{1}(\mathbf{x}_\varepsilon^\top\mathbf{s}_P \geq 0)\end{bmatrix}$. Finally, the non-concave objective function can be linearized using another binary variable, ζ:

$$\max_{\hat{y},\hat{y}_\varepsilon,\mathbf{x},\boldsymbol{\varepsilon},\mathbf{A},\mathbf{B},\boldsymbol{\gamma},\boldsymbol{\eta}} \quad \alpha \tag{3.33}$$

$$\text{s.t.} \quad \hat{y}_\varepsilon - \hat{y} \leq M\zeta \tag{3.34}$$

$$\hat{y}_\varepsilon - \hat{y} \geq M(\zeta - 1) \tag{3.35}$$

$$\alpha \leq \hat{y}_\varepsilon - \hat{y} + M(1 - \zeta) \tag{3.36}$$

$$\alpha \geq \hat{y}_\varepsilon - \hat{y} - M(1 - \zeta) \tag{3.37}$$

$$\alpha \leq \hat{y} - \hat{y}_\varepsilon + M\zeta \tag{3.38}$$

$$\alpha \geq \hat{y} - \hat{y}_\varepsilon - M\zeta \tag{3.39}$$

$$\zeta \in \{0, 1\} \tag{3.40}$$

$$(3.10) - (3.32),$$

which concludes the proof. □

Besides pure certification, this formulation is interesting because it could be used in a pessimistic bilevel program, see Dempe (2002), that integrates SCNN training and certification into the same problem. We leave this opening for future research directions. We remark that M can be tuned separately for each constraint to be as tight as possible and that recent works have greatly accelerated similar procedures for deep ReLU neural networks, e.g., see Zhou et al. (2024).

3.7 Outlier generation on benchmark functions

In this chapter, to generate empirical distributions with complex patterns, we propose to use the mathematical expressions of traditional non-convex benchmark functions as a starting point. These functions are normally used to evaluate optimization solvers because of their interesting patterns and their generalizability to n dimensions.

We use a methodology inspired by Pallage et al. (2024) to generate outliers. This idea is presented later in Section 4. They propose a method based on the sliced-Wasserstein (SW) distance to remove out-of-distribution outlier points from training sets. In this chapter, we do the opposite and use a SW-based method to ensure that introduced outliers are out-of-distribution with respect to the original data distribution.

Computing the Wasserstein distance is generally of high computational complexity, but the one-dimensional Wasserstein distance is an exception as it possesses a closed-form solution. The SW distance exploits this special case by first computing, through linear projections, an *infinite* amount of one-dimensional representations for each original n-dimensional distribution. It then computes the average Wasserstein distance between their one-dimensional projections (Kolouri et al. 2019). In practice, the order-t SW distance's approximation under a Monte Carlo scheme of L samples is defined as:

$$SW_{\|\cdot\|,t}\left(\mathbb{U},\mathbb{V}\right) \approx \left(\frac{1}{L}\sum_{l=1}^{L} W_{\|\cdot\|,t}\left(\mathfrak{R}\mathbb{U}\left(\cdot,\theta_l\right),\mathfrak{R}\mathbb{V}\left(\cdot,\theta_l\right)\right)^t\right)^{1/t},$$

where $\mathfrak{R}\mathbb{D}(\cdot,\boldsymbol{\theta}_l)$ is a single projection of the Radon transform of distribution function $\mathbb{D}$ over a fixed sample $\boldsymbol{\theta}_l \in \mathbb{S}^{d-1} = \{\boldsymbol{\theta} \in \mathbb{R}^d | \theta_1^2 + \theta_2^2 + \cdots + \theta_d^2 = 1\}\ \forall l \in [\![L]\!]$. Note that $\mathbb{S}^{d-1}$ is often referred to as the $(d-1)$-dimensional unit sphere as it encloses every d-dimensional unit-norm vector in a sphere of radius 1. It can be shown that the WD and SWD are closely related (Bonnotte 2013, Theorem 5.1.5). We refer interested readers to Bonneel et al. (2015) and Kolouri et al. (2019) as they offer deeper mathematical insights.

Our procedure is as follows. Consider an empirical distribution of N samples $\hat{\mathbb{P}}_N = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{z}_i}$ obtained by sampling uniformally a non-convex benchmark function denoted by $\mathbb{P}$. We use the notation $\hat{\mathbb{P}}_{N+m}$ to denote the empirical distribution with m added samples from $\mathbb{P}$ and $\hat{\mathbb{P}}_N \cup \mathcal{O}_m$ to represent the union of the empirical distribution with a set of m outlier samples $\mathcal{O}_m$. Let $\varphi_m > 0$ represent a threshold SW distance for sample m. The SW distance deals with distributions of equal weights and must account for present outliers when generating a new one. Assuming we want to generate a total of $\overline{m} \in \mathbb{Z}_{>0}$ out-of-sample outliers, the iterative procedure is based on:

$$\hat{\mathbb{P}}_N \cup \mathcal{O}_m = \left\{\hat{\mathbb{P}}_N \cup \mathcal{O}_{m-1} \cup \{\tilde{\mathbf{z}}\} \,\middle|\, \tilde{\mathbf{z}} \in \mathcal{Z}, \left(SW_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N+1} \cup \mathcal{O}_{m-1}, \hat{\mathbb{P}}_N \cup \mathcal{O}_{m-1} \cup \{\tilde{\mathbf{z}}\}) \geq \varphi_m\right)\right\} \quad m \in [\![\overline{m}]\!],$$

where $\mathcal{O}_0 = \emptyset$ is the empty set. This procedure ensures that each new outlier creates a shift of at least φ with respect to the empirical distribution and previous outliers $1,\ 2, \ldots, m-1$. We must consider previous outliers to ensure we do not create any new involuntary pattern in the data. Because we do not make a single assumption on $\mathbb{P}$, this methodology is adaptable to generate outliers on any function.

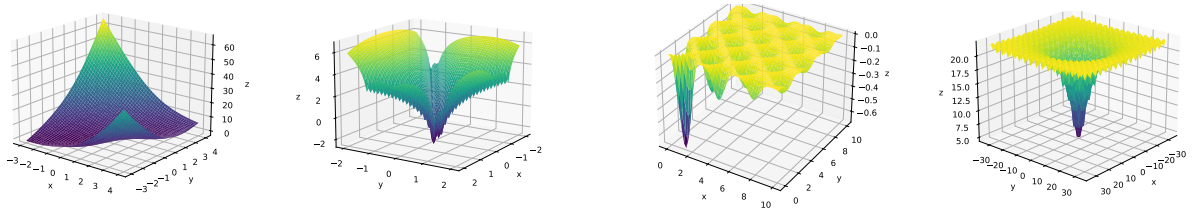
3.8 Numerical study

This section is split into three parts. We first benchmark our approach in a numerical study with data from a controlled environment before showcasing its performance in a real-world VPP application

and evaluating model stability on classical benchmark datasets.

3.8.1 Synthetic experiments

Setting. We aim to assess the performance of our model at learning nonlinear functions in heavily corrupted settings, i.e., at being robust against outliers, with a limited amount of training data. To avoid biasing the underlying distribution, we introduce outliers with the SW distance as presented in section 3.7 and a Gaussian noise, in both the training and validation datasets. We choose traditional non-convex benchmark functions: McCormick (Adorio 2005), Picheny-Goldstein-Price (PGP) (Picheny et al. 2013), Keane (Keane 1994), and Ackley (Ackley 2012). These functions strike a balance between complexity and tractability. Despite their non-convexity, they exhibit discernable trends and patterns that facilitate learning compared to some other benchmark functions, e.g., Rana or egg-holder (Whitley et al. 1996). The variation granularity within these functions is moderate ensuring that the learning process is not hindered by excessively fine and extreme fluctuations in the codomain. In a way, this is what we would expect of most real-world regression datasets, e.g., University of California Irvine's ML repository (Kelly et al. 2023), with the added advantage of being able to control the degree of corruption of the datasets. These functions were generated with the Python library `benchmark functions` (Baronti and Castellani 2024) and are illustrated in Figure 3.3.



(a) McCormick (b) PGP (c) Keane (d) Ackley

Figure 3.3 Benchmark functions

We compare our model to an SCNN, a regularized SCNN, a regularized linear regression, a WaDiRo linear regression, and a 4-layer deep ReLU feedforward neural network (FNN) with batch training and dropout layers. Each of these approaches uses an ℓ_1-loss function. WaDiRo-SCNNs and SCNNs are modelled and optimized with `cvxpy` (Diamond and Boyd 2016) and with Clarabel (Goulart and Chen 2024) as the solver. We then extract the weights and encode them into `pytorch`'s neural network module (Paszke et al. 2019). Similarly, the WaDiRo linear regression and the regularized linear regression are optimized with `cvxpy` and Clarabel, while the deep FNN is fully implemented with `pytorch`. We use simulated annealing (Ingber 1993) for hyperparameter optimization with

`hyperopt` (Bergstra et al. 2015). We run 400 trials per ML model, per benchmark function, except for the deep FNN which we limit to 100 trials to mitigate its substantial training time. We split the data such that the training data accounts for 60% of the whole dataset and we split the rest evenly between the validation and the testing phases. We note that the testing data is uncorrupted to measure the ability of each method to learn the true underlying functions without having access to it. We use the mean absolute error (MAE) for hyperparameter tuning. We also measure the root mean squared error (RMSE) for additional performance insights. Our whole experiment setup is available on our GitHub page[1] for reproducibility and is delineated in Appendix A.2. In Experiment A, the outliers are introduced solely in the training data while they are split randomly between training and validation for all the other experiments. Table 3.1 details the experiment contexts.

Numerical results. We now present and analyze the results of each synthetic experiment. Figure 3.4 shows the normalized error metrics of each model on each function. The normalization is done within each column so that 1.00 and 0.00 represent, respectively, the highest and lowest error between all models for a specific metric and function, e.g., the MAE on McCormick's function. We propose this representation because different functions have patterns of different complexity and it highlights how models relatively compare to each other when facing the same level of pattern complexity. Nonetheless, the absolute values are also presented in Appendix A.2 as a complement.

The objective of Experiment A is to evaluate the ability of each model at learning over a highly adversarial training set while having the possibility to refine, indirectly through the validation phase, on a slightly out-of-sample distribution. This experiment was designed to see how different models' regularization techniques may prevent them from overfitting on training outliers. As we can see, the WaDiRo-SCNN, regularized SCNN, and deep FNN perform similarly except when trying to learn Ackley where the WaDiRo-SCNN has the lowest test error. In this experiment, the linear models perform poorly compared to other models. We hypothesize that the validation phase is uncorrupted enough to let nonlinear models adjust to the non-convex functions during hyperparameter optimization. As expected the non-regularized SCNN leads to performance lower than its

Table 3.1 Degree of corruption of each experiment

Experiment	**Outlier percentage**			**Gaussian noise**		
	Train.	**Val.**	**Test**	**Train.**	**Val.**	**Test**
A	0.4	0	0	True	True	False
B	0.1		0	True	True	False
C	0.4		0	True	True	False

[1] https://github.com/jupall/WaDiRo-SCNN

regularization counterpart in this particular setting.

In Experiments B and C, outliers are randomly split between training and validation. This implies that models may overfit on outliers during the hyperparameter optimization. Interestingly, in a less corrupted setting, i.e., in Experiment B, we observe that the best models are the nonlinear ones. Yet, when increasing the corruption level, i.e., the ratio of outliers in the data, the linear models become competitive. In both experiments, the WaDiRo-SCNN's error is low while not necessarily being the lowest of all models. This is expected as the DRO framework is conservative by definition, maybe even more than other regularization methodologies. This conservatism is what makes it appealing for applications in critical sectors.

We remark that the deep FNN often excels but is subject to its highly stochastic training as there is no guarantee that a good model will be found within the 100 runs made for hyperparameter tuning. Other models, having low-stochasticity training, are easier to tune. We also remark that the non-regularized SCNN performs well but lacks the constancy of its regularized counterparts throughout the experiments.

3.8.2 Baseline estimation of non-residential buildings

Setting. Montréal's continental humid climate delivers harsh winters and hot summers. As such, a significant amount of energy is consumed for both electric heating and air conditioning. Moreover, date and time features have a strong influence on non-residential buildings' consumption patterns as their activities may change within days and weeks, e.g., opening hours, occupancy, and seasonal activities.

In this experiment, we seek to predict the year-round hourly energy consumption in kWh of 16 distinct non-residential buildings located in three adjacent distribution substations in Montréal, Québec, Canada. These buildings are enrolled in a non-residential DR program. Being able to predict the baseline consumption of these assets under different contexts allows VPPs to better characterize grid needs through time and to retroactively estimate the relief given by each building during DR events. As we do not have any information on building activity, this task is done using only hourly consumption history and historical meteorological features. We use a 12-week history to train each model to predict the following week. We have approximately 2.5 years of collected data on each building. We proceed with a 12-week sliding window and collect the predictions for each new week. We use the data of the first year and a half for hyperparameter tuning and the data of the last year for performance assessment. The features and label are listed in Table 3.2. We set cyclical features with a $\cos / \sin$ encoding similarly to (Moon et al. 2018) and introduce a binary flag to indicate weekends and public holidays. The cyclical encoding consists of projecting each feature space onto the unit circle to model proximity between lower and higher value of a temporal

Table 3.2 Features and label used for the baseline prediction of non-residential buildings

Feature	Name	Domain
1	Day of the week, cosine encoding	$[0, 1]$
2	Day of the week, sine encoding	$[0, 1]$
3	Hour of the day, cosine encoding	$[0, 1]$
4	Hour of the day, sine encoding	$[0, 1]$
5	Week-end and holiday flag	$\{0, 1\}$
6	Average outside temperature [°C]	$\mathbb{R}$
7	Average solar irradiance [W/m^2]	$\mathbb{R}_+$
8	Average wind speed [m/s]	$\mathbb{R}_+$

Label	Name	Domain
1	Hourly energy consumption [kWh]	$\mathbb{R}_+$

feature, e.g., 23:00 is close to 1:00, but this is not represented numerically as is. The dataset is created from proprietary data obtained through our collaboration with Hydro-Québec.

To follow the literature, we compare the WaDiRo-SCNN to a Gaussian process (GP), a WaDiRo linear regression, and a linear support vector regression (SVR). Both the GP and the linear SVR were implemented with `scikit-learn` (Pedregosa et al. 2011) while the WaDiRo-SCNN and WaDiRo linear regression have been implemented similarly to the previous experiment. We refer readers to Appendix A.3 for details on each model's hyperparameter search space and the definition of the GP kernel. We limit the hyperparameter tuning to 15 runs per model to represent industrial computational constraints and use the tree-structured Parzen estimator (TPE) algorithm proposed by (Bergstra et al. 2011) to navigate the search space. We have observed experimentally that TPE converges faster than simulated annealing to good solutions, which suits the limit on runs. The mean absolute error is used to guide the hyperparameter tuning. We also test a simple physics-constrained WaDiRo-SCNN in which we enforce that energy consumption must be greater or equal to zero.

With this experiment, we first aim to showcase how our models perform on real-world VPP data compared to the forecasting models actually used in critical industrial VPP applications. We also seek to demonstrate how the inclusion of hard physical constraints in the SCNN training affects the solution. We acknowledge that these models are applied naively and that more sophisticated methods could be preferable for this specific application.

Numerical results. We now proceed with the analysis of our numerical experiments on the prediction of buildings' consumption patterns. Figure 3.5 presents the normalized testing errors of each algorithm on each of the 16 tested buildings.

Table 3.3 presents the number of test samples violating the constraint, on non-negative energy consumption, for each model over all buildings and the percentage of deviation from the best model. We observe that the WaDiRo-SCNN performs on par with the GP as both models seem adequate to model nonlinear patterns. Still, its performance is slightly above as it has the lower MAE between the two on 10 buildings out of 16 while having similar RMSEs. Note that trained GPs may only converge to local optima. We observe that the linear SVR has the worst performance of all models as it has a consistently lower performance than the WaDiRo linear regression. We observe no significant performance gap between the unconstrained and physics-constrained WaDiRo-SCNNs. We also remark, as shown in Table 3.3, that the WaDiRo-PCSCNN has the lowest number of constraint violations throughout the experiment and that the WaDiRo-SCNN has the second lowest amount with yet 59.12% more. Additional figures, viz., the absolute errors and test prediction plots, are presented in Appendix A.3 for completeness.

We remark that both the GP and WaDiRo-SCNNs have different strengths. Our model can include hard physical constraints easily, does not require making assumptions on the type of data seen in training, and is conservative by design. GPs offer empirical uncertainty bounds and can be defined for multiple outputs.

3.8.3 Stability certification

We now design a final experiment to evaluate the stability of SCNNs trained with varying numbers of neurons and degrees of regularization.

Setting. We propose a grid search over the maximal number of neurons and the regularization parameter. We proceed on different popular real-world datasets from the UCI ML repository, namely the energy efficiency dataset from Tsanas and Xifara (2012) and the concrete compressive strength dataset from Yeh (1998). Additional tests on other real-world datasets are found on our GitHub page. The random seed for the sampling of $\mathcal{S}$ is fixed throughout the experiments, meaning that any instability that may arise from the sampling will be highlighted similarly for each model. We collect predictions in the SCNN form without mapping back to the non-convex SNN. We provide testing MAEs as well. Both the stability bounds and the MAEs are in the scale of the original dataset, but a standard scaler is used during training. As such, the inverse transform from the normalized outputs

Table 3.3 Number of constraint violations in the testing phase for each model

Model	**WaDiRo-PCSCNN**	**WaDiRo-SCNN**	**GP**	**SVR**	**WaDiRo lin. reg.**
Violations	**1431**	2277	2404	4625	3495
Deviation [%]	**0.00**	59.12	67.99	223.20	144.23

to the original scale may have an additional effect on the size of L_ε.

Numerical results. Experiment results are presented in Figures 3.6 and 3.7. We observe some general tendencies across the two datasets and the different models. The minimum and maximum of each scale are identified in the top left corner of each subfigure. First, the lowest test errors are achieved for high neuron counts and low regularization. Instability seems inversely correlated with the size of the regularization parameter. Thus, there seems to be a tradeoff between precision and stability. We note that some neurons count are *anomalously* less stable than others, e.g., 6 neurons in Figure 3.6, we believe that this is related to the population of sampling vectors used as the same outlying patterns emerge similarly for all models.

The experiment on the `energy` dataset demonstrates that neural networks can be highly unstable with lower regularization. Indeed, with the added effect of the scaling, L_ε reaches the millions. For a high number of neurons and an intermediate regularization parameter, we note that there seems to be a sweet spot where test errors are low and stability is high.

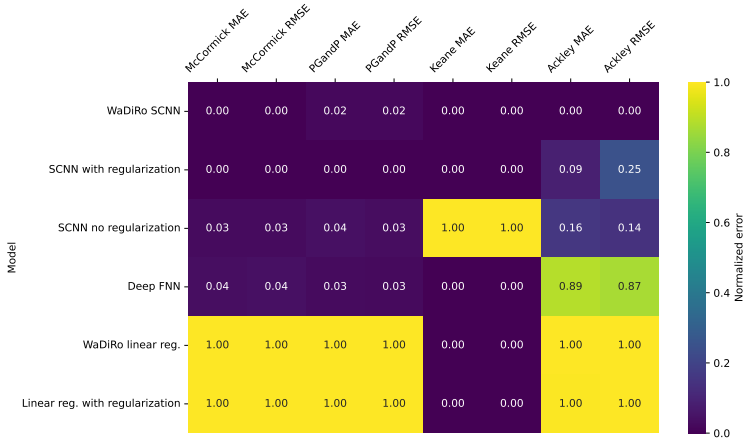
Overall, this experiment validates experimentally that our WaDiRo training methodology operates as a regularizer.

3.9 Closing remarks on Chapter 3

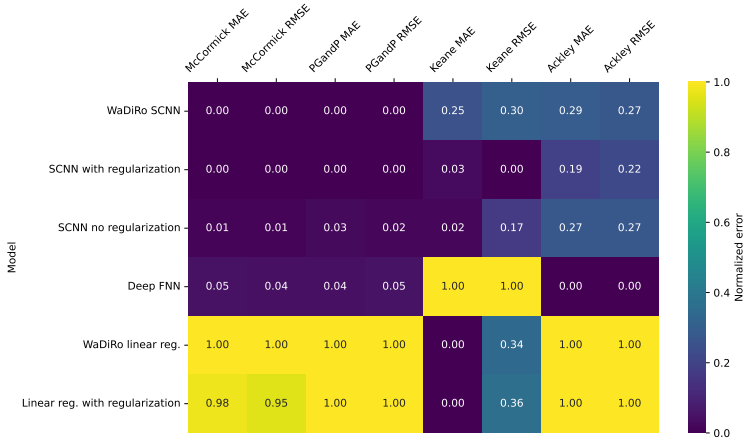
In this chapter, we reformulate the ReLU-SCNN training problem into its tractable convex order-1 Wasserstein distributionally robust counterpart by using the ℓ_1-loss function and by conditioning on the set of sampling vectors required by the SCNN. We demonstrate that the training problem has inherent out-of-sample performance guarantees. We show that our settings allow the inclusion of convex physical constraints, which are then guaranteed to be respected during training, and easy integration in mixed-integer convex post-training verification programs. These results showcase the strength and potential of convex learning for critical applications. We showcase the conservatism of our model in three experiments: a synthetic experiment with controlled data corruption, the real-world prediction of non-residential buildings' hourly energy consumption, and two datasets from the UCI ML repository on which we certify model stability.

The propositions of this chapter do have some limitations that must be highlighted. The WaDiRo-SCNN training problem is only defined for one output neuron, it is not yet GPU parallelizable, and, as most convex optimization problems, it takes an increasing amount of time to solve when augmenting its dimensionality, e.g., the maximal numbers of neurons in the hidden layers, the size of the training set, and the number of features. As such, adapting its formulation for GPU acceleration or adapting the algorithm to a more efficient solving method, similarly to work done by Mishkin

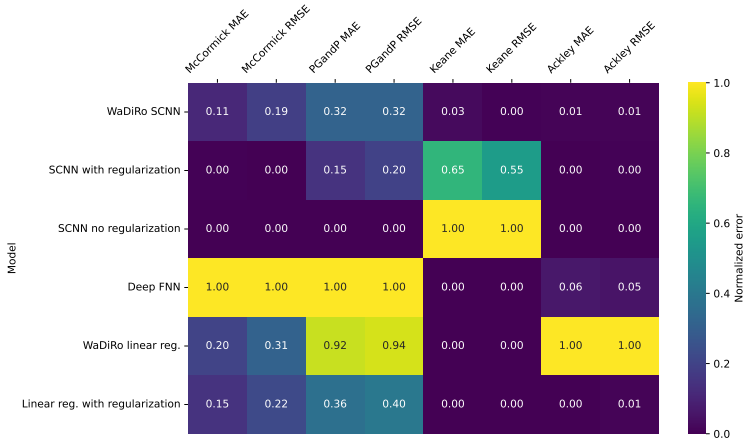
et al. (2022), Bai et al. (2023b) and more recently by Miria Feng and Pilanci (2024), would be highly interesting. We remark that the integration of physical constraints has not been covered much as it will be investigated further in future works on nonlinear optimal control.



(a) Experiment A

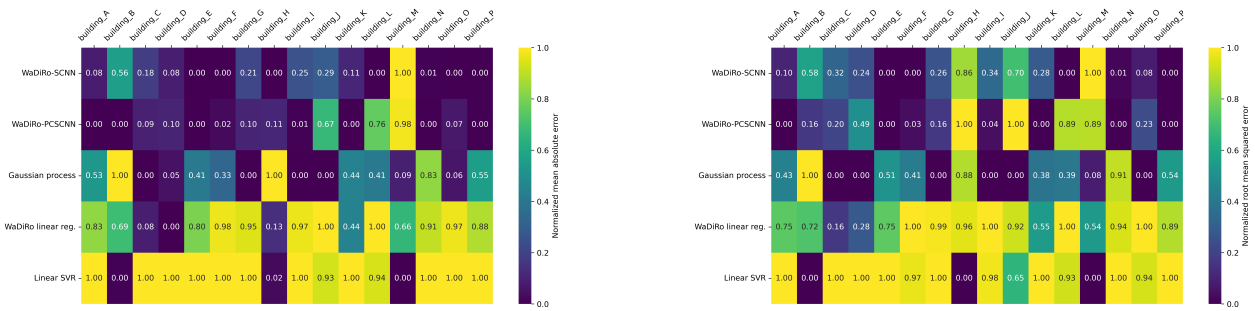


(b) Experiment B



(c) Experiment C

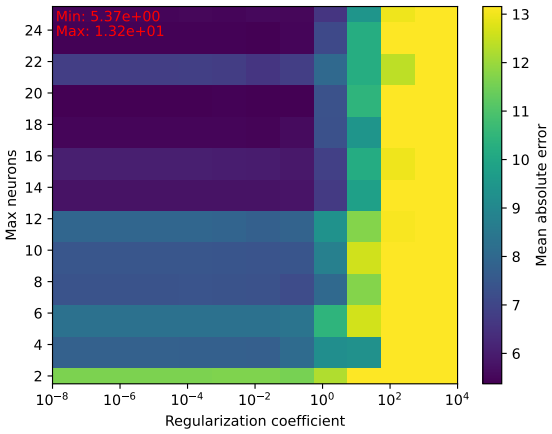
Figure 3.4 Normalized error of each synthetic experiment



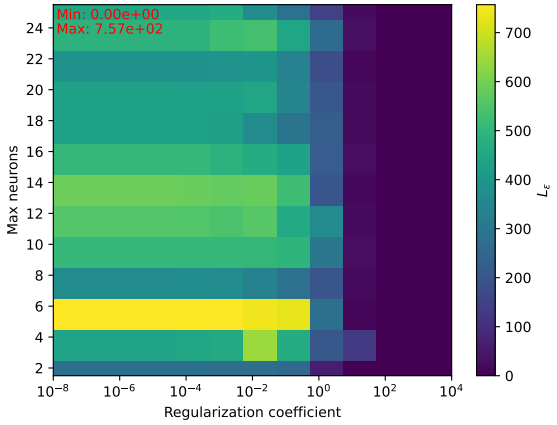
(a) Normalized MAE

(b) Normalized RMSE

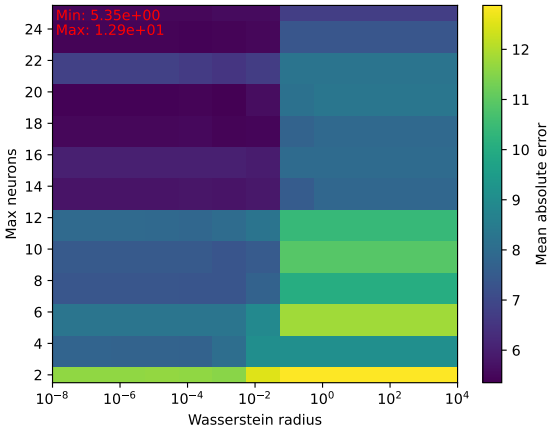
Figure 3.5 Normalized error of the experiments on non-residential buildings



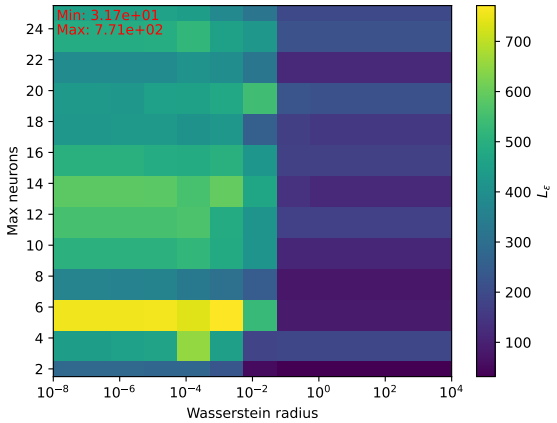
(a) MAE test: LASSO-SCNN



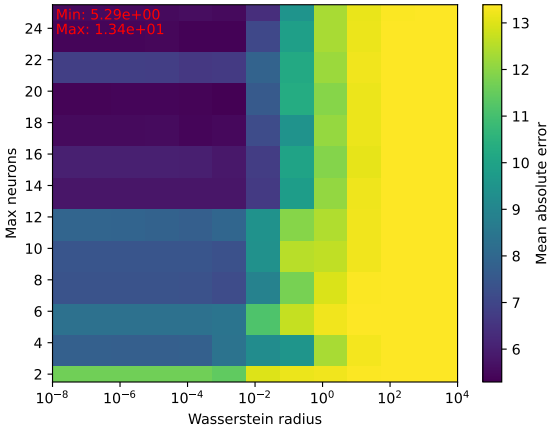
(b) L_ε: LASSO-SCNN



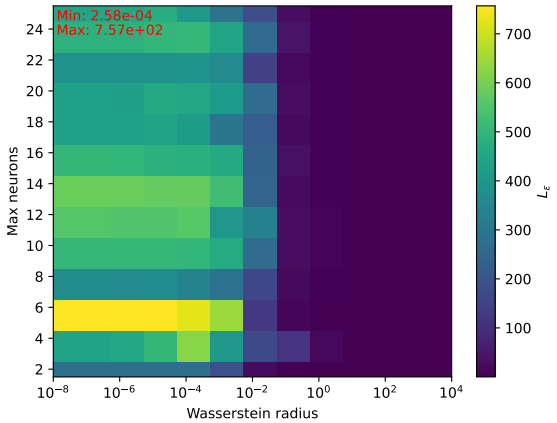
(c) MAE test: WaDiRo-SCNN $\left(W_{\|\cdot\|_1,1}\right)$



(d) L_ε: WaDiRo-SCNN $\left(W_{\|\cdot\|_1,1}\right)$

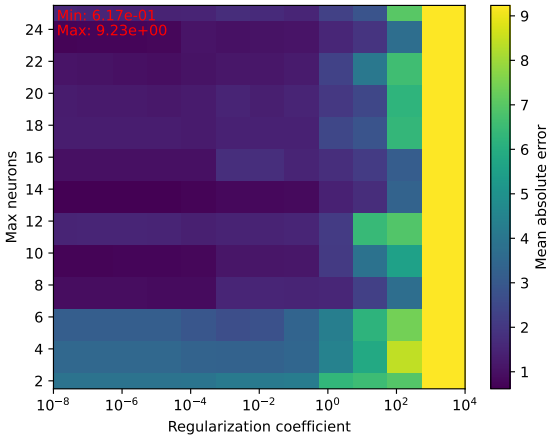


(e) MAE test: WaDiRo-SCNN $\left(W_{\|\cdot\|_2,1}\right)$

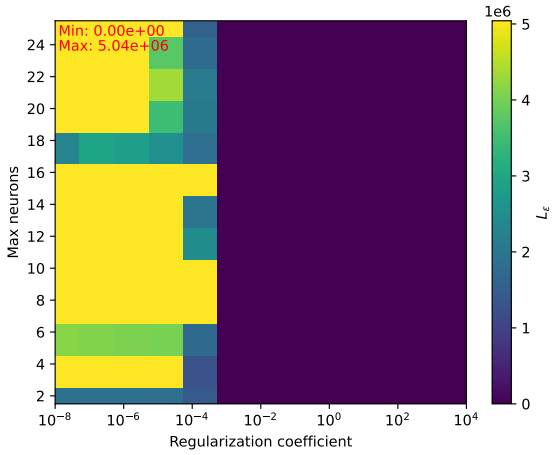


(f) L_ε: WaDiRo-SCNN $\left(W_{\|\cdot\|_2,1}\right)$

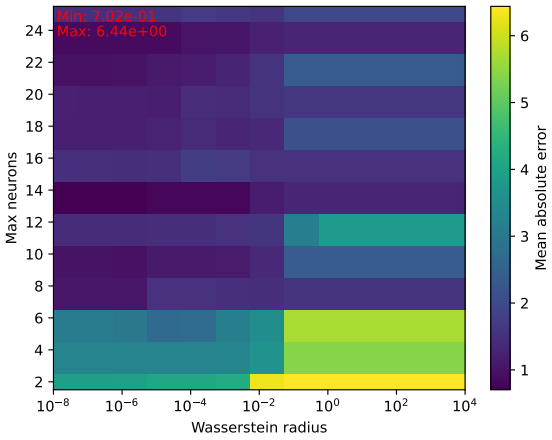
Figure 3.6 Experiments on `concrete`



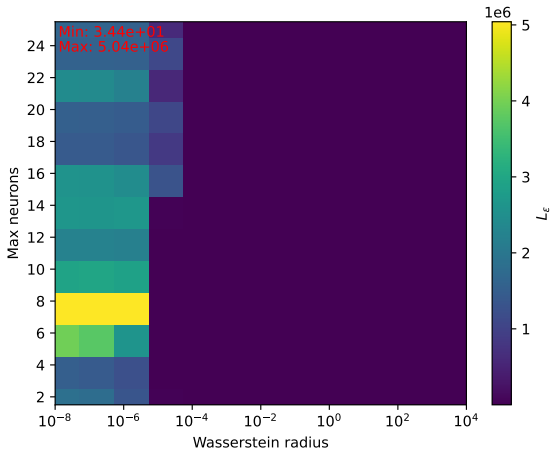
(a) MAE test: LASSO-SCNN



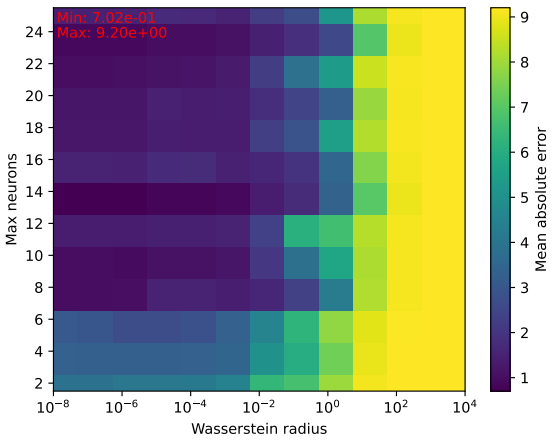
(b) L_ε: LASSO-SCNN



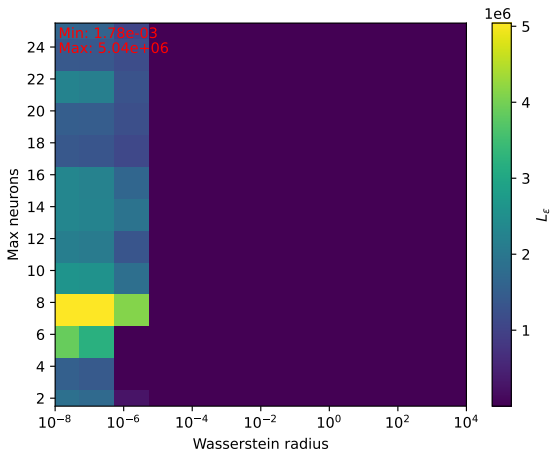
(c) MAE test: WaDiRo-SCNN $W_{\|\cdot\|_1,1}$



(d) L_ε: WaDiRo-SCNN $W_{\|\cdot\|_1,1}$



(e) MAE test: WaDiRo-SCNN $W_{\|\cdot\|_2,1}$



(f) L_ε: WaDiRo-SCNN $W_{\|\cdot\|_2,1}$

Figure 3.7 Experiments on `energy`

CHAPTER 4 SLICED-WASSERSTEIN-BASED ANOMALY DETECTION

In this chapter, we present the second part of our work. The results and general structure of the section are adapted and improved from Pallage et al. (2024) and Pallage and Lesage-Landry (2025).

Conceptually, our propositions contribute to the literature on data filtering and data selection, as highlighted in Figure 4.1.

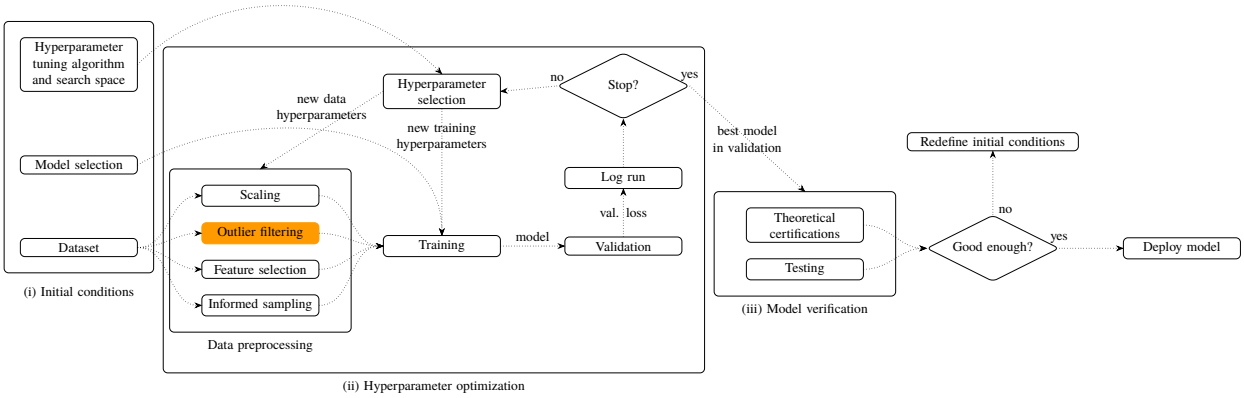


Figure 4.1 Concept from the trustworthy ML pipeline covered in Chapter 4

On a different note, we find motivation for our method in localized critical peak rebates (LCPR) DR, a unique DR mechanism from Québec, Canada, deployed by Hilo: Québec's VPP operator. In this chapter, we notably present a new open-source dataset showcasing LCPR in Montréal, Québec, Canada.

4.1 Contributions

In this chapter, we address the challenges of training ML models with unfiltered data by proposing a new unsupervised outlier filter leveraging the sliced-Wasserstein distance (Bonneel et al. 2015). We showcase the performance of this filter for training data selection.

We also address the lack of open-source datasets showcasing CPR demand response mechanisms in northern climates by releasing two years of aggregated consumption data for customers participating in an LCPR program in Montréal, Québec, Canada[1]. These customers are spread in three adjacent distribution substations. With it, we hope to stimulate research on trustworthy ML models

[1] Available at `https://github.com/jupall/lcpr-data`, as well as, `https://donnees.hydroquebec.com/explore/dataset/consommation-clients-evenements-pointe`

for demand response applications. We leverage our AD method to present a first benchmark for the prediction of the localized energy consumption of LCPR participants on our open-source dataset.

To be specific, our contributions are as follows:

- We propose a method, sliced-Wasserstein-based anomaly detection (`SWAD`), utilizing the sliced-Wasserstein distance to empirically assess the transportation cost of individual data samples with respect to the rest of the dataset. We motivate this method for training data selection and outlier filtering.
- We propose two scalable variations of `SWAD` that aim at deployment with large datasets. The first method, `split-SWAD`, parallelizes computations by working with multiple reduced-cardinality representations (splits) of the original dataset. The second method, `FEAD`, uses an Euclidian approximation.
- We motivate our Euclidian approximation by showing that it is exactly equivalent to using the Wasserstein distance when the order is 1, and its mechanism can be used as an upper bound for the order-t Wasserstein distance between empirical distributions that differ by a single sample.
- We showcase the strengths and weaknesses of our method in experiments. We start by qualitatively exposing the filtering paradigm of our methods. We then numerically compare our methods to classical AD algorithms for both anomaly detection and training data selection.
- We open the first dataset that demonstrates LCPR mechanisms in a northern climate and use our filtering method in the making of a first forecasting benchmark.

The rest of the chapter has the following structure. In Section 4.2, we introduce an example of a simplified data pipeline in IoT applications like VPPs and elaborate on the different corruption sources. In Section 4.3, we present our new empirical methods for data filtering. In Sections 4.4 and 4.5, we present a qualitative study and the different numerical experiments, respectively. Finally, we present our open-source dataset in Section 4.6 and provide closing remarks in Section 4.7.

4.2 Data corruption in data-intensive applications

Data corruption can happen at different stages of the data pipeline. To support our discussion, we present a stylized version of this pipeline in Figure 4.2 for an application involving communicating sensors and controllers, e.g., VPPs (Ruan et al. 2024). We refer readers to the work of Reis and Housley (2022) and Kleppmann (2017) for a more global and rigorous dive into the concepts of data engineering and the requirements of reliable data-intensive applications.

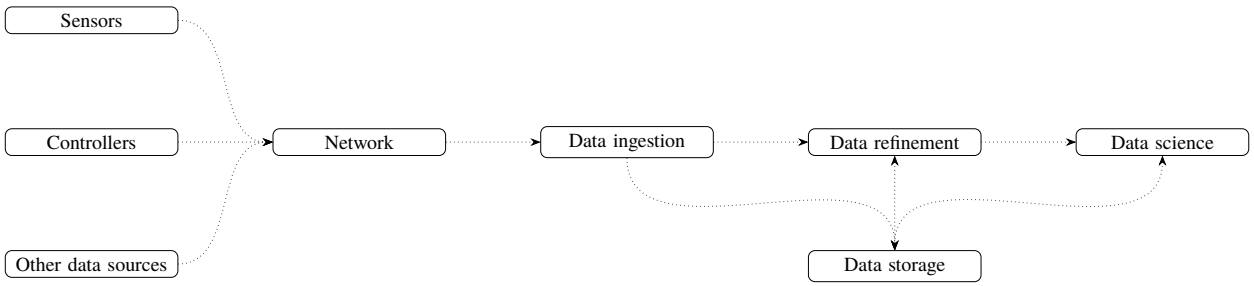


Figure 4.2 Simplified data pipeline

We now provide a general layout of the procedure, starting with data ingestion. Data ingestion is the process of collecting and importing raw data from various sources into a centralized system. This data can come from IoT devices, sensors, communicating controllers, web resources, or external databases and often requires pre-processing to remove duplicates, handle missing values, and standardize formats. Once ingested, data refinement ensures that the raw data is cleaned, structured, and transformed to be usable. This step includes filling in missing values with extrapolation methods, normalizing formats, correcting errors, and filtering out anomalies. After refinement, data science extracts knowledge from the processed data using statistical analysis, machine learning, and domain-specific techniques. This allows for pattern detection, predictive modelling, and data-driven decision-making (Reis and Housley 2022).

Corruption can occur throughout the pipeline. At the edge, sensors may produce noisy readings due to environmental interference, poor calibration, or hardware failures. As data moves through the network, telemetry errors, packet loss, and connectivity issues like internet or power outages can create gaps in the data. During ingestion, duplicates and timestamp misalignments can also introduce inconsistencies. Refinement itself can lead to numerical errors, incorrect transformations, or loss of precision. Finally, other sources like outlier events, computational bugs, human design errors, and cascading failures can mislead the analysis and introduce further biases (Kleppmann 2017).

Even though different data validation mechanisms can be implemented at each stage to limit data corruption, it is important to integrate data selection methods into the machine learning process to promote a foolproof procedure and enhance general trustworthiness.

4.3 Main proposition

Recall from Section 3.7 the definition of the finite approximation of the sliced-Wasserstein distance:

$$SW_{\|\cdot\|,t}\left(\mathbb{U},\mathbb{V}\right) \approx \left(\frac{1}{L}\sum_{l=1}^{L} W_{\|\cdot\|,t}\left(\mathfrak{R}\mathbb{U}\left(\cdot,\theta_l\right),\mathfrak{R}\mathbb{V}\left(\cdot,\theta_l\right)\right)^t\right)^{1/t},$$

with $\boldsymbol{\theta}_l \in \mathbb{S}^{d-1} = \{\boldsymbol{\theta} \in \mathbb{R}^d | \sqrt{\theta_1^2+\theta_2^2+\cdots+\theta_d^2} = 1\}$.

The SW distance has some interesting relation with the Wasserstein distance as demonstrated by the following theorem from Bonnotte (2013).

Theorem 1 (Equivalence of $SW_{\|\cdot\|_t,t}$ and $W_{\|\cdot\|_t,t}$, (Bonnotte 2013))**.** *There exists a constant $C(d,t) > 0$ such that, for all $\mathbb{U},\mathbb{V} \in \mathcal{P}(B(0,R))$, where $B(0,R)$ is the closed ball of radius R in $\mathbb{R}^d$:*

$$SW_{\|\cdot\|_t,t}(\mathbb{U},\mathbb{V})^t \leq c(d,t)W_{\|\cdot\|_t,t}(\mathbb{U},\mathbb{V})^t \leq C(d,t)R^{t-1/(d+1)}SW_{\|\cdot\|_t,t}(\mathbb{U},\mathbb{V})^{1/(d+1)},$$

and where $0 < c(d,t) = \frac{1}{d}\int_{\mathbb{S}^{d-1}} \|\theta\|_t^t \,\mathrm{d}\theta \leq 1$.

Theorem 1 effectively shows that the sliced-Wasserstein metric can be used to bound the Wasserstein distance to the power of t.

We now utilize the finite approximation of the SW distance to formulate a simple outlier filter. Let $\mathcal{D} = \bigcup_{i=1}^{N}\{\mathbf{z}_i\}$ be a dataset of N independent samples $\mathbf{z}_i \in \mathbb{R}^d$. Consider an empirical distribution $\hat{\mathbb{P}}_N = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{z}_i}$, where $\delta_{\mathbf{z}_i}$ is a Dirac delta function assigning a probability mass of one at each known sample $\mathbf{z}_i$. Let the notation $\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_i}$ denote a variation of $\hat{\mathbb{P}}_N$ in which we remove sample $\mathbf{z}_i$. As the weight of each sample is identical and because the SW distance compares distributions of equal weights, we propose an outlier filter by using a voting system in which we compare the SW distance between the empirical distribution minus the outlier candidate and the empirical distribution minus a random sample. Let $\mathcal{O} \subseteq \mathcal{D}$ and $\mathcal{O}^{\mathrm{c}} \subseteq \mathcal{D}$ denote the sets of outlier and inlier samples, respectively, under the perspective of the filter; where $\mathcal{D} = \mathcal{O} \cup \mathcal{O}^{\mathrm{c}}$. A vote is positive if the distance between the two empirical distributions exceeds the threshold $\epsilon > 0$. A sample is labelled an outlier if the proportion of positive votes is greater or equal to the threshold p.

$$\mathcal{O} = \left\{ \mathbf{z}_i \in \mathcal{D} \;\middle|\; p \leq \frac{1}{n} \sum_{\mathbf{z}_j \sim \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_i} : j \in [\![n]\!]} \mathbb{1}\left(SW_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_i},\, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_j}) \geq \epsilon\right), \forall i \in [\![N]\!] \right\}, \qquad \texttt{(SWAD)}$$

where $\mathbb{1}(\cdot)$ denotes an indicator function, n is the number of points used for the vote, $p \in [0,1]$ is the voting threshold required to label a sample as an outlier, and $[\![N]\!] \equiv \{1,2,\ldots,N\}$.

This filter is interesting for several reasons. It is unsupervised, purely analytical, and uses a well-known explainable mathematical distance function to filter data points that seem out-of-sample un-

der the SW metric. The intuition is that we remove samples that are costly in the transportation plan when compared to other random samples of the same distribution. We can use the voting percentage to measure the algorithm's confidence in its labelling. It is also parallelizable, as seen in our implementation provided on our GitHub page[2].

Unfortunately, this method does not scale well with large datasets as it is: the SW distance computational burden increases when larger datasets are processed. We thus propose two methods to accelerate computations for real-world applications.

4.3.1 Smart splitting

We start by proposing a *smart splitting* variation. Instead of considering the whole dataset, we first use a clustering method, e.g., KNN, to obtain $K \in \mathbb{N}$ clusters. We then randomly split each cluster into S splits and assemble S reduced-cardinality representations of our original dataset $\mathcal{D}$ by collecting one split from each cluster. We denote each split of the original dataset by $\mathcal{D}_s$ such that $\bigcup_{s=1}^{S} \mathcal{D}_s = \mathcal{D}$. This *smart* splitting allows us to parallelize the outlier detection by treating each split concurrently. It is a viable option when multiple compute threads are available.

Abusing of notation, consider splits of the empirical distribution $\hat{\mathbb{P}}_{|\mathcal{D}_s|} = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{z}_i \in \mathcal{D}_s} \delta_{\mathbf{z_i}}(\mathbf{z})$ where $|\mathcal{D}_s|$ denotes the cardinality of the split s, $\forall s \in [\![S]\!]$. The methodology is now as follows:

$$\mathcal{O}_s = \left\{ \mathbf{z}_i \in \mathcal{D}_s \;\middle|\; p \leq \frac{1}{n_s} \sum_{\mathbf{z}_j \sim \hat{\mathbb{P}}^{-\mathbf{z}_i}_{|\mathcal{D}_s|} : j \in [\![n_s]\!]} \mathbb{1}\left(SW_{\|\cdot\|,t}(\hat{\mathbb{P}}^{-\mathbf{z}_i}_{|\mathcal{D}_s|-1}, \hat{\mathbb{P}}^{-\mathbf{z}_j}_{|\mathcal{D}_s|-1}) \geq \epsilon_s \right), \forall i \in [\![|\mathcal{D}_s|]\!] \right\}, \quad (4.1)$$

where $n_s = \frac{|\mathcal{D}_s|}{|\mathcal{D}|} n$ and $\epsilon_s = \frac{|\mathcal{D}_s|}{|\mathcal{D}|} \epsilon$ and from which we assemble an approximation of our original outlier set such that:

$$\mathcal{O} \approx \bigcup_{s=1}^{S} \mathcal{O}_s. \qquad \texttt{(split-SWAD)}$$

4.3.2 Fast Euclidian approximation

We now present another approximation for when computational efficiency is the main requirement, e.g., large datasets, limited amount of computational power, and low number of threads.

We remark that the transportation plan between a distribution minus a sample and the same distribution minus another sample can be roughly approximated by the norm between the two removed samples. We clarify this statement with some deeper explanations, which ultimately lead to Proposition 6.

[2] `https://github.com/jupall/swfilter`

Consider the order-t Wasserstein distance between two empirical distributions, $\hat{\mathbb{U}}_N$ and $\hat{\mathbb{V}}_N$, constructed with the same number of samples N, denoted $\mathbf{u}_i$ and $\mathbf{v}_i$, respectively $\forall i \in [\![N]\!]$:

$$W_{\|\cdot\|,t}(\hat{\mathbb{U}}_N, \hat{\mathbb{V}}_N) = \inf_{\pi} \left(\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{u}_i - \mathbf{v}_{\pi(i)}\|^t \right)^{1/t}, \tag{4.2}$$

over π, all possible permutations of N elements and for $t \geq 1$. In this setting, there are $N!$ possible transportation plans. In our case, we consider two distributions which only differ by a single sample, viz., $\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}$ and $\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}$ for $l, k \in \mathbb{N}$ with $k \neq l$. Visualize the two distributions as scatter plots. Only one difference should appear between the two: in one plot, $\mathbf{z}_k$ appears and $\mathbf{z}_l$ is missing; in the latter, we see the opposite situation. The easiest transportation plan to conceive, between each plot, is to send $\mathbf{z}_l$ to $\mathbf{z}_k$ without moving other samples. We denote this suboptimal single-sample transportation plan as $\tilde{W}$. The plan $\tilde{W}$ is as follows:

$$\begin{aligned} W_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) \leq \tilde{W}_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) &= \left(\frac{1}{N-1}(0 + 0 + \ldots + 0 + \|\mathbf{z}_k - \mathbf{z}_l\|^t) \right)^{1/t} \\ &= \frac{\|\mathbf{z}_k - \mathbf{z}_l\|}{\sqrt[t]{(N-1)}}, \end{aligned} \tag{4.3}$$

for $l, k \in [\![N]\!]$ with $k \neq l$. Note that $\tilde{W}$ is generally suboptimal because the power of norms do not respect the triangle inequality for $t > 1$ (Carlen et al. 2021). Thus, a more cost-effective transport plan could involve the displacement of more than one sample. Nevertheless, it is an intuitive and computationally efficient transportation. In the special case where $t = 1$, we can show that the inequality becomes an equality, as shown next.

Lemma 3. *For $t = 1$, the following inequality holds:*

$$W_{\|\cdot\|,1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) = \tilde{W}_{\|\cdot\|,1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) \quad \forall l, k \in [\![N]\!] : k \neq l.$$

Proof: Denote by $\mathbf{z}_{j_i}$ any intermediate samples in the transport of $\mathbf{z}_l$ to $\mathbf{z}_k$ such that $j_i \neq l \neq k \quad \forall l, k, j_i \in [\![N]\!] \, \forall i \in [\![N-1]\!]$ where index i is used to differentiate distinct intermediate samples.

Using this notation and the triangle inequality, we can show that:

$$\begin{aligned}\|\mathbf{z}_k - \mathbf{z}_l\| &= \|\mathbf{z}_k - \mathbf{z}_{j_1} + \mathbf{z}_{j_1} - \mathbf{z}_l\| \\ &\leq \|\mathbf{z}_k - \mathbf{z}_{j_1}\| + \|\mathbf{z}_{j_1} - \mathbf{z}_l\| \\ &\leq \|\mathbf{z}_k - \mathbf{z}_{j_1}\| + \|\mathbf{z}_{j_1} - \mathbf{z}_{j_2}\| + \|\mathbf{z}_{j_2} - \mathbf{z}_l\| \\ &\leq \|\mathbf{z}_k - \mathbf{z}_{j_1}\| + \|\mathbf{z}_{j_1} - \mathbf{z}_{j_2}\| + \|\mathbf{z}_{j_2} - \mathbf{z}_{j_3}\| + \|\mathbf{z}_{j_3} - \mathbf{z}_l\| \dots \\ &\leq \dots\end{aligned} \tag{4.4}$$

which means that for $t = 1$, the single sample transport is smaller or equal to any other permutation involving the transport of more than one sample.

From (4.4), we conclude that $\tilde{W}_{\|\cdot\|,1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) = W_{\|\cdot\|,1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l})$ and thus it follows from (4.3) that $\tilde{W}_{\|\cdot\|,t>1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) \geq W_{\|\cdot\|,t>1}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l})$. □

To continue, we will need the following lemma.

Lemma 4 (Remark 6.6 Villani (2008))**.** *For $1 \leq t \leq q < \infty$, the following inequality stands*

$$W_{\|\cdot\|,t} \leq W_{\|\cdot\|,q}.$$

Using this inequality, we now bound $W_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l})$.

Proposition 6. *For $1 \leq t < \infty$ and $\forall l, k \in [\![N]\!] : k \neq l$, we obtain the following bounds:*

$$\frac{\|\mathbf{z}_k - \mathbf{z}_l\|}{(N-1)} \leq W_{\|\cdot\|,t}(\hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_k}, \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_l}) \leq \frac{\|\mathbf{z}_k - \mathbf{z}_l\|}{\sqrt[t]{(N-1)}} \tag{4.5}$$

Proof: The proof follows from combining Lemmata 3 and 4 as well as (4.3). □

We present a visual representation of Proposition 6 on an empirical measure of 100 samples generated with a Gaussian distribution. In Figure 4.3, we compare the Wasserstein distance between the empirical measure diminished by a random sample and the empirical measure diminished by another sample. As can be seen, the property holds.

As such, based on Proposition 6, we introduce our fast Euclidian anomaly detection method for $W_{\|\cdot\|_2,t}$:

$$\mathcal{O} \approx \left\{ \mathbf{z}_i \in D \;\middle|\; p \leq \frac{1}{n} \sum_{\mathbf{z}_j \sim \hat{\mathbb{P}}_{N-1}^{-\mathbf{z}_i}} \mathbb{1}\left(\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2}{\sqrt[t]{N-1}} \geq \eta \right), \forall i \in [\![N]\!] \right\}, \tag{FEAD}$$

where η is the threshold of the Euclidian distance. This method might not be as accurate as (SWAD) to filter out-of-sample data points when $t \neq 1$, but it is extremely fast and has a direct relation with

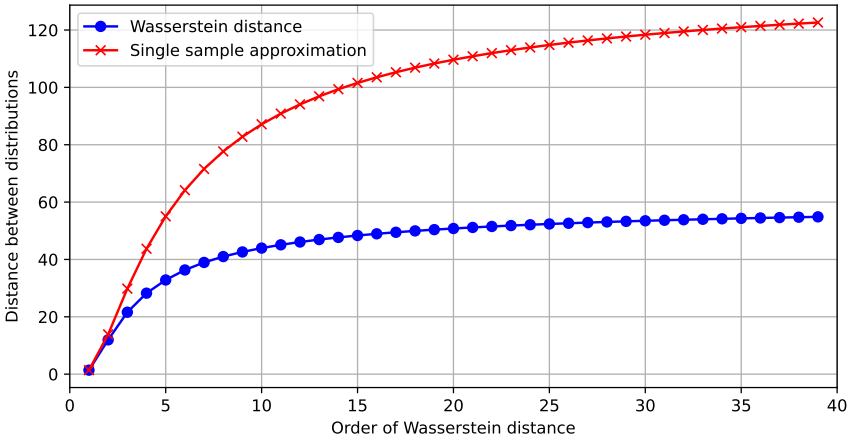


Figure 4.3 Empirical validation of Proposition 6

the Wasserstein distance. Because (SWAD) and (FEAD) use the same principle, they share similar classification patterns when ϵ and η are tuned accordingly, as can be seen in Figure 4.6 of Section 4.4. We remark that, in practice, the order-t used in each outlier filtering method is a hyperparameter that can be chosen empirically.

4.4 Qualitative study

We first begin this section by showing the AD mechanism of the SW filter on a small two-dimensional example. We generate three Gaussian distributions with different population sizes. As shown in Figure 4.4, the first distribution is the majority group, the second is the minority group, and the third represents clear statistical outliers. We now merge the three distributions into a single one to test our SW filter. We vary the radius of the Wasserstein ball to see how the filter behaves qualitatively. The results are presented in Figure 4.5. We observe that we can generate three filtering scenarios by modifying the value of ϵ. With $\epsilon = 0.1$, we filter only the statistical outliers. With $\epsilon = 0.05$, we only keep the majority group. And, with $\epsilon = 0.01$, we only keep the samples closest to the barycenter of the majority group. This is very interesting in a safe ML pipeline as we can tune the *conservatism* of the training dataset that is used at each run during hyperparameter optimization.

In Figure 4.6, we compare different AD algorithms on synthetic datasets provided in `scikit-learn`'s example collection (Pedregosa et al. 2011). Recall that most of these algorithms were introduced in Section 2.2 except for Robust Covariance, which is a method originating from `scikit-learn` (Pedregosa et al. 2011). Hyperparameters are fixed for each algorithm to see how a single hyperparameter choice influences the labelling on each dataset. As can be seen, the SW filter and its fast Euclidian approximation are better at isolating outliers when there is a clear majority

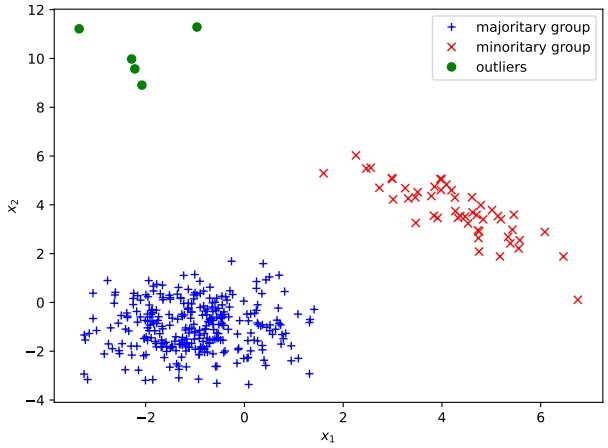
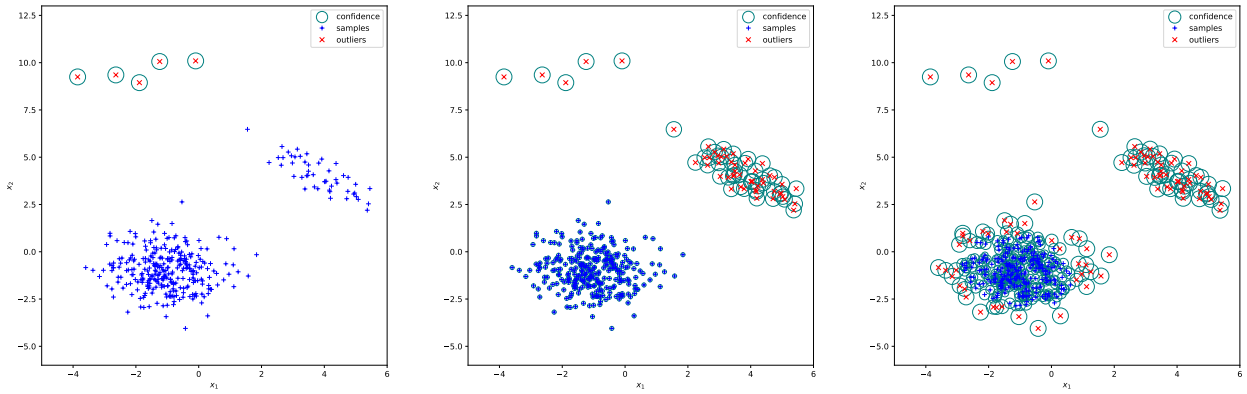


Figure 4.4 Illustration of different groups



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$ (c) $\epsilon = 0.01$

Figure 4.5 Labelling of the SW filter for different values of ϵ

group but are not as precise in identifying local outliers based on local density like, e.g., one-class SVM.

4.5 Numerical experiments

We now evaluate the performance of our methods for anomaly detection and data selection.

4.5.1 Numerical study for anomaly detection

We run a thorough experiment with commonly used real-world benchmark datasets. We run experiments on the *Lymphography*, *Ionosphere*, *Glass*, *Shuttle*, *WPBC*, *Arrhythmia*, and *Pima*, datasets

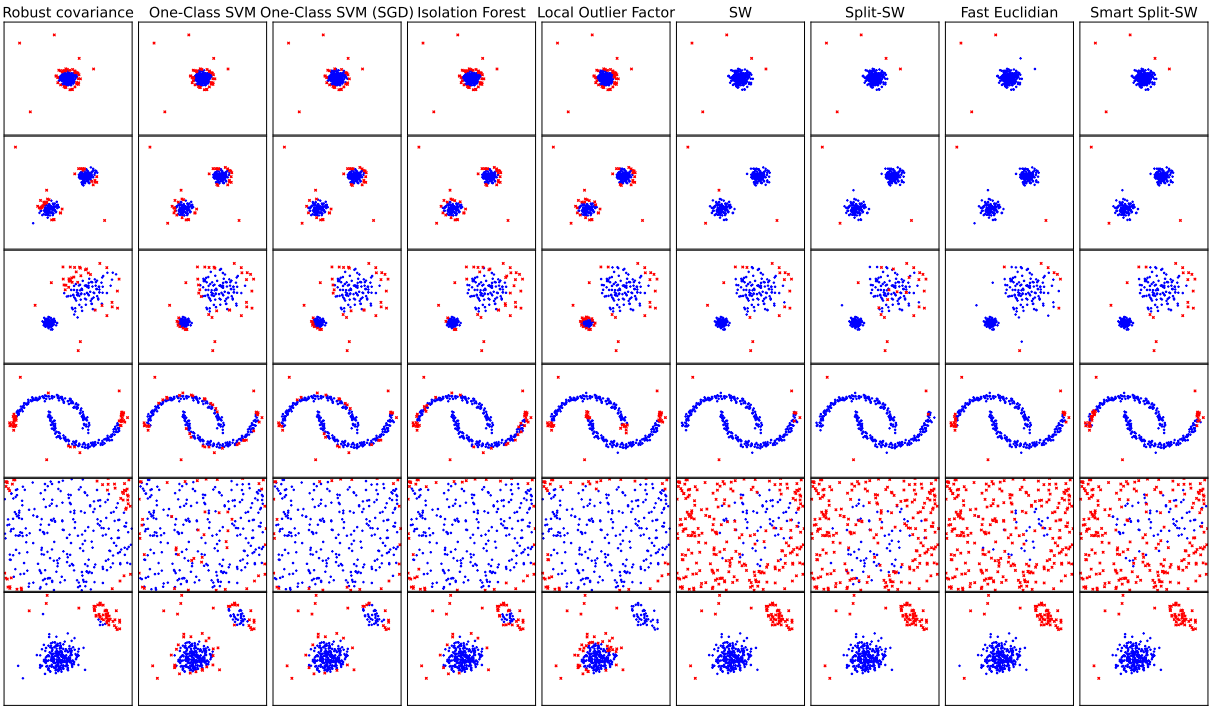
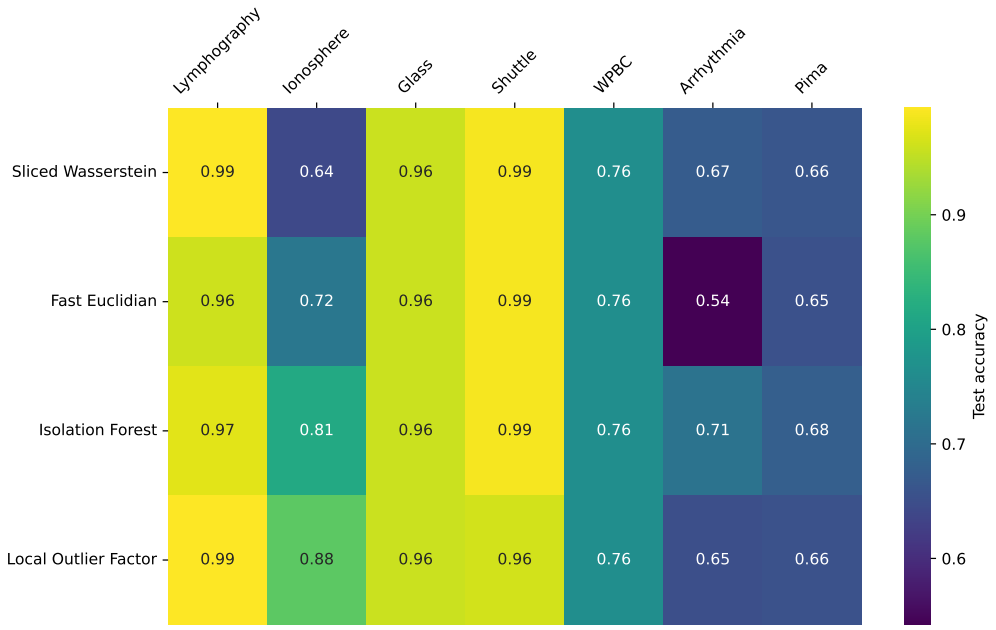


Figure 4.6 AD comparison on multiple synthetic datasets

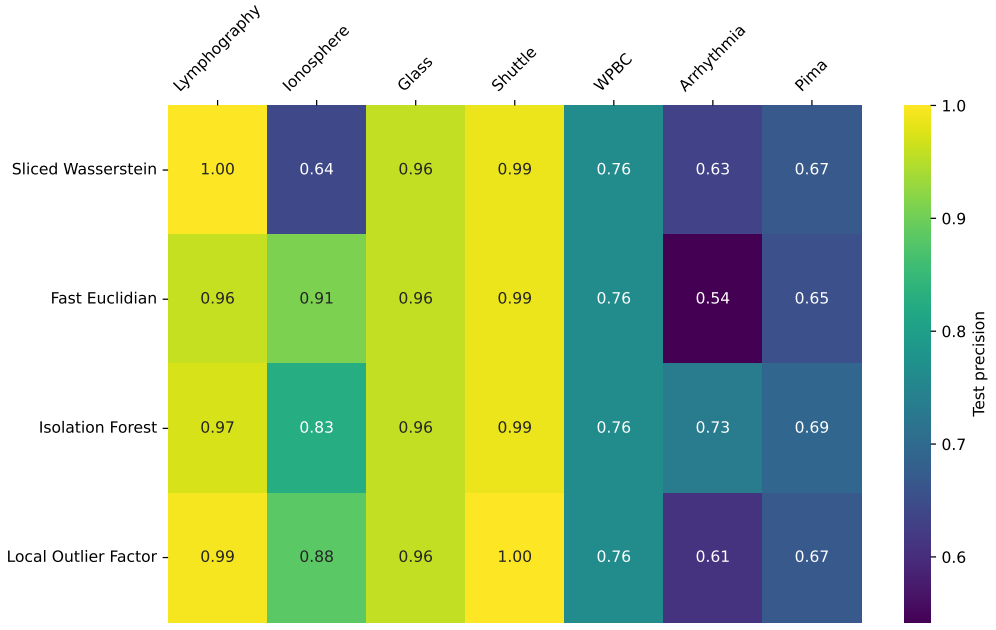
presented in Campos et al. (2016) for AD benchmarking. We compare isolation forest and LOF to order-2 (`SWAD`) and (`FEAD`). We omit (`split-SWAD`) because the size of the experiment permits the use of (`SWAD`). We implement a grid search with `hyperopt` (Bergstra et al. 2015) and select each model's run with the best accuracy A. We then extract the precision score P from this run. The performance indicators are defined as follows:

$$A = \frac{T_\mathrm{p} + T_\mathrm{n}}{T_\mathrm{p} + F_\mathrm{p} + T_\mathrm{n} + F_\mathrm{n}}, \qquad P = \frac{T_\mathrm{p}}{T_\mathrm{p} + F_\mathrm{p}},$$

where T_p, F_p, T_n, and F_n stand for true positives, false positives, true negatives, and false negatives, respectively. In all our tests, we use the order-2 sliced-Wasserstein distance for our method. The results are presented in Figure 4.7. We observe a similar performance between each model, except on the *Ionosphere* dataset where the SW filter lags behind the other models, and on *Arrhythmia* where the fast Euclidian approximation is underperforming. The SW filter's strength is that it considers the global distributional properties of the population to guide its labelling. We remark that our method fails at detecting local outliers as it is purely designed to locate global outliers that are *risky* to keep in the training set. This is a reasonable choice to make when designing data preprocessing methods for safer ML training, but not necessarily to filter local outliers. Every experiment is available on our GitHub page, hyperparameters are provided in Appendix B.1.



(a) Accuracy A



(b) Precision P

Figure 4.7 Results of the grid search for each AD model on each dataset

4.5.2 Numerical study for training data selection

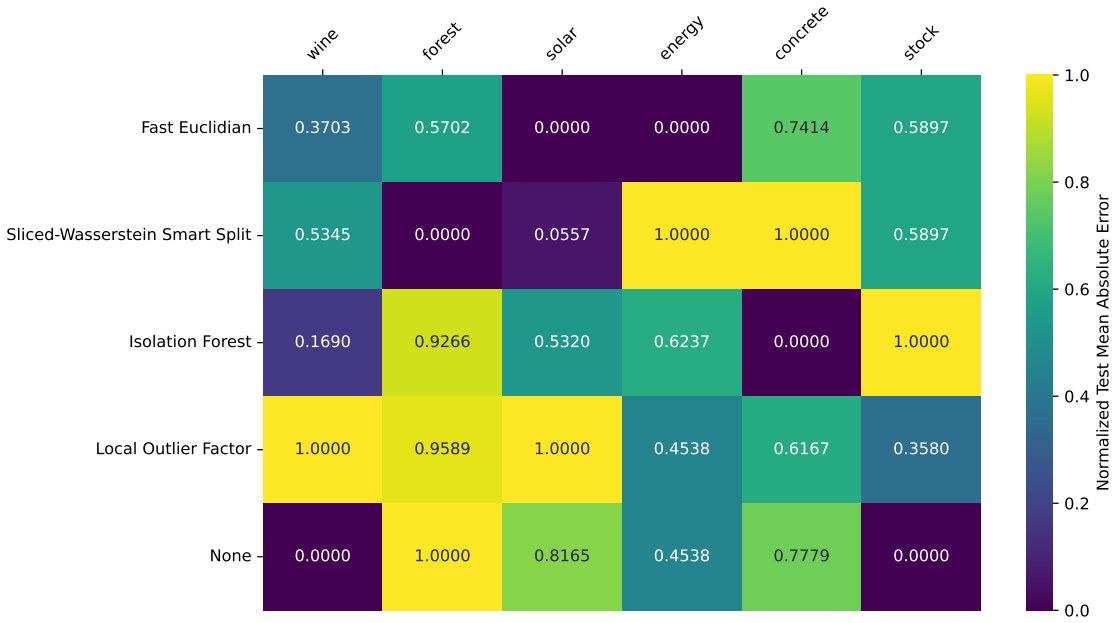
We now test our method for training data selection. We use regression datasets found on the UCI ML repository (Kelly et al. 2023). To keep computation requirements low, we compare (`FEAD`), (`split-SWAD`), LOF, Isolation Forest, and no filtering. For each dataset, 10% of the data is kept for the testing phase. From the remaining 90%, we use 30% of it during validation and train on the rest. Both training and validation samples are corrupted by a Gaussian noise applied after data scaling. The Gaussian noise is centred at the origin with a variance of 0.05 on all features and the label. In each trial, we train a non-regularized shallow convex neural network constructed with a maximum of 25 neurons in the hidden layer. The SCNN is trained using the ℓ_1-loss function on the data filtered by each AD method. We allow 100 trials per filtering method per dataset and use the tree-structured Parzen estimator for hyperparameter optimization (Bergstra et al. 2011). The full hyperparameter search space is available in Appendix B.2. The datasets comprise `wine` (Cortez et al. 2009), `forest` (Cortez and Morais 2007), `solar` (Brashaw 1989), `energy` (Tsanas and Xifara 2012), `concrete` (Yeh 1998), and `stock` (Akbilgic 2013).

We repeat each experiment four times on different testing splits and extract the average results. The normalized average MAE obtained in testing for the best trials in validation are displayed in Figure 4.8, where we also provide the average number of training samples filtered by each method. In each column, the highest MAE is equal to 1, and the lowest is equal to 0. Note that, for transparency, the absolute errors can be seen in Figure B.1 from Appendix B.2.

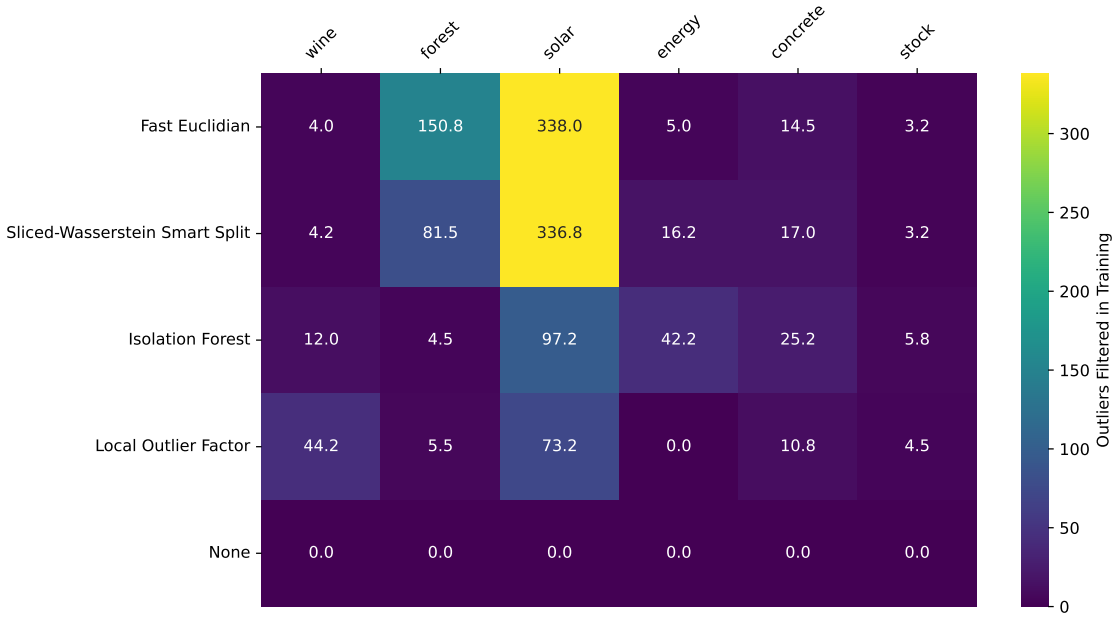
From Figure 4.8, we make the following observations. While no method completely outperforms the others, we remark that (`FEAD`) never has the highest error compared to other methods as they have at least one column equal to 1. Surprisingly, the no-filtering method has the lowest error on two datasets: we believe that the use of the ℓ_1-loss and the decision to train SCNNs may have helped to avoid overfitting during training and validation. Nevertheless, as seen in Table 4.1, we observe that (`FEAD`) has by far the lowest average normalized errors over the six datasets.

Table 4.1 Average normalized error per algorithm for the data selection experiment

Algorithm	**Average normalized error**
Fast Euclidian	0.3786
SW Smart Split	0.5300
Isolation Forest	0.5419
Local Outlier Factor	0.7312
None	0.5081



(a) Normalized MAE in testing



(b) Average number of samples filtered in the training set

Figure 4.8 Results of the data selection experiment

4.6 Open source dataset

The dataset we share contains the aggregated hourly consumption of 197 anonymous LCPR testers located in three neighbouring substations of Montréal, Québec, Canada. The dataset was developed in collaboration with Hydro-Québec with the data from Hilo. Additional hourly weather data and LCPR information are also present. Table 4.2 details the features and the label. Note that we also provide cyclical encoding of temporal features, e.g., month, day of the week, and hour. We remark that outliers and anomalies are present in the dataset because of metering and telemetry issues or even blackouts, e.g., an aberrant (and impossible) 32.2 MWh energy consumption is registered at some point.

Table 4.2: Description of features and label of the dataset

Name	**Description**	**Possible values**
substation	Substation identifier	{'A', 'B', 'C'}
timestamp_local	Timestamp in local time (UTC-5) and ISO 8601 format [AAAA-MM-DD hh:mm:ss]	–
connected_clients	Number of clients connected to the substation during the considered hour	$\{9, 10, \ldots, 104\}$
connected_smart_tstats	Number of smart thermostats connected to the substation during the considered hour	$\{59, 60, \ldots, 1278\}$
average_inside_temperature	Hourly average indoor temperature measured by smart thermostats in substation [°C]	$[16.21, 27.08]$
average_temperature_setpoint	Hourly average setpoint of smart thermostats in substation [°C]	$[9.31, 21.03]$
average_outside_temperature	Hourly average outside temperature at substation [°C]	$[-32.0, 35.2]$
average_solar_radiance	Hourly average solar radiance at substation [W/m^2]	$[0, 961]$
average_relative_humidity	Hourly average relative humidity at substation [%]	[0,100]
average_snow_precipitation	Hourly average amount of snow precipitation at substation [mm]	[0.0,306.0]
average_wind_speed	Hourly average wind speed at substation [m/s]	[0, 15.68]
date	Date [AAAA-MM-DD]	[2022-01-01, 2024-06-30]
month	Month	$\{1, 2, \ldots, 12\}$
day	Day of the month	$\{1, 2, \ldots, 31\}$

Continued on next page

Table 4.2: Description of features and label of the dataset (Continued)

day_of_week	Day of the week with Sunday and Saturday being 1 and 7, respectively	$\{1, 2, \ldots, 7\}$
hour	Hour of the day	$\{0, 1, \ldots, 23\}$
challenge_type	Type of challenge during the given hour	{'None', 'CPR', 'LCPR'}
challenge_flag	Flag indicating hours in challenge	$\{0, 1\}$
pre_post_challenge_flag	Flag indicating hours in pre-challenge or post-challenge	$\{0, 1\}$
is_weekend	Flag indicating weekends	$\{0, 1\}$
is_holiday	Flag indicating Québec holidays	$\{0, 1\}$
weekend_holiday	Flag indicating whether a weekend or a holiday	$\{0, 1\}$
total_energy_consumed	Hourly energy consumption of the substation [kWh]	$[7.45, 32240.17]$

We refer readers to Appendix B.3 for additional analyses and visualizations of the dataset's features and labels.

4.6.1 First benchmark

We now propose a first benchmark on our LCPR dataset. Our goal is to predict the aggregated hourly consumption at each substation during winter when peak demand is critical. To follow the literature (Weng and Rajagopal 2015), and propose a simple yet meaningful benchmark, we implement a Gaussian process (Williams and Rasmussen 1995) in a rolling horizon fashion. Samples dating from before 2023-12-15 are used for hyperparameter tuning, while those between 2023-12-15 and 2024-04-15, during the most recent winter DR season, are used for testing. We train one model per week and the training window size is a hyperparameter to be tuned. Because of its good performance in the data selection experiment of Section 4.5.2 and its low computational needs, we use (FEAD) to preprocess the data during hyperparameter optimization.

Our method is interesting because it can filter clear out-of-sample points while hopefully avoiding

sparse LCPR events that other methods could consider as local outliers. We refer interested readers to our GitHub page for the full overview of the hyperparameter search space and the construction of the kernel used in GP training. The kernel combines a radial basis function kernel, an exponential sine squared kernel to model periodicity, and a white noise kernel. Testing mean absolute errors (MAE) and root mean squared errors (RMSE) are presented in Table 4.3 for each substation.

Table 4.3 Absolute test errors of the best validation run for each substation

Substation	**A**	**B**	**C**
MAE [kWh]	20.49103	17.90663	41.21746
RMSE [kWh]	26.08270	22.39355	51.25044

We remark that the errors for substation C are higher than for substations A and B, but the value of its label is also higher. During testing, substations A and B record consumptions between approximately 50kWh and 300kWh compared to approximately 100kWh to 600kWh for substation C. See Appendix B.4 to visualize the test predictions for each substation.

4.7 Closing remarks on Chapter 4

In this section, we propose an anomaly detection and data selection method based on the sliced-Wasserstein distance. To keep computational tractability for larger datasets, we propose two approximations: (i) a split method in which we decompose the original dataset in multiple reduced-cardinality representations (splits), (ii) a fast method based on the Euclidian distance. We remark that method (i) is especially interesting when multi-threading is available.

We observe that our proposition is good at filtering global outliers yet sometimes fails at detecting local outliers. We argue that this is quite interesting to robustify adversarial training datasets. We insist that like most methods in computer science, our proposition is not a *jack-of-all-trades* but it is quite interesting for specific kinds of data corruption, relies on a proven mathematical distance, is unsupervised, and can easily be included in the hyperparameter optimization phase of a pre-deployment ML pipeline. One of the biggest limitations of this method is the lack of proper theoretical guarantees for out-of-sample model predictive performance. We leave this for future work.

We also propose a new open-source dataset on LCPR. We wish to highlight the help of Christophe Bélanger from Hydro-Québec during that process.

CHAPTER 5 CONCLUSION

We conclude this Master's thesis by first reviewing our contributions. In Chapter 1, we summarize the core ideas and challenges behind trustworthy machine learning pipelines and virtual power plants. In Chapter 2, we outlined the literature and positioned our propositions in it. In Chapter 3, we propose a low-stochasticity distributionally robust training procedure for shallow convex neural networks. We obtain theoretical guarantees for the out-of-sample performance of this training procedure using the Rademacher complexity. We show that our training procedure can also accommodate convex physical constraints. We obtain a mixed-integer convex post-training verification framework for SCNNs to certify their worst-case output deviation under small perturbations, allowing us to study the effect of different regularizers on model stability. In Chapter 4, we propose a new empirical method to filter training datasets using the sliced-Wasserstein distance. While this method is not adapted for all scenarios and datasets, we show experimentally that it compares to other classical anomaly detection methods. In this chapter, we also propose a new open-source dataset showcasing localized critical peak rebate demand response in Montréal. This dataset is extremely interesting for benchmarking trustworthy ML models for real-world applications because it has irregularities typically associated with IoT data-intensive applications, e.g., gaps, numerical errors, and outliers.

5.1 Future work

Some limitations motivate future work. For example, results of Chapter 3 only apply for SCNNs with a single output neuron and training complexity does not scale well with large datasets. As discussed in Section 3.9, there are many different research directions to increase training speed. For example, methods using GPU acceleration for convex NNs similarly to Mishkin et al. (2022) and Miria Feng and Pilanci (2024), or methods leveraging convergence properties like the alternating direction method of multipliers Bai et al. (2023b). We also note the similarity between our training formulation and the assumptions of the Representer's theorem, which is widely known to help cast high-dimensional, possibly infinite-dimensional, learning problems in more efficient ways (Schölkopf et al. 2001). In Chapter 4, because our propositions are mostly empirically motivated, the main limitation is the lack of proper theoretical performance guarantees. It would be interesting to study furthermore the theoretical implications of using our method to preprocess training datasets on the out-of-sample predictive performance of trained models. We believe the literature on probabilistic machine learning could be interesting as a starting point (Krause and Hübotter 2025). Ultimately, we believe that some interesting research tracks could stem from our work at the

frontier of trustworthy ML and safe control. We hint at these tracks in the following section.

5.2 Open perspectives in control

We finish with a short section on some open perspectives in safe nonlinear data-driven control. The related concepts are highlighted in Figure 5.1.

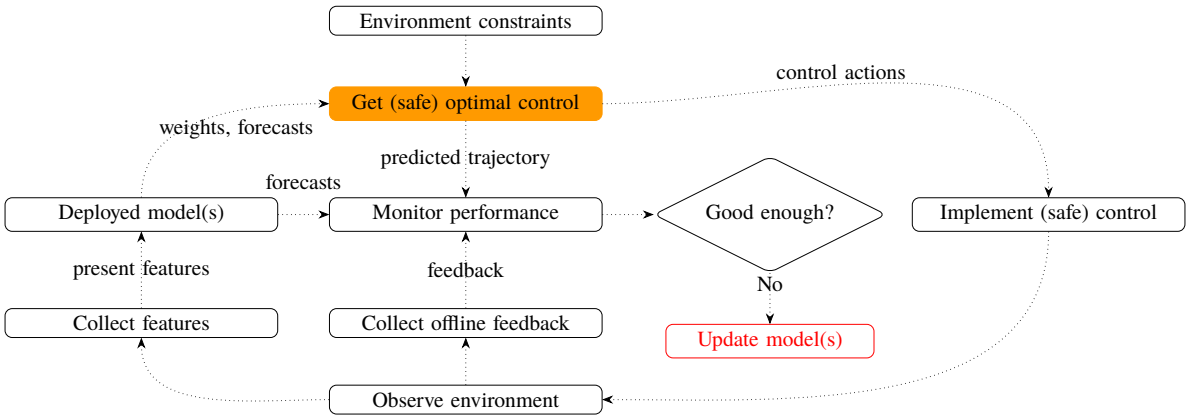


Figure 5.1 Concept from the trustworthy ML pipeline covered in Chapter 5.2

5.2.1 Contributions

To the best of our knowledge, we make the following contributions.

- We demonstrate that a mixed-integer convex program is obtainable for nonlinear data-driven model predictive control using SCNNs.
- We show how ϵ-stability and similar metrics obtained during post-training verification of SCNNs can be leveraged and refined to enhance constraint satisfaction in model predictive control.

5.2.2 Preliminaries on data-driven nonlinear model predictive control

Consider a discrete-time nonlinear dynamical system of the form:

$$\mathbf{x}_{k+1} = \Phi_{\mathbf{x}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \tag{5.1}$$

$$\mathbf{y}_k = \Phi_{\mathbf{y}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \tag{5.2}$$

$$\mathbf{p}_{k+1} = \Phi_{\mathbf{p}}(\mathbf{p}_k, \mathbf{c}_k) \tag{5.3}$$

with state $\mathbf{x}_k \in \mathfrak{X} \subseteq \mathbb{R}^d$, control action $\mathbf{u}_k \in \mathfrak{U} \subseteq \mathbb{R}^m$, output $\mathbf{y}_k \in \mathfrak{Y} \subseteq \mathbb{R}^n$, exogenous disturbance $\mathbf{p}_k \in \mathfrak{P} \subseteq \mathbb{R}^l$, and context $\mathbf{c}_k \in \mathbb{R}^p$, at round $k \in \{0, 1, \ldots, T-1\}$ in a finite horizon. Let $\Phi_{\mathbf{x}} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^d$, $\Phi_{\mathbf{y}} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^n$, and $\Phi_{\mathbf{p}} : \mathbb{R}^l \times \mathbb{R}^p \to \mathbb{R}^l$ be nonlinear mappings. Denote by $\tilde{\Phi}_{\mathbf{x}}$, $\tilde{\Phi}_{\mathbf{y}}$, and $\tilde{\Phi}_{\mathbf{p}}$ the approximation of mappings obtained by nonlinear learning models. Context $\mathbf{c}_k$ accounts for outside knowledge that impacts exogenous perturbations but does not directly influence state evolution or system outputs, such that it can be used to predict the perturbations.

The *vanilla* data-driven model predictive control (MPC) problem for the dynamical system (5.1)–(5.3) can be formulated as:

$$
\begin{aligned}
\min_{\{\mathbf{u}_k\}_{k=t}^{t+\tau}, \{\mathbf{x}_{k+1}\}_{k=t}^{t+\tau}, \{\mathbf{y}_k\}_{k=t}^{t+\tau}} \quad & \mathcal{J}\left(\{\mathbf{u}_k\}_{k=t}^{t+\tau}, \{\mathbf{x}_{k+1}\}_{k=t}^{t+\tau}, \{\mathbf{y}_k\}_{k=t}^{t+\tau}\right) && \text{(MPC)} \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = \tilde{\Phi}_{\mathbf{x}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) && \forall k \in \{t, t+1, \ldots, t+\tau\} \\
& \mathbf{y}_k = \tilde{\Phi}_{\mathbf{y}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) && \forall k \in \{t, t+1, \ldots, t+\tau\} \\
& \mathbf{p}_{k+1} = \tilde{\Phi}_{\mathbf{p}}(\mathbf{p}_k, \mathbf{c}_k) && \forall k \in \{t, t+1, \ldots, t+\tau\} \\
& \mathbf{u}_k \in \mathfrak{U}^{\text{safe}},\ \mathbf{x}_{k+1} \in \mathfrak{X}^{\text{safe}},\ \mathbf{y}_k \in \mathfrak{Y}^{\text{safe}} && \forall k \in \{t, t+1, \ldots, t+\tau\},
\end{aligned}
$$

where $\mathcal{J} : \mathbb{R}^{m\times\tau} \times \mathbb{R}^{d\times\tau} \times \mathbb{R}^{n\times\tau} \to \mathbb{R}$ is a loss function, $\mathfrak{U}^{\text{safe}} \subset \mathfrak{U}$, $\mathfrak{X}^{\text{safe}} \subset \mathfrak{X}$, and $\mathfrak{Y}^{\text{safe}} \subset \mathfrak{Y}$ are safety sets with respect to the context, t is the initial round of the decision-making such that $0 \leq t < \tau \leq T$. The safety sets refer to the intersection of constraints dictating a safe control of the system, e.g., operational constraints, physical constraints, and stochastic constraints. The problem is solved in a rolling horizon where only the first $0 < \Delta \leq \tau$ control actions are implemented before re-optimizing from the next observable state onwards.

Solving (MPC) to optimality allows us to obtain control actions that minimize $\mathcal{J}$ with respect to the learned models. As such, models that capture the dynamics well will lead to good control actions. We remark that preserving convexity in the objective and the constraints is beneficial to guarantee global optimality.

5.2.3 An SCNN-based nonlinear MPC

We now present our main proposition from this section which is analogous to Proposition 5.

Proposition 7. *Suppose that $\mathcal{U}^{safe}$, $\mathfrak{X}^{safe}$, and $\mathfrak{Y}^{safe}$ are convex sets and that $\mathfrak{J}$ is jointly convex with respect to the variables of the optimization problem, then, (MPC) can be cast as a mixed-integer convex program using shallow convex neural networks as the learning models.*

Proof: The ReLU-like MIP mechanism, based on the set of sampling vectors $\mathcal{S}$, from Proposition 5

is directly adaptable using stacks of SCNNs. For example, consider the state predictor $\tilde{\Phi}_{\mathbf{x}} : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}^d$, stacking d SCNNs together, we form:

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_{1,k+1} \\ \vdots \\ x_{d,k+1} \end{bmatrix} = \tilde{\Phi}_{\mathbf{x}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) = \begin{bmatrix} \tilde{\Phi}_{\mathbf{x}_1}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \\ \vdots \\ \tilde{\Phi}_{\mathbf{x}_d}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \end{bmatrix} \quad \forall k \in \{t, t+1, \dots, t+\tau\}, \tag{5.4}$$

where each $\tilde{\Phi}_{\mathbf{x}_i}, i \in [\![d]\!]$ is a trained SCNN. Then, for each element of the stack, a mixed-integer convex formulation is obtainable for each time-step of (MPC) following the mechanisms introduced in the proof of Proposition 5. □

5.2.4 Enforcing tube constraints from ϵ-stability

In Section 3.6, we have shown in Proposition 5 that the worst-case output deviation of a SCNN for any input in the feature space subject to a bounded perturbation ϵ can be certified by a mixed-integer convex program. We have used the term ϵ-stability, denoted L_ϵ, to characterize this deviation.

We remark that L_ϵ, in the case where it is *small* with respect to the control application as to preserve feasibility, can be leveraged to improve constraint satisfaction when employing data-driven system models.

For example, consider that system outputs are subject to box constraints, leading to:

$$\mathbf{y}_{\min} \preceq \tilde{\Phi}_{\mathbf{y}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \preceq \mathbf{y}_{\max} \quad \forall k \in \{t, t+1, \dots, t+\tau\}, \tag{5.5}$$

where $\preceq$ is an element-wise $\leq$ operator, $\mathbf{y}_{\min}, \mathbf{y}_{\max} \in \mathbb{R}^n$ such that $\mathbf{y}_{\min} \preceq \mathbf{y}_{\max}$ and where we know L_ϵ from PTV. Then, we can modify constraint conservatism to ensure constraint satisfaction during control by accounting for this worst possible deviation. Recall that L_ϵ is the maximal deviation in the input space for a small input perturbation $\boldsymbol{\epsilon} \in \mathcal{E}$, not to be confused with $\mathbf{p}_k$. We now formulate a variation of (5.5) using L_ϵ for $n = 1$:

$$y_{\min} + L_\epsilon \leq \tilde{\Phi}_{\mathbf{y}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k) \leq y_{\max} - L_\epsilon \quad \forall k \in \{t, t+1, \dots, t+\tau\}. \tag{5.6}$$

We remark that L_ϵ depends on the definition of the input space support, denoted $\mathcal{X}$ in (3.32). We highlight this dependence with the notation $L_\epsilon(\mathcal{X})$. Assuming real-time state and perturbation knowledge during control, L_ϵ can be refined around local *neighbourhoods* to scale conservatism. To avoid notation mixup, denote by $(\mathfrak{X}, \mathfrak{U}, \mathfrak{P})$ the whole input space of the learning models in (MPC) and by $\mathcal{N}_x(\mathbf{x}_k) \subset \mathfrak{X},$ and $\mathcal{N}_p(\mathbf{p}_k) \subset \mathfrak{P}$ convex neighbourhoods around state at time k and exogenous perturbations. We argue that in parallel to the real-time control, a local refinement of L_ϵ can be

obtained with $(\mathcal{N}_x(\mathbf{x}_k), \mathfrak{U}, \mathcal{N}(\mathbf{p}_k)_p) \subset (\mathfrak{X}, \mathfrak{U}, \mathfrak{P})$ such that $L_\epsilon\left(\mathcal{N}_x(\mathbf{x}_k), \mathfrak{U}, \mathcal{N}_p(\mathbf{p}_k)\right) \leq L_\epsilon\left(\mathfrak{X}, \mathfrak{U}, \mathfrak{P}\right)$. In other words, through control, the value of L_ϵ can be tuned according to the stability of the learning model around its present state.

We refer readers to the literature on tube MPC for further reading on these ideas, e.g., see the work of Lopez et al. (2019).

5.2.5 Closing remarks

In this chapter, we have hinted at how the post-deployment section of the trustworthy ML pipeline can benefit from contributions in the pre-deployment section. Understanding and studying ML models theoretically inevitably leads to better real-world usage.

We showed that the PTV mechanisms developed for SCNNs could be adapted for nonlinear model predictive control. We have also demonstrated that PTV metrics, like L_ϵ, can be leveraged and refined to enhance control reliability.

By doing so, we hope to motivate future works where the trustworthy ML pipeline is considered from deployment to exploitation.

REFERENCES

Ackley D (2012) *A connectionist machine for genetic hillclimbing*, volume 28 (Springer science & business media).

Adorio EP (2005) MVF - Multivariate Test Functions Library in C for Unconstrained Global Optimization. Technical report, Department of Mathematics, U.P. Diliman.

Akbilgic O (2013) ISTANBUL STOCK EXCHANGE. UCI Machine Learning Repository.

Albadi MH, El-Saadany EF (2007) Demand Response in Electricity Markets: An Overview. *2007 IEEE Power Engineering Society General Meeting*, 1–5.

Albarghouthi A (2021) Introduction to neural network verification. *Foundations and Trends® in Programming Languages* 7(1–2):1–157.

Amari Si (1993) Backpropagation and stochastic gradient descent method. *Neurocomputing* 5(4-5):185–196.

Amasyali K, El-Gohary NM (2018) A review of data-driven building energy consumption prediction studies. *Renewable and Sustainable Energy Reviews* 81:1192–1205.

Amer M, Goldstein M, Abdennadher S (2013) Enhancing one-class support vector machines for unsupervised anomaly detection. *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description* 8–15.

Angiulli F, Pizzuti C (2002) Fast Outlier Detection in High Dimensional Spaces. *Proceedings of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery* 2431:15–26.

Audet C, Le Digabel S, Montplaisir VR, Tribes C (2022) Algorithm 1027: NOMAD Version 4: Nonlinear Optimization with the MADS Algorithm. *ACM Trans. Math. Softw.* 48(3).

Bai X, He G, Jiang Y, Obloj J (2023a) Wasserstein distributional robustness of neural networks. Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, eds., *Advances in Neural Information Processing Systems*, volume 36, 26322–26347.

Bai Y, Gautam T, Sojoudi S (2023b) Efficient global optimization of two-layer relu networks: Quadratic-time algorithms and adversarial training. *SIAM Journal on Mathematics of Data Science* 5(2):446–474.

Baronti L, Castellani M (2024) A Python Benchmark Functions Framework for Numerical Optimisation Problems. Technical report, School of Computer Science, University of Birmingham.

Bartlett PL, Mendelson S (2002) Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3(Nov):463–482.

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2017) Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research* 18(1):5595–5637.

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. volume 24, 1–9.

Bergstra J, Komer B, Eliasmith C, Yamins D, Cox DD (2015) Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery* 8(1):014008.

Bishop CM, Bishop H (2023) *Deep learning: Foundations and concepts* (Springer Nature).

Bishop CM, Nasrabadi NM (2006) *Pattern recognition and machine learning* (Springer).

Bonneel N, Rabin J, Peyré G, Pfister H (2015) Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision* 51:22–45.

Bonnotte N (2013) *Unidimensional and evolution methods for optimal transportation*. Ph.D. thesis, Université Paris Sud-Paris XI; Scuola normale superiore (Pise, Italie).

Boyd S, Park J (2014) Subgradient methods. *Notes for EE364b, Stanford University, Spring 2014* .

Boyd S, Vandenberghe L (2004) *Convex optimization* (Cambridge University Press).

Brashaw G (1989) Solar Flare. UCI Machine Learning Repository.

Breunig MM, Kriegel HP, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* 93–104.

Campos GO, Zimek A, Sander J, Campello RJ, Micenková B, Schubert E, Assent I, Houle ME (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery* 30:891–927.

Carlen EA, Frank RL, Ivanisvili P, Lieb EH (2021) Inequalities for lp-norms that sharpen the triangle inequality and complement hanner's inequality. *The Journal of geometric analysis* 31:4051–4073.

Chatzivasileiadis S, Venzke A, Stiasny J, Misyris G (2022) Machine Learning in Power Systems: Is It Time to Trust It? *IEEE Power and Energy Magazine* 20(3):32–41.

Chen R, Paschalidis IC (2018) A robust learning approach for regression models based on distributionally robust optimization. *Journal of Machine Learning Research* 19(13):1–48.

Chen R, Paschalidis IC (2020) Distributionally robust learning. *Foundations and Trends® in Optimization* 4(1-2):1–243.

Chen T, Guestrin C (2016) XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794, KDD '16 (New York, NY, USA: Association for Computing Machinery).

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Wine Quality. UCI Machine Learning Repository.

Cortez P, Morais A (2007) Forest Fires. UCI Machine Learning Repository.

Cuomo S, Di Cola VS, Giampaolo F, Rozza G, Raissi M, Piccialli F (2022) Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing* 92(3):88.

Dempe S (2002) *Foundations of bilevel programming* (Springer Science & Business Media).

Diamond S, Boyd S (2016) CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17(83):1–5.

Dong B, Cao C, Lee SE (2005) Applying support vector machines to predict building energy consumption in tropical region. *Energy and Buildings* 37(5):545–553.

Ducoffe M, Haloui I, Gupta JS (2019) Anomaly detection on time series with Wasserstein GAN applied to PHM. *International Journal of Prognostics and Health Management* 10(4).

Gao H, Sun L, Wang JX (2021) PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics* 428:110079.

Garcia CE, Morari M (1982) Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development* 21(2):308–323.

Garcia CE, Prett DM, Morari M (1989) Model predictive control: Theory and practice—a survey. *Automatica* 25(3):335–348.

Gerbet T (2023) Performance énergétique des bâtiments : Québec manque de courage, déplorent des experts. *Radio-Canada* .

Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter MA, Kagal L (2018) Explaining explanations: An approach to evaluating interpretability of machine learning. *CoRR* abs/1806.00069.

Giudici P, Raffinetti E (2023) SAFE Artificial Intelligence in finance. *Finance Research Letters* 56:104088.

Goldstein M, Uchida S (2016) A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLOS ONE* 11(4):1–31.

Goulart PJ, Chen Y (2024) Clarabel: An interior-point solver for conic programs with quadratic objectives. Technical report, Department of Engineering Science, University of Oxford.

Herter K (2007) Residential implementation of critical-peak pricing of electricity. *Energy Policy* 35(4):2121–2130.

Householder AS (1941) A theory of steady-state activity in nerve-fiber networks: I. definitions and preliminary lemmas. *The bulletin of mathematical biophysics* 3:63–69.

Huang X, Kwiatkowska M, Wang S, Wu M (2017) Safety verification of deep neural networks. *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, 3–29 (Springer).

Huang Y, Zhang H, Shi Y, Kolter JZ, Anandkumar A (2021) Training Certifiably Robust Neural Networks with Efficient Local Lipschitz Bounds. Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, eds., *Advances in Neural Information Processing Systems*, volume 34, 22745–22757.

Hydro-Québec (2023) Une énergie renouvelable pour un québec durable, rapport sur le développement durable 41–43.

Ingber L (1993) Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling* 18(11):29–57.

Jospin LV, Laga H, Boussaid F, Buntine W, Bennamoun M (2022) Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine* 17(2):29–48.

Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L (2021) Physics-informed machine learning. *Nature Reviews Physics* 3(6):422–440.

Kawaguchi K, Sun Q (2021) A recipe for global convergence guarantee in deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8074–8082.

Keane A (1994) Experiences with optimizers in structural design. *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, volume 94, 14–27.

Kelly M, Longjohn R, Nottingham K (2023) UCI Machine Learning Repository. URL `http://archive.ics.uci.edu/ml`.

Kleppmann M (2017) *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems* (O'Reilly Media, Inc.).

Kolouri S, Nadjahi K, Simsekli U, Badeau R, Rohde G (2019) Generalized Sliced Wasserstein Distances. Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, eds., *Advances in Neural Information Processing Systems*, volume 32, 1–12.

Krause A, Hübotter J (2025) Probabilistic artificial intelligence. *arXiv preprint arXiv:2502.05244* .

Kreuzberger D, Kühl N, Hirschl S (2023) Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access* 11:31866–31879.

Kuelbs D, Lall S, Pilanci M (2024) Adversarial Training of Two-Layer Polynomial and ReLU Activation Networks via Convex Optimization. *arXiv preprint arXiv:2405.14033* .

Kuhn D, Esfahani PM, Nguyen VA, Shafieezadeh-Abadeh S (2019) Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning. *Operations Research & Management Science in the Age of Analytics*, 130–166 (INFORMS).

Levy D, Carmon Y, Duchi JC, Sidford A (2020) Large-Scale Methods for Distributionally Robust Optimization. *Advances in Neural Information Processing Systems*.

Li F, Kocar I, Lesage-Landry A (2024) A Rapid Method for Impact Analysis of Grid-Edge Technologies on Power Distribution Networks. *IEEE Transactions on Power Systems* 39(1):1530–1542.

Lijesen MG (2007) The real-time price elasticity of electricity. *Energy Economics* 29(2):249–258.

Liu FT, Ting KM, Zhou ZH (2008) Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422.

Liu J, Shen Z, Cui P, Zhou L, Kuang K, Li B (2022) Distributionally robust learning with stable adversarial training. *IEEE Transactions on Knowledge and Data Engineering* 35(11):11288–11300.

Lopez BT, Slotine JJE, How JP (2019) Dynamic tube MPC for nonlinear systems. *2019 American Control Conference (ACC)*, 1655–1662 (IEEE).

Lu L, Pestourie R, Yao W, Wang Z, Verdugo F, Johnson SG (2021) Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing* 43(6):B1105–B1132.

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777, NIPS'17 (Red Hook, NY, USA: Curran Associates Inc.).

Massana J, Pous C, Burgas L, Melendez J, Colomer J (2015) Short-term load forecasting in a non-residential building contrasting models and attributes. *Energy and Buildings* 92:322–330.

Mercado A, Mitchell R, Earni S, Diamond R, Alschuler E (2014) Enabling interoperability through a common language for building performance data. *Proceedings of the 2014 ACEEE Summer Study on Energy Efficiency in Buildings* .

Miria Feng ZF, Pilanci M (2024) CRONOS: Convex Neural Networks via Operator Splitting. *Proceedings of the Neural Information Processing Systems (NeurIPS)*, volume 37, 1–25.

Mishkin A, Sahiner A, Pilanci M (2022) Fast Convex Optimization for Two-Layer ReLU Networks: Equivalent Model Classes and Cone Decompositions. *International Conference on Machine Learning*, 15770–15816 (PMLR).

Misyris GS, Venzke A, Chatzivasileiadis S (2020) Physics-informed neural networks for power systems. *2020 IEEE Power & Energy Society General Meeting (PESGM)*, 1–5.

Mohajerin Esfahani P, Kuhn D (2018) Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming* 171(1):115–166.

Moon J, Park J, Hwang E, Jun S (2018) Forecasting power consumption for higher educational institutions based on machine learning. *The Journal of Supercomputing* 74:3778–3800.

Newsham GR, Birt BJ (2010) Building-level occupancy data to improve ARIMA-based electricity use forecasts. *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, 13–18, BuildSys '10 (New York, NY, USA: Association for Computing Machinery).

Pallage J (2024) Advances in the Trustworthy Machine Learning Pipeline, Seminars at the Automatic Control Lab, École Polytechnique Fédérale de Lausanne, Online.

Pallage J (2025) The Trustworthy Machine Learning Pipeline for Virtual Power Plants, 5th GERAD's Student Day, Montréal, Québec, Canada.

Pallage J, Lesage-Landry A (2024a) (Trustworthy) AI for Québec's Virtual Power Plant. *Cahiers du GERAD,* IVADO Digital Futures 2024, Montréal, Québec, Canada.

Pallage J, Lesage-Landry A (2024b) Wasserstein Distributionally Robust Shallow Convex Neural Networks. *arXiv preprint arXiv:2407.16800* .

Pallage J, Lesage-Landry A (2024c) Wasserstein Distributionally Robust Shallow Convex Neural Networks, 25th International Symposium on Mathematical Programming, Montréal, Québec, Canada.

Pallage J, Lesage-Landry A (2024d) Wasserstein Distributionally Robust Shallow Convex Neural Networks, Optimization Days, Montréal, Québec, Canada.

Pallage J, Lesage-Landry A (2025) Sliced-wasserstein distance-based data selection. *arXiv preprint arXiv:2504.12918* .

Pallage J, Scherrer B, Naccache S, Bélanger C, Lesage-Landry A (2024) Sliced-Wasserstein-based Anomaly Detection and Open Dataset for Localized Critical Peak Rebates. *NeurIPS 2024 Workshop on Tackling Climate Change with Machine Learning*.

Panaretos VM, Zemel Y (2019) Statistical aspects of Wasserstein distances. *Annual Review of Statistics and its Application* 6(1):405–431.

Panaretos VM, Zemel Y (2020) *An invitation to statistics in Wasserstein space* (Springer Nature).

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, volume 32, 8024–8035.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pelletier F, Faruqui A (2022) Does dynamic pricing work in a winter-peaking climate? A case study of Hydro Quebec. *The Electricity Journal* 35(2):107080, ISSN 1040-6190.

Picheny V, Wagner T, Ginsbourger D (2013) A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization* 48:607–626.

Pilanci M (2022) The hidden convex optimization landscape of deep neural networks.

Pilanci M, Ergen T (2020) Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. *International Conference on Machine Learning*, 7695–7705 (PMLR).

Pudjianto D, Ramsay C, Strbac G (2007) Virtual power plant and system integration of distributed energy resources. *IET Renewable Power Generation* 1(1):10–16.

Qi M, Cao Y, Shen ZJ (2022) Distributionally robust conditional quantile prediction with fixed design. *Management Science* 68(3):1639–1658.

Rasheed K, Qayyum A, Ghaly M, Al-Fuqaha A, Razi A, Qadir J (2022) Explainable, trustworthy, and ethical machine learning for healthcare: A survey. *Computers in Biology and Medicine* 149:106043.

Reis J, Housley M (2022) *Fundamentals of data engineering* (O'Reilly Media, Inc.).

Ruan G, Qiu D, Sivaranjani S, Awad AS, Strbac G (2024) Data-driven energy management of virtual power plants: A review. *Advances in Applied Energy* 100170.

Sagawa S, Koh PW, Hashimoto TB, Liang P (2020) Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. *International Conference on Learning Representations*, 1–19.

Sarker IH (2021) Machine learning: Algorithms, real-world applications and research directions. *SN computer science* 2(3):160.

Schölkopf B, Herbrich R, Smola AJ (2001) A generalized representer theorem. *International Conference on Computational Learning Theory*, 416–426 (Springer).

Shafieezadeh-Abadeh S, Kuhn D, Esfahani PM (2019) Regularization via mass transportation. *Journal of Machine Learning Research* 20(103):1–68.

Siano P (2014) Demand response and smart grids—a survey. *Renewable and sustainable energy reviews* 30:461–478.

Sohl-Dickstein J (2024) The boundary of neural network trainability is fractal. *arXiv preprint arXiv:2402.06184* .

Spearman C (1904) The proof and measurement of association between two things. *American Journal of Psychology* 15:88–103.

Stiasny J, Chevalier S, Nellikkath R, Sævarsson B, Chatzivasileiadis S (2022) Closing the loop: A framework for trustworthy machine learning in power systems. *Proceedings of 11th Bulk Power Systems Dynamics and Control Symposium (IREP 2022)* 1–21.

Tang J, Chen Z, Fu AWC, Cheung DWL (2002) Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 535–548, PAKDD '02 (Berlin, Heidelberg: Springer-Verlag).

Tsanas A, Xifara A (2012) Energy Efficiency. UCI Machine Learning Repository.

van der Vaart AW (2000) *Asymptotic statistics*. Cambridge Series in Statistical and Probabilistic Mathematics (Cambridge University Press).

Varshney KR (2022) *Trustworthy Machine Learning* (Chappaqua, NY, USA: Independently Published).

Venzke A, Chatzivasileiadis S (2021) Verification of neural network behaviour: Formal guarantees for power system applications. *IEEE Transactions on Smart Grid* 12(1):383–397.

Venzke A, Qu G, Low S, Chatzivasileiadis S (2020) Learning optimal power flow: Worst-case guarantees for neural networks. *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, volume 11, 1–7.

Villani C (2008) *Optimal transport: old and new*, volume 338 (Springer).

Wang Y, Lacotte J, Pilanci M (2021) The hidden convex optimization landscape of regularized two-layer ReLU networks: an exact characterization of optimal solutions. *International Conference on Learning Representations*, 1–26.

Wang Y, Sun W, Jin J, Kong Z, Yue X (2024) WOOD: Wasserstein-Based Out-of-Distribution Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46(2):944–956.

Weng Y, Rajagopal R (2015) Probabilistic baseline estimation via Gaussian process. *2015 IEEE Power & Energy Society General Meeting*, 1–5.

Whitley D, Rana S, Dzubera J, Mathias KE (1996) Evaluating evolutionary algorithms. *Artificial Intelligence* 85(1-2):245–276.

Whitmore J, Pineau PO (2024) État de l'énergie au québec. Préparé pour le gouvernement du Québec.

Wiggins S (2003) *Introduction To Applied Nonlinear Dynamical Systems And Chaos*, volume 4.

Williams C, Rasmussen C (1995) Gaussian processes for regression. *Advances in Neural Information Processing Systems* 8.

Wilson AG, Izmailov P (2020) Bayesian deep learning and a probabilistic perspective of generalization. *Advances in Neural Information Processing Systems* 33:4697–4708.

Xu Y, Kohtz S, Boakye J, Gardoni P, Wang P (2023) Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges. *Reliability Engineering & System Safety* 230:108900.

Yang L, Meng X, Karniadakis GE (2021) B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics* 425:109913.

Yang L, Zhang D, Karniadakis GE (2020) Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations. *SIAM Journal on Scientific Computing* 42(1):A292–A317.

Yang P, Tang G, Nehorai A (2013) A game-theoretic approach for optimal time-of-use electricity pricing. *IEEE Transactions on Power Systems* 28(2):884–892.

Yeh IC (1998) Concrete Compressive Strength. UCI Machine Learning Repository.

Yue MC, Kuhn D, Wiesemann W (2022) On linear optimization over Wasserstein balls. *Mathematical Programming* 195(1):1107–1122.

Zhang G, Jiang C, Wang X (2019) Comprehensive review on structure and operation of virtual power plant in electrical system. *IET Generation, Transmission & Distribution* 13(2):145–156.

Zhang L, Jeong D, Lee S (2021) Data Quality Management in the Internet of Things. *Sensors* 21(17).

Zhang Q, Li J (2012) Demand response in electricity markets: A review. *2012 9th International Conference on the European Energy Market*, 1–8.

Zhang R, Liu Y, Sun H (2020) Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering* 369:113226.

Zhou D, Brix C, Hanasusanto GA, Zhang H (2024) Scalable Neural Network Verification with Branch-and-bound Inferred Cutting Planes. *Proceedings of the Neural Information Processing Systems (NeurIPS)*, volume 37.

Zhu Q, Liu Z, Yan J (2021) Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics* 67:619–635.

APPENDIX A SUPPLEMENTARY CONTENT OF CHAPTER 3

A.1 Proof of Proposition 1

Our objective is to rewrite (WDR-SCNNT) in the form of (TWDRO). This proof is derived from results of Chen and Paschalidis (2020) and is adapted here for completeness. Consider the ℓ_1-norm loss function. We have $h_{\boldsymbol{\beta}}(\hat{\mathbf{z}}) = |\hat{\mathbf{z}}^\top \boldsymbol{\beta}|$ and

$$h^*_{\boldsymbol{\beta}}(\boldsymbol{\theta}) < +\infty \iff \sup_{\hat{\mathbf{z}}:\hat{\mathbf{z}}^\top\boldsymbol{\beta}\geq 0} \{\boldsymbol{\theta}^\top\hat{\mathbf{z}} - \boldsymbol{\beta}^\top\hat{\mathbf{z}}\} < +\infty \text{ and } \sup_{\hat{\mathbf{z}}:\hat{\mathbf{z}}^\top\boldsymbol{\beta}\leq 0} \{\boldsymbol{\theta}^\top\hat{\mathbf{z}} + \boldsymbol{\beta}^\top\hat{\mathbf{z}}\} < +\infty.$$

Considering the two linear optimization problems, we can find the equivalent dual problems A and B with dual variables λ_A and λ_B, respectively:

$$\begin{aligned} &\max\ (\boldsymbol{\theta}-\boldsymbol{\beta})^\top\hat{\mathbf{z}} \xrightarrow{\text{dual}} \min\ 0\cdot\lambda_A \\ &\text{s.t. } \hat{\mathbf{z}}^\top\boldsymbol{\beta}\geq 0 \qquad\qquad \text{s.t. } \boldsymbol{\beta}\lambda_A = \boldsymbol{\theta}-\boldsymbol{\beta},\ \lambda_A\leq 0, \end{aligned} \tag{Dual-A}$$

and,

$$\begin{aligned} &\max(\boldsymbol{\theta}+\boldsymbol{\beta})^\top\hat{\mathbf{z}} \xrightarrow{\text{dual}} \min\ 0\cdot\lambda_B \\ &\text{s.t. } \hat{\mathbf{z}}^\top\boldsymbol{\beta}\leq 0 \qquad\qquad \text{s.t. } \boldsymbol{\beta}\lambda_B = \boldsymbol{\theta}+\boldsymbol{\beta},\ \lambda_B\geq 0. \end{aligned} \tag{Dual-B}$$

Because the objective function of both dual problems is 0, to have finite optimal values for both primals, (Dual-A) and (Dual-B) need to have a non-empty feasible set. This implies that these two constraints must be respected:

$$\begin{aligned} &\exists\lambda_A\leq 0, \quad \text{s.t.} \quad \boldsymbol{\beta}\lambda_A = \boldsymbol{\theta}-\boldsymbol{\beta} \\ &\exists\lambda_B\geq 0, \quad \text{s.t.} \quad \boldsymbol{\beta}\lambda_B = \boldsymbol{\theta}+\boldsymbol{\beta}. \end{aligned}$$

From this we find that,

$$|\theta_i| \leq |\beta_i|\ \forall i, \tag{A.1}$$

because

$$\begin{aligned} &\boldsymbol{\beta}(1-\lambda_B) = \boldsymbol{\theta} \text{ where } \lambda_B\geq 0 \\ &\boldsymbol{\beta}(1+\lambda_A) = \boldsymbol{\theta} \text{ where } \lambda_A\leq 0. \end{aligned}$$

If (A.1) holds, then both primals are finite. Thus, we have

$$\kappa(\boldsymbol{\beta}) = \sup\{\|\boldsymbol{\theta}\|_* : |\theta_i| \leq \beta_i, \forall i\},$$

and,

$$\kappa(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_* = \sup_{\|\mathbf{k}\| \leq 1} \boldsymbol{\beta}^\top \mathbf{k}.$$

For the ℓ_1-loss, $\|\boldsymbol{\beta}\|_* = \|\boldsymbol{\beta}\|_\infty$, and the problem becomes:

$$\inf_{\boldsymbol{\beta}} \epsilon\|\boldsymbol{\beta}\|_\infty + \frac{1}{N}\sum_{i=1}^{N} h_{\boldsymbol{\beta}}(\hat{\mathbf{z}}_i). \tag{A.2}$$

Using the slack variables $a \in \mathbb{R}$ and $\mathbf{c} \in \mathbb{R}^N$ in, respectively, the first and second terms of the objective, yields the equivalent problem:

$$\begin{aligned}
\min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & \epsilon a & & + \frac{1}{N}\sum_{j=1}^{N} c_j \\
\text{s.t.} \quad & \boldsymbol{\beta}_i & & \leq a \quad \forall i \in [\![2Pd+1]\!] \\
& -\boldsymbol{\beta}_i & & \leq a \quad \forall i \in [\![2Pd+1]\!] \\
& \boldsymbol{\beta}^\top \hat{\mathbf{z}}_j & & \leq c_j \quad \forall j \in [\![N]\!] \\
& -(\boldsymbol{\beta}^\top \hat{\mathbf{z}}_j) & & \leq c_j \quad \forall j \in [\![N]\!] \\
& \boldsymbol{\beta} & & = (\mathrm{vec}(\boldsymbol{\nu}), \mathrm{vec}(-\boldsymbol{\omega}), -1) \\
& \boldsymbol{\nu}_i, \boldsymbol{\omega}_i & & \in \mathcal{K}_i \quad \forall i \in [\![P]\!].
\end{aligned}$$

For the ℓ_2-loss, $\|\boldsymbol{\beta}\|_*$ is equal to the ℓ_2-norm (Boyd and Vandenberghe 2004). By the same process and noting that $a \geq 0$ because a norm is non-negative, we obtain the second equivalent problem:

$$\begin{aligned}
\min_{\boldsymbol{\nu},\boldsymbol{\omega}} \quad & \epsilon a & & + \frac{1}{N}\sum_{j=1}^{N} c_j \\
\text{s.t.} \quad & \|\boldsymbol{\beta}\|_2^2 & & \leq a^2 \\
& a & & \geq 0 \\
& \boldsymbol{\beta}^\top \hat{\mathbf{z}}_j & & \leq c_j \quad \forall j \in [\![N]\!] \\
& -(\boldsymbol{\beta}^\top \hat{\mathbf{z}}_j) & & \leq c_j \quad \forall j \in [\![N]\!] \\
& \boldsymbol{\beta} & & = (\mathrm{vec}(\boldsymbol{\nu}), \mathrm{vec}(-\boldsymbol{\omega}), -1) \\
& \boldsymbol{\nu}_i, \boldsymbol{\omega}_i & & \in \mathcal{K}_i \quad \forall i \in [\![P]\!],
\end{aligned}$$

which completes the proof. □

A.2 Complementary materials for the synthetic experiment

In this section, we detail the synthetic experiments of Section 3.8.1 and we present additional figures.

A.2.1 Hyperparameters and setting

The hyperparameters of each model are presented in Table A.1. Note that all models are formulated with bias weights by default and that different regularization methodologies are set as hyperparameters in this experiment.

We sample 2000 data points for each benchmark function before splitting them between training, validation, and testing. For the benchmark functions that can be defined in dimension higher than two, Ackley and Keane, we impose a dimension of four.

A.2.2 Additionnal figures

Figure A.1 presents the absolute values of the errors presented in Section 3.8.1. Label values for each function on the feature domain are needed to better understand the figure. We proceed to identify the max and min sampled values. For McCormick, the minimal value is -1.91 and the maximal value is 65.32. For PGandP, the minimal value is -3.12 and the maximal value is 6.78. For Keane, the minimal value is -9.13 and the maximal value is 0. For Ackley, the minimal value is 0 and the maximal value is 22.29.

Compared to the normalized results, McCormick is extremely polarized: some models have low nominal test errors while others have bigger ones. While the WaDiRo-SCNN performs well in Experiments A and B, it sees a medium performance in Experiment C. This is hidden by the extremely poor performance of the deep FNN in Experiment C. Figure 3.3 shows that McCormick has linear tendencies in the center of its domain but steep increases on its edges that linear models fail to capture.

Keane, with its small codomain, has a small error range. We observe that the SCNN with no regularization is the only one to fail in Experiment A, while both standard SCNN formulations fail comparatively to the other models in Experiment C.

One interesting takeaway from Figure A.1 is that WaDiRo-SCNN, at its worst performance, is on par with linear regression models. We do see the standard SCNN formulations have a poorer performance than the linear models on Keane and the deep FNN has consistently worse performances in the corrupted setting from Experiment C.

Table A.1 Hyperparameters of the synthetic experiment

Model	Hyperparameter	Description	Possible values
WaDiRo-SCNN	`radius`	Radius of the Wasserstein ball	$[\mathrm{e}^{-8}, \mathrm{e}^{1}]$
	`max_neurons`	Max number of neurons	$\{10, 11, \ldots, 300\}$
	`wasserstein`	Norm of the Wasserstein metric	ℓ_1 or ℓ_2
Regularized SCNN	`lambda_reg`	Regularizing parameter	$[\mathrm{e}^{-8}, \mathrm{e}^{1}]$
	`max_neurons`	Max number of neurons	$\{10, 11, \ldots, 300\}$
	`regularizer`	Regularization framework	Lasso or Ridge
Non-regularized SCNN	`max_neurons`	Max number of neurons	$\{10, 11, \ldots, 300\}$
WaDiRo lin. reg.	`radius`	Radius of the Wasserstein ball	$[\mathrm{e}^{-8}, \mathrm{e}^{1}]$
	`wasserstein`	Norm of the Wasserstein metric	ℓ_1 or ℓ_2
Regularized lin. reg.	`lambda_reg`	Regularizing parameter	$[\mathrm{e}^{-8}, \mathrm{e}^{1}]$
	`regularizer`	Regularization framework	Lasso or Ridge
Deep FNN	`batch_size`	Size of each batch	$\{2, 3, \ldots, 100\}$
	`n_hidden`	Number of neurons per layer	$\{10, 11, \ldots, 300\}$
	`learning_rate`	Learning rate	$[\mathrm{e}^{-8}, \mathrm{e}^{1}]$
	`n_epochs`	Number of epochs	$\{2, 3, \ldots, 10^3\}$
	`dropout_p`	Dropout probability at each layer	$[0.01, 0.4]$

A.2.3 Computational time

The tests presented in Figure A.2 were conducted on a computer with 64GB of RAM DDR5 and an AMD Ryzen 9 7950x3d CPU. The training was performed with the open-source solver Clarabel and was formulated in Python with `cvxpy`. As we can see, more neurons lead to smaller training errors while a greater dataset size increases the training error. Indeed, sampling more data points increases the difficulty of fitting them all correctly. As for the training time, to keep the computational time low, there is a tradeoff between the maximal number of neurons and the size of the dataset. This is expected partly because we do not yet have a GPU acceleration method for this training procedure. We also remark that a more direct implementation, e.g., by using a lower-level programming language and avoiding an interface like `cvxpy` for the constraint implementation, could also diminish the training time. As such, comparing these running times with methods implemented in C++ would be unfair.

A.3 Complementary materials for the building experiment

In this section, we detail the experiments on non-residential buildings' baseline prediction from Section 3.8.2 and we showcase some additional figures of interest.

A.3.1 Hyperparameters and setting

First, we define the kernel used for the Gaussian process regression. Following our inspection of the data, we construct the GP kernel by adding a radial basis function (RBF) kernel to an exponential sine kernel to model periodicity and a white noise kernel to capture the possible noise.

Similarly to the previous section, we present, in Table A.2, the hyperparameters used in the experiments.

Table A.2 Hyperparameters of the building experiment

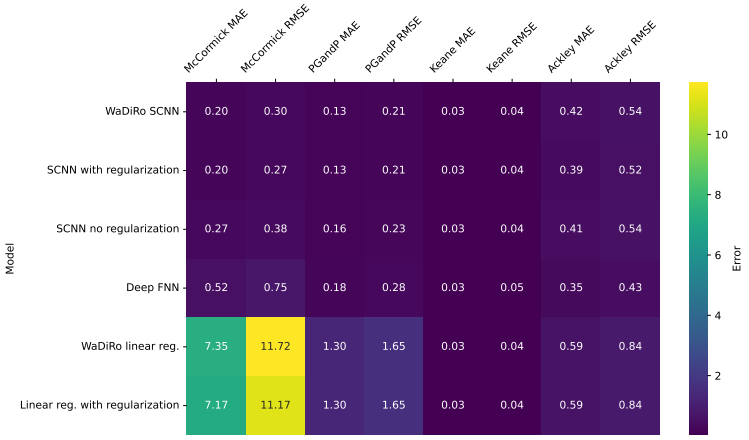
Model	**Hyperparameter**	**Description**	**Possible values**
WaDiRo-SCNN	`radius`	Radius of the Wasserstein ball	$[\mathrm{e}^{-6}, \mathrm{e}^{1}]$
	`max_neurons`	Max number of neurons	$\{10, 11, \ldots, 70\}$
	`wasserstein`	orm used in Wasserstein metric	ℓ_1 or ℓ_2
WaDiRo lin. reg.	`radius`	Radius of the Wasserstein ball	$[\mathrm{e}^{-6}, \mathrm{e}^{1}]$
	`wasserstein`	orm used in Wasserstein metric	ℓ_1 or ℓ_2
SVR	`C`	Regularizing parameter	$[\mathrm{e}^{0.1}, \mathrm{e}^{7}]$
	`fit_intercept`	Bias	True or False
	`loss`	Loss function used	ℓ_1 or ℓ_2
GP	`length_scale`	Length-scale of the RBF kernel	$[\mathrm{e}^{0.1}, \mathrm{e}^{5}]$
	`periodicity`	Periodicity of the exp.-sine kernel	$[\mathrm{e}^{0.1}, \mathrm{e}^{5}]$
	`lenght_scale_sine`	Lenght-scale of the exp.-sine kernel	$[\mathrm{e}^{0.1}, \mathrm{e}^{5}]$
	`noise_level`	Noise level of the white noise kernel	$[\mathrm{e}^{0.001}, \mathrm{e}^{2}]$

A.3.2 Additional figures

In Figure A.3, we show the absolute values of the errors corresponding to Figure 3.5. We observe that errors have the same magnitude for each building. This is to be expected as each building has its own consumption range and its own level of pattern complexity. Because we use a high number of features and test for the winter season only, linear regressions, while having worse performances than nonlinear models in most cases, are tested in a setting close to their training set.

In Figure A.4, we present hourly winter predictions from the testing phase on a subset of buildings. This figure has qualitative purposes more than anything else and is better examined with colors. We observe that applying models naively does not always work for every building, e.g., building M, while it may perform quite well on others, e.g., building N.

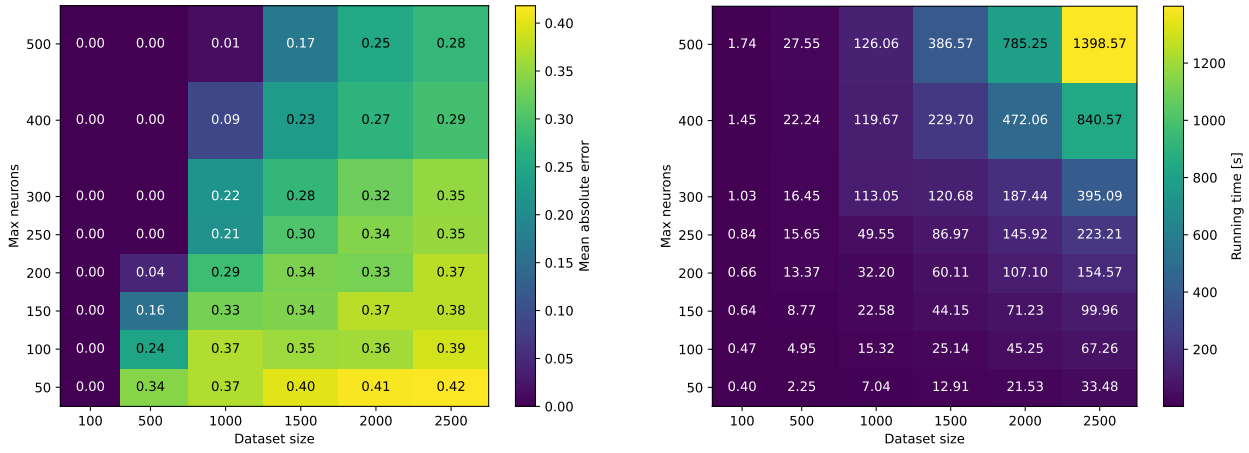
(a) Experiment A



(b) Experiment B

(c) Experiment C

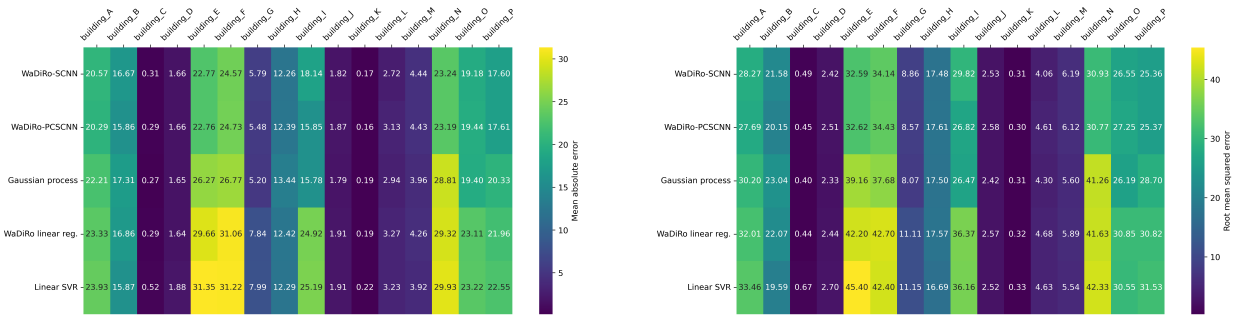
Figure A.1 Absolute errors of each synthetic experiment



(a) Training MAE

(b) Training time

Figure A.2 Training error and time on Ackley



(a) MAE

(b) RMSE

Figure A.3 Absolute errors of the hourly baseline prediction of non-residential buildings

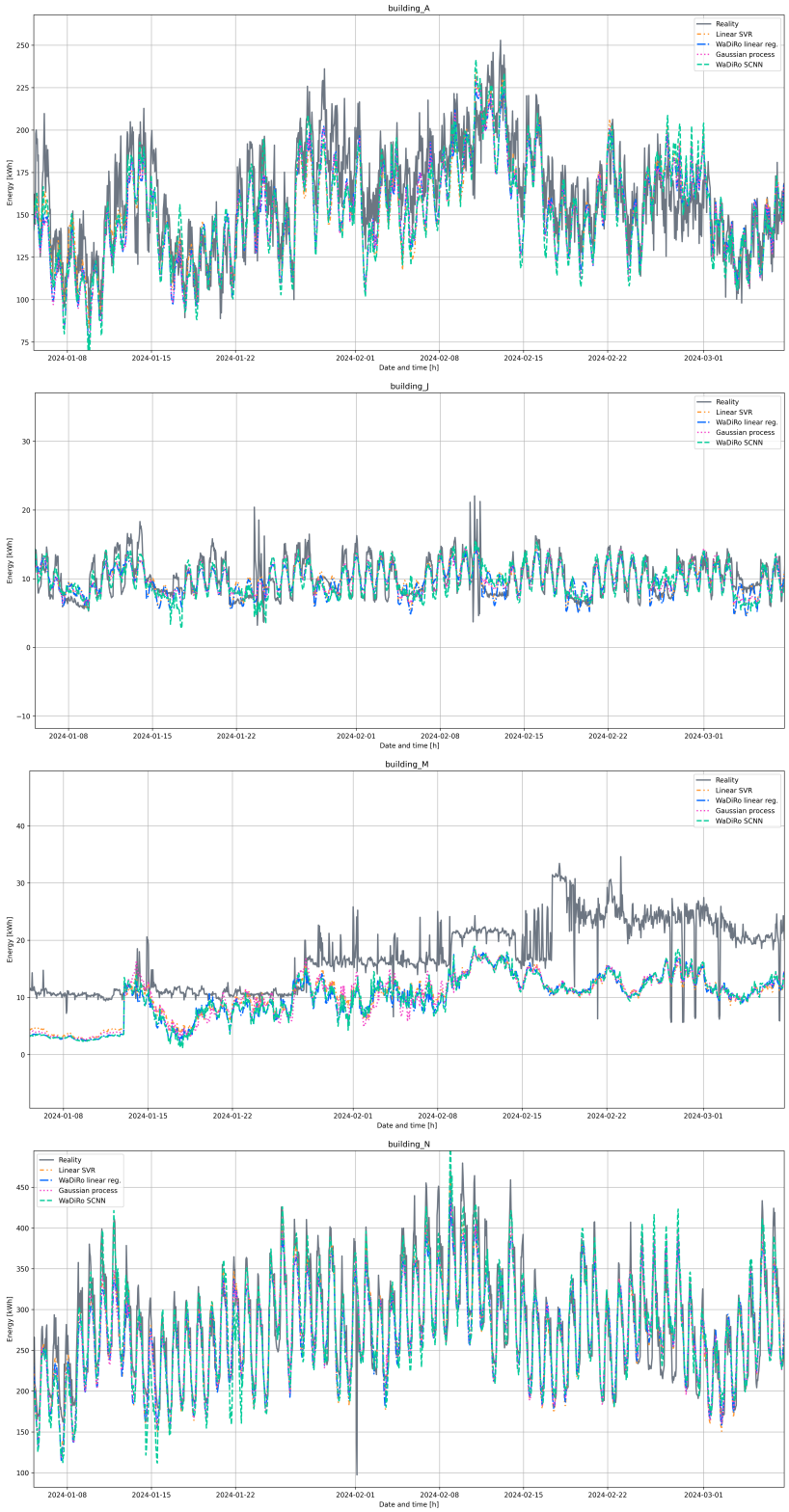


Figure A.4 Winter predictions of the testing phase for four buildings

APPENDIX B SUPPLEMENTARY CONTENT OF CHAPTER 4

B.1 Supplementary content to the anomaly detection experiment

In this section, we present the hyperparameters of the anomaly detection experiment. The experiment is a grid search, so the search space is discrete. The hyperparameters are presented in Table B.1.

Table B.1 Hyperparameters for the anomaly detection experiments

Model	**Hyperparameter**	**Description**	**Possible values**
LOF	`n_neighbors`	Number of neighbours	$\{10, 25, 50, 75, 100, 500\}$
	`algorithm`	Algorithm	{'auto', 'ball_tree', 'kd_tree', 'brute'}
	`leaf_size`	Leaf size	$\{5, 10, 25, 50, 75, 100, 500\}$
	`metric`	Distance metric	{'euclidean', 'manhattan', 'chebyshev', 'minkowski'}
`SWAD`	`eps`	Distance threshold	$\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000\}$
	`n`	Number of neighbours	$\{25, 75\}$
	`n_projections`	Number of projections	$\{100, 200\}$
	`p`	Voting threshold	$\{0.5, 0.7, 0.9\}$
`FEAD`	`eps`	Distance threshold	$\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000\}$
	`n`	Number of neighbours	$\{25, 75, 150, 200, 500\}$
	`p`	Voting threshold	$\{0.5, 0.7, 0.9\}$
Isol. Forest	`n_estimators`	Number of base estimators	$\{10, 50, 100, 500\}$
	`max_samples`	Number of samples	$\{1, n/4, n/2, 3n/4, n\}$
	`contamination`	Expected proportion of outliers	$\{0.01, 0.15, 0.3, 0.45\}$
	`max_features`	Number of features	$\{1, \frac{d}{2}, d\}$

B.2 Supplementary content to the data selection experiment

In this section, we present the averaged absolute MAE values of the experiment. See Figure B.1.

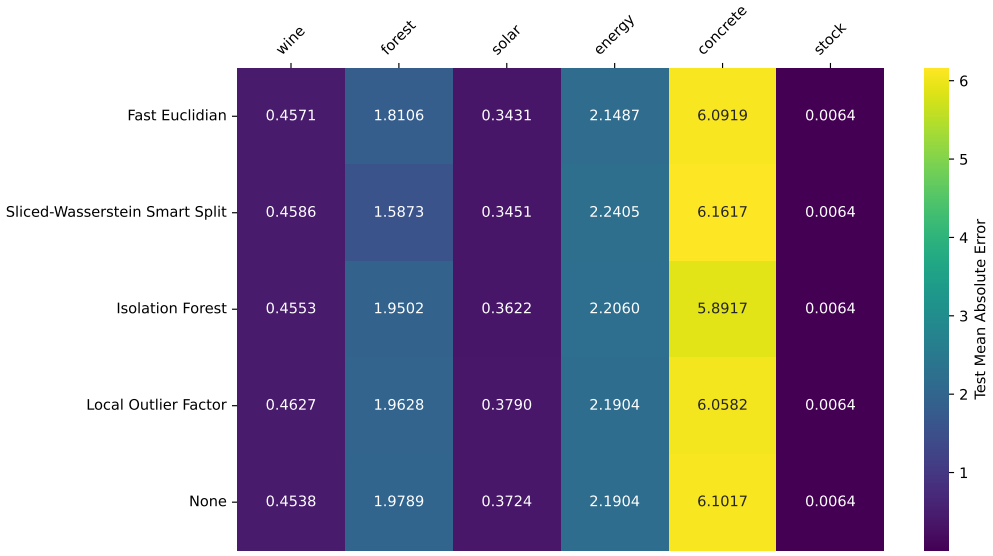


Figure B.1 Absolute MAE in testing of the data selection experiment

We also present the hyperparameters of the experiment in Table B.2.

Table B.2 Hyperparameters for the data selection experiment

Model	**Hyperparameter**	**Description**	**Possible values**
LOF	`n_neighbors`	Number of neighbours	$[5, 300]$ (integer)
	`algorithm`	Algorithm	{'auto', 'ball_tree', 'kd_tree', 'brute'}
	`leaf_size`	Leaf size	$[5, 300]$ (integer)
	`metric`	Distance metric	{'euclidean', 'manhattan'}
`split-SWAD`	`eps`	Distance threshold	$[\mathrm{e}^{-12}, \mathrm{e}^{1}]$
	`n`	Number of neighbours	$\{150, 300\}$
	`n_projections`	Number of projections	$\{40\}$
	`p`	Voting threshold	$\{0.7, 0.8, 0.9\}$
	`n_clusters`	Number of clusters	$\{3, 4\}$
	`n_splits`	Number of splits	$\{3, 4\}$
`FEAD`	`eps`	Distance threshold	$[\mathrm{e}^{-12}, \mathrm{e}^{7}]$
	`n`	Number of neighbours	$\{150, 300\}$
	`p`	Voting threshold	$\{0.7, 0.8, 0.9\}$
Isol. Forest	`n_estimators`	Number of base estimators	$[5, 300]$ (integer)
	`max_samples`	Proportion of samples	$(0, 1]$
	`contamination`	Expected proportion of outliers	$[\mathrm{e}^{-7}, \mathrm{e}^{-0.7}]$
	`max_features`	Proportion of features	$(0, 1]$

B.3 Detailed analysis of the LCPR dataset

In this section, we present some additional insights on the LCPR dataset. Figure B.2 presents the distribution count of some key features for each substation. We observe that meteorological features are identical for each substation as they are geographically adjacent and located in dense neighbourhoods.

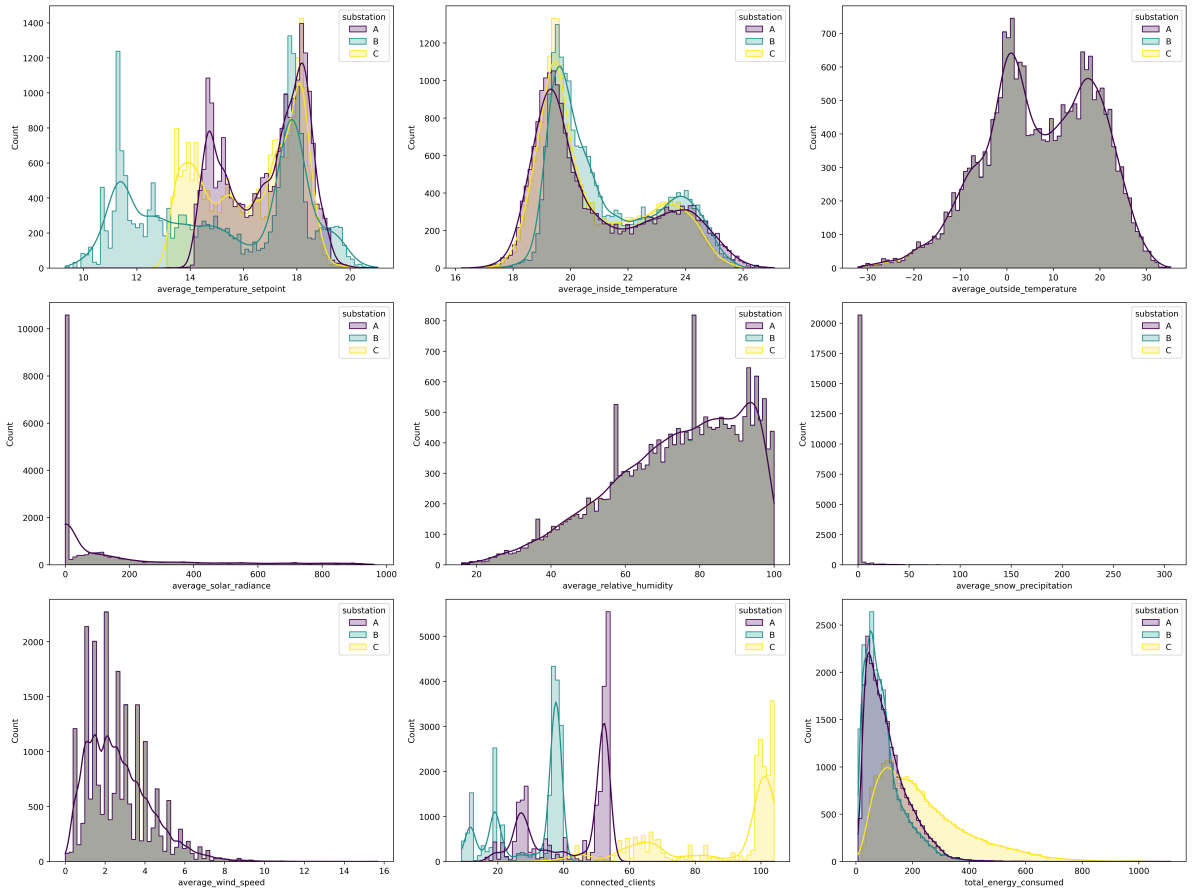
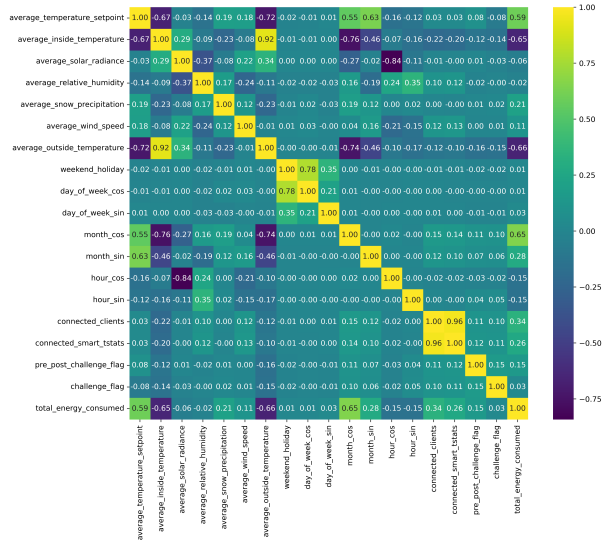


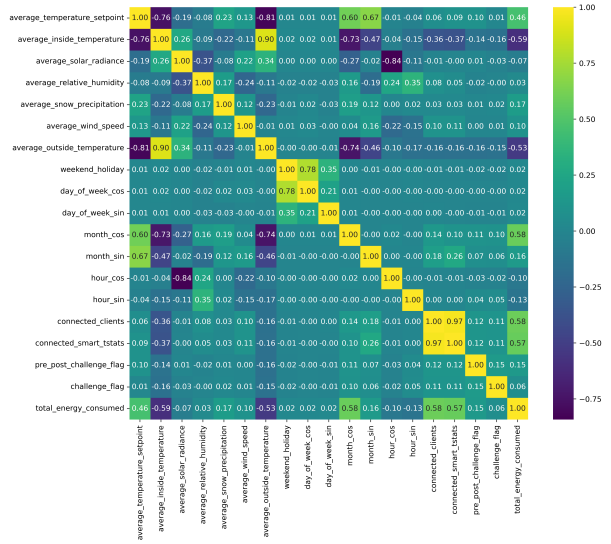
Figure B.2 Distribution of key features for each substation

Figure B.3 shows a correlation heatmap of important features and label for each substation. We observe that each substation follows the same general tendencies. We remark significant correlations between the energy consumed, the month of the year, the outside temperatures, and the temperature setpoints. This is also highlighted in Figure B.4 which presents the Spearman coefficients ranking Spearman (1904) between each feature and the label. A positive sign indicates that both the label and the feature grow or decrease in the same direction while a negative sign indicates an opposite direction. The coefficients are ranked in decreasing importance from left to right.

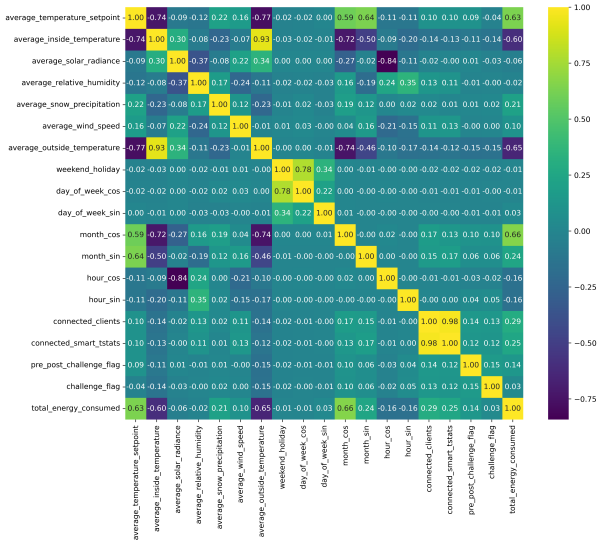
To have a more nuanced analysis of the contribution of each feature to the output, we also pro-



(a) Substation A



(b) Substation B



(c) Substation C

Figure B.3 Correlation heatmap of key features and label for each substation

vide in Figure B.5 an analysis of Shapley values of a trained extreme gradient boosting model (`XGBoost`) Chen and Guestrin (2016) for each substation. These analyses were realized with the Python package `SHAP` Lundberg and Lee (2017). As we see, some lower-ranked features, viz., the challenge flags, sometimes have a strong impact on the model's output even though their general impact is null.

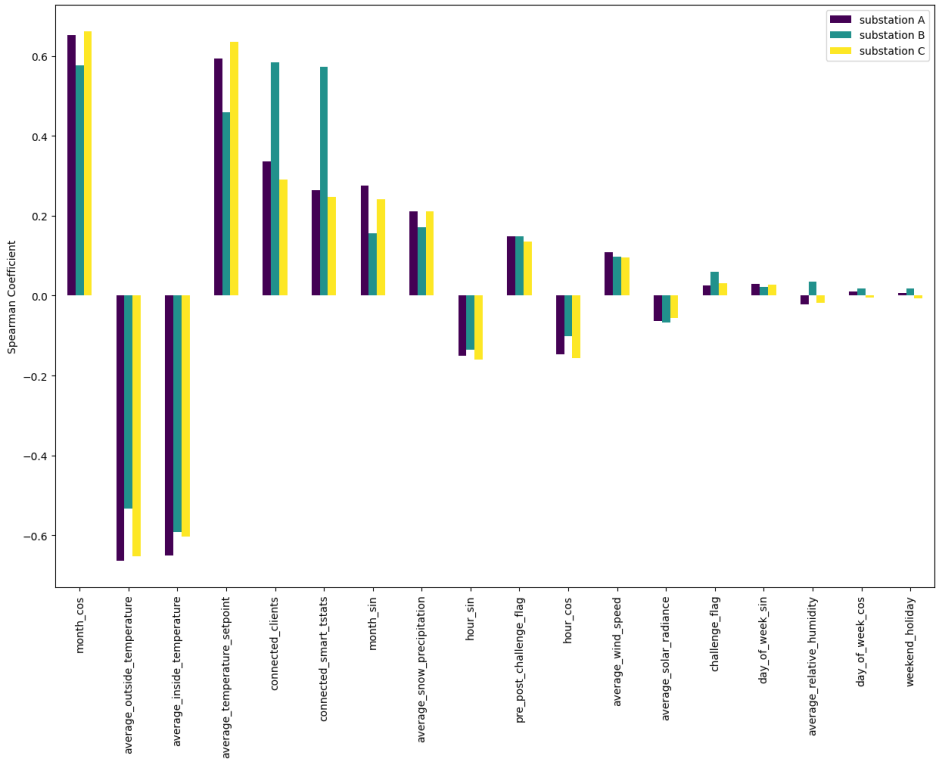
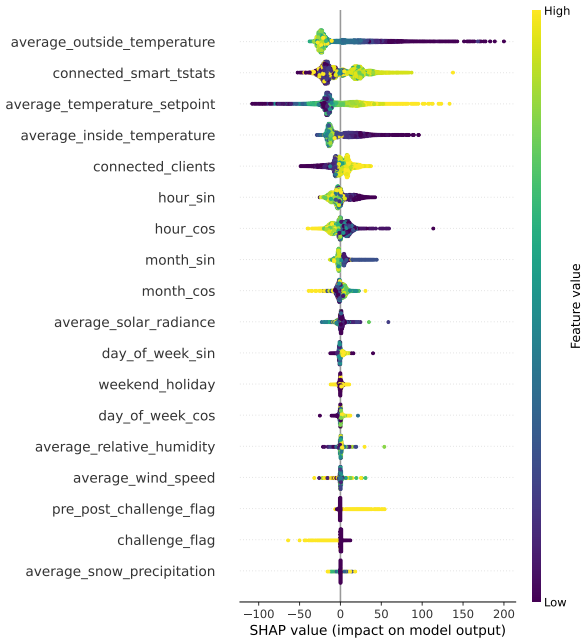


Figure B.4 Spearman coefficients of key features for each substation

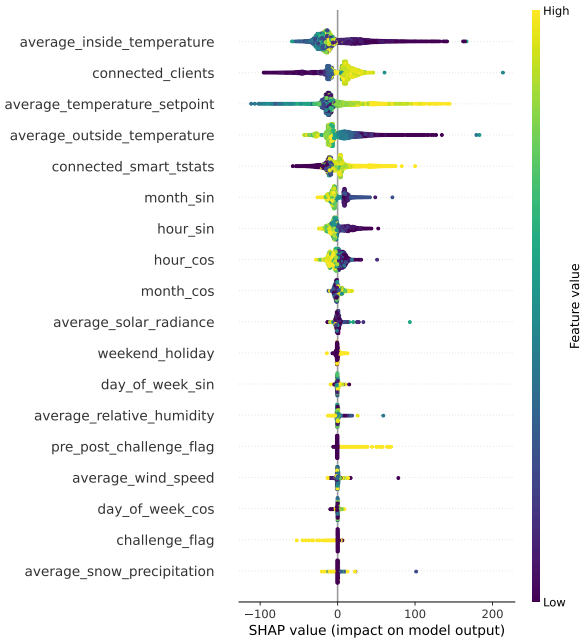
B.4 Supplementary content to the benchmark

This section presents visual test predictions of the best validation runs for each substation.

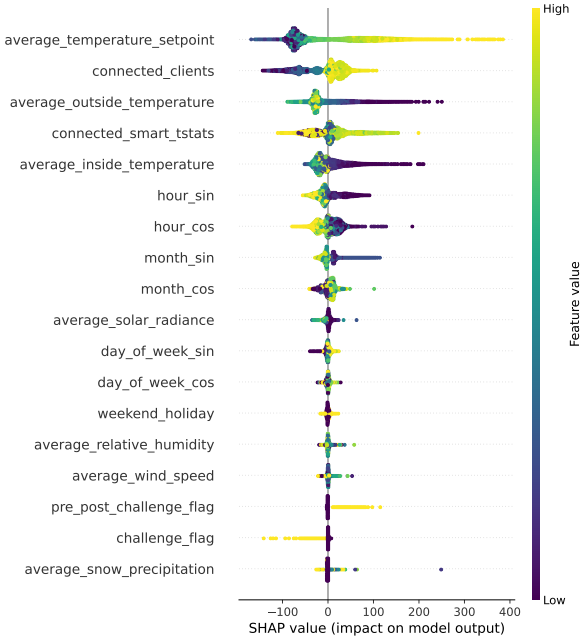
As can be seen in both Figure B.6 and Table 4.3, substation C is the hardest to predict for the Gaussian process. In general, the confidence of the model is also low (standard deviation is high). This hints at the complexity of the patterns.



(a) Substation A

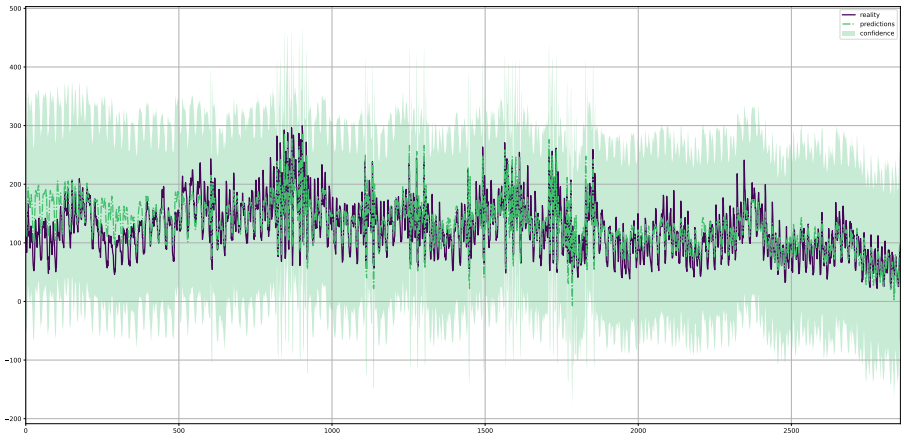


(b) Substation B

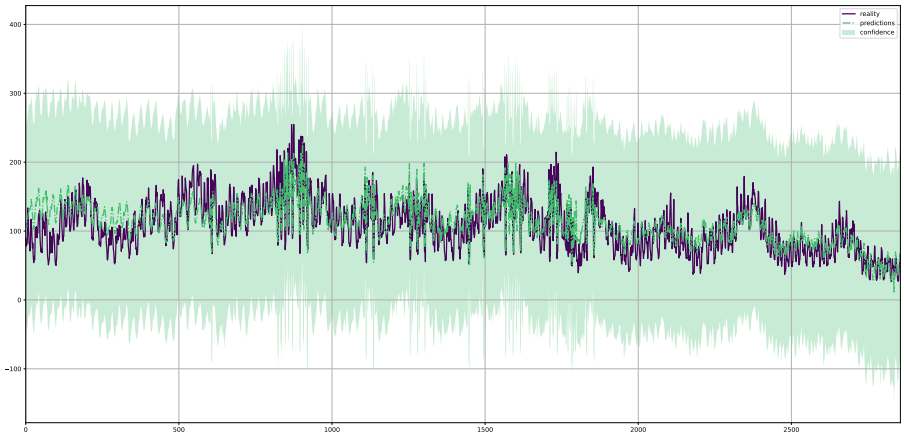


(c) Substation C

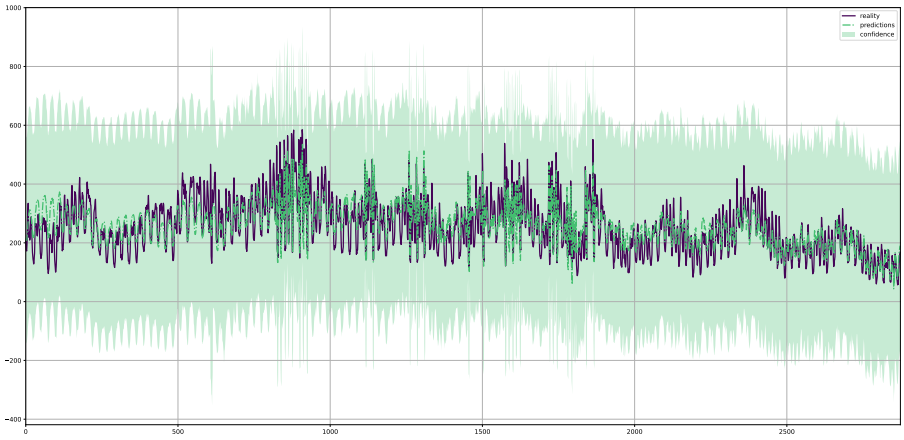
Figure B.5 Shapley analysis of key features for each substation on trained `XGBoost`



(a) Substation A



(b) Substation B



(c) Substation C

Figure B.6 Test predictions of the benchmark at each substation