Integrating Optimization and Modern Machine Learning: Theory, Computation, and Healthcare Applications

by

Kimberly M. Villalobos Carballo

B.S. Mathematics, MIT, 2019
B.S. Computer Science, MIT, 2019

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Kimberly M. Villalobos Carballo Sloan School of Management May 3, 2024
Certified by:	Dimitris Bertsimas Boeing Leaders for Global Operations Professor of Management Associate Dean for Business Analytics Thesis Supervisor
Accepted by:	Georgia Perakis John C Head III Dean (Interim), MIT Sloan School of Management Professor, Operations Management, Operations Research & Statistics Co-director, Operations Research Center

Integrating Optimization and Modern Machine Learning: Theory, Computation, and Healthcare Applications

by

Kimberly M. Villalobos Carballo

Submitted to the Sloan School of Management
on May 3, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

ABSTRACT

Optimization and machine learning are two predominant fields for decision-making today. The increasing availability of data over the past years has facilitated advancements in the intersection of these two domains, which in turn has led to better decision support tools. Optimization has significantly enhanced traditional machine learning models by refining their training methods, and machine learning has improved many optimization algorithms by enabling better decision-making through accurate predictions.

However, integrating optimization theory with modern machine learning methods, like neural networks and kernel functions, faces two primary challenges. Firstly, these models don't meet the fundamental convexity assumptions of optimization theory. Secondly, these models are primarily used in tasks with numerous parameters and high-dimensional data, requiring highly efficient and scalable algorithms. This focus on efficiency limits consideration for discrete variables and general constraints that are typical in optimization. This thesis introduces novel algorithms to address these challenges.

The work is divided into four chapters, encompassing rigorous theory, computational tools, and diverse applications. In Chapter 1, we extend state-of-the-art tools from robust optimization to non-convex and non-concave settings, allowing us to generate neural networks that are robust against input perturbations. In Chapter 2, we develop a holistic deep learning framework that jointly optimizes for neural network robustness, stability and sparsity by appropriately modifying the loss function. In Chapter 3 we introduce TabText, a flexible methodology that leverages the power of Large Language Models for patient flow predictions from tabular data. Lastly, in Chapter 4 we present a data-driven approach for solving multistage stochastic optimization problems via sparsified kernel methods.

Thesis supervisor: Dimitris Bertsimas
Title: Boeing Leaders for Global Operations Professor of Management
Associate Dean for Business Analytics

Acknowledgments

I want to sincerely thank Dimitris Bertsimas, my academic advisor, for his unwavering guidance, support, and encouragement during my five-year PhD journey. Dimitris has shown me how impactful research can be and how important it is to focus on problems that can help others. Beyond academics, Dimitris has taught me about leadership, communication, teamwork, and more – I never leave his office without some invaluable piece of life advice. As a mentor, he has consistently acknowledged my strengths while also helping me identify and address my weaknesses to foster improvement. Dimitris, I am extremely grateful to you for persuading and supporting me in pursuing a career as a professor. If I achieve even a fraction of what you have accomplished, I will consider it a success. Thank you for caring for me and inspiring me to become a better researcher, teacher, and person.

I also want to express my gratitude to Dick den Hertog, Vivek Farias, and Swati Gupta for being part of my thesis committee and supporting me in my faculty job search. I truly appreciate all your time and advice. Dick, I'm also deeply grateful for your mentorship as both my professor and collaborator. Your guidance has been incredibly impactful, and I've learned so much from you. I also thank Georgia Perakis for her support and kindness throughout my PhD; your academic and emotional advice has been exceptional.

I am grateful for all the collaborators I have been lucky to work with throughout my doctoral studies. Special thanks to Jean Pauphillet, Xavier Boix, Ignacio Fuentes, and Barry Stein, whom I admire deeply and thank for all their mentorship and support.

At MIT I have had the privilege of meeting people that aren't just outstanding academics, but also genuine and loving friends. I am profoundly grateful to Irra, for being the best teammate and friend I could have asked for. The panic attacks when our code broke, our late-night girl calls, as well as our joint academic recognitions are memories that I will keep very close to my heart. I'm also immensely grateful to Adrian, Giancarlo, and Shalom for making the pandemic times bearable and for being such unconditional friends. Thank you Yu, Cynthia, Leonard, Amine, Moise, for so much love and support over the past years. Thanks to Manu, Shuvo, El Ghali, Leann, Patricio, Ted, Ryan, Michael, Vassili, and Holy, for gifting me so many great memories of my PhD.

Thanks to all my friends from home who helped me get here, including Santiago, Juleana, Colleen, Tomas, Daniel, Marco and Jonathan. Thanks to OLCOMA, UCR, UNA, Education USA, and all other institutions of Costa Rica that supported my education. Thanks to all my teachers and mentors. Special thanks to Jairo Villegas, Mariechen Wust, Julio Salazar, and Ronald Bustamante, who have always believed in my big dreams.

Gracias a todos mis amigos de Costa Rica que me ayudaron a llegar hasta aquí, incluyendo a Santiago, Juleana, Colleen, Tomas, Daniel, Marco y Jonathan. Gracias a OLCOMA, la UCR, UNA, Education USA, y todas las demás instituciones que apoyaron mi educación. Gracias a todos mis profesores y mentores. Un agradecimiento especial a Jairo Villegas, Mariechen Wust, Julio Salazar y Ronald Bustamante, quienes siempre han creído y apoyado mis sueños.

To my boyfriend, Sean, thank you for standing by my side all these years. Thank you for holding my hand on the most difficult days, for staying up late on my long nights, and for celebrating with me every good moment. I look forward to many more years next to you.

Finally, I am infinitely thankful for my family; to whom I owe everything. Thanks to my grandparents Fermina and Gregorio for their loving and encouraging words, to my grandma Ana for always praying and lighting up candles on my stressful days, and to my grandpa Noe, who left us too early but to this day motivates me to believe in myself. Thanks to my sister Jessica, who is also my best teacher and my first role model. Thanks to my beautiful nephews

Jeremy and Daniel, who inspire me every day. Above all, thanks to my parents Maricruz Carballo and Felipe Villalobos, who despite all the financial distress gave me the greatest example of courage and hard work. Only God knows the extent of sacrifices you both made for me to be here. It is with the most heartfelt gratitude that I dedicate this thesis to you.

Finalmente, estoy infinitamente agradecida por mi familia; a quienes les debo todo lo que soy. Gracias a mis abuelos Fermina y Gregorio por sus palabras amorosas y alentadoras, a mi abuela Ana por siempre rezar y encender velitas en días importantes, y a mi abuelo Noé, quien nos dejó demasiado pronto pero hasta el día de hoy me sigue motivando a creer en mí misma. Gracias a mi hermana Jéssica, quien también es mi mejor maestra y mi primer modelo a seguir. Gracias a mis bellos sobrinos Jeremy y Daniel, quienes me inspiran cada día. Sobre todo, gracias a mis padres Maricruz Carballo y Felipe Villalobos, quienes a pesar de todas las dificultades financieras me dieron el mejor ejemplo de coraje y esfuerzo. Solo Dios sabe el alcance de los sacrificios que ambos hicieron para que yo esté aquí. Con el más sincero agradecimiento, esta tesis se la dedico a ustedes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization and machine learning stand as two highly successful disciplines in decision-making, with important applications across various sectors such as healthcare, energy, finance, and more. Historically, the two were studied independently – optimization as a fundamental pillar of Operations Research and machine learning burgeoning in Computer Science. The escalating availability of data in recent years has naturally promoted the joint study of machine learning with other scientific domains, with optimization being no exception. Optimization has contributed significantly to the improvement of traditional machine learning models by enhancing their training methodologies. Conversely, machine learning has helped improving optimization algorithms, as access to accurate predictions of future outcomes allows to make better decisions.

Integrating optimization theory with modern machine learning methods, such as neural networks and kernel functions, presents unique challenges compared to more conventional models like logistic regression, trees, and support vector machines. Firstly, neural networks do not satisfy the fundamental convexity assumptions that underlie optimization theory. Secondly, these models are mostly applied in tasks involving a large number of parameters and high dimensional observations, requiring the algorithms employed to be highly efficient and scalable. Maintaining those properties leaves little or no room for discrete variables and general constraints, which are typical in optimization.

This thesis presents novel algorithms that alleviate the challenges described above and successfully integrate optimization with modern machine learning models. We focus on two key problems:

- *Machine Learning Classification*: Deep learning is one of the most important scientific developments of our time; however, deep learning models face notable challenges when applied in practical domains. They struggle with small datasets, tabular data formats, and they lack robustness and stability. Additionally, their computational demands make them intractable in settings constrained by memory or time. We propose the use of tools from robust optimization, non-convex optimization, and large language models to address these limitations.

- *Multistage Stochastic Optimization*: This active research area has applications to various problems like patient flow optimization and inventory control. Recent efforts have utilized predictive analytics to inform decision-making using available side information and historical data. However, dynamic methods like these are hindered by the curse of dimensionality and struggle to scale in settings with large datasets or high-dimensional decisions. We propose leveraging machine learning with kernel methods and functional optimization to develop a scalable data-driven algorithm for this problem.

This thesis is divided into four chapters, encompassing rigorous theory, computational tools, and diverse applications. In Chapter 1, we extend state-of-the-art tools from robust optimization to non-convex and non-concave settings, allowing us to generate neural networks that are robust against input perturbations. In Chapter 2, we develop a holistic deep learning framework that jointly optimizes for neural network robustness, stability and sparsity by appropriately modifying the objective function. In Chapter 3 we introduce TabText, a flexible methodology that leverages the power of Large Language Models for patient flow predictions from tabular data. Lastly, in Chapter 4 we present a data-driven approach for solving multistage stochastic optimization problems via sparsified kernel methods.

The following sections offer a summary of the work as well as the contributions within each chapter of the thesis.

1.1 Robust, Stable and Sparse Optimization for Neural Networks

Deep learning has emerged as the predominant technology of our times, with important applications in areas like computer vision, natural language processing, and structural biology. However, in recent years it has been exposed that standard neural networks lack robustness – they are susceptible to being misled by both natural and artificial noise in the input data. This problem becomes particularly relevant when considering applications related to self-driving cars or medicine, in which these perturbed inputs represent an important security threat.

In Chapter 1, we develop a novel algorithm to train robust neural networks by extending state-of-the-art tools from robust optimization to non-convex and non-concave settings. In particular, we find a closed-form expression that provides an upper bound on the worst-case training loss with respect to bounded input perturbations. By minimizing this upper bound, our method, Robust Upper Bounds (RUB), not only achieves state-of-the-art performance empirically, but also provides performance guarantees against the perturbations considered. We also propose a simple method (Approximated Robust Upper Bound or aRUB) which uses the first order approximation of the network as well as basic tools from linear robust optimization to obtain an empirical upper bound of the adversarial loss that can be easily implemented. Across a variety of tabular and vision data sets we demonstrate the effectiveness of our approach —RUB is substantially more robust than state-of-the-art methods for larger perturbations, while aRUB matches the performance of state-of-the-art methods for small perturbations (Bertsimas et al. 2021a).

Robustness is; however, not the only challenge that neural networks face. These models frequently exhibit instability during the training process, where different train-validation splits can result in models with significantly varied performance. Additionally, these networks contain millions of non-zero parameters that need to be stored and accessed for evaluation. Previous works on robustness, stability and sparsity of neural networks have only addressed these challenges in isolation, and often at the expense of a substantial increase in computational time. In Chapter 2, we focus on developing practical algorithms to optimize for these properties jointly and analyze their trade-offs.

In order to enhance stability with respect to train-validation splits, we modify the loss function to optimize for the empirical value-at-risk of the error (instead of the average error) using a mixed-integer programming formulation. In addition, we incorporate a continuous approximation of the L0 pseudo-norm as a penalty in the objective to enforce sparsity. We combine these algorithms with the robustness methods from Chapter 1 to develop a holistic deep learning framework (HDL) that jointly optimizes for the metrics of interest. HDL demonstrates that it is often possible to simultaneously improve robustness, stability, and sparsity without sacrificing performance on accuracy. In fact, we show that adding robustness and stability can significantly improve the accuracy of the network, especially for tabular data sets. Since our approach (and our code) supports a variety of loss functions, we also provide guidance to practitioners to help them align the training objective with their specific use case (Bertsimas et al. 2024).

Contributions

- We develop two new methods for training deep learning models that are robust against input perturbations. The first method (*Approximated Robust Upper Bound* or aRUB), minimizes an empirical upper bound of the adversarial loss for L_p norm bounded uncertainty sets, for general p. It is simple to implement and performs similar to state-of-the-art defenses on small uncertainty sets. The second method (*Robust Upper Bound* or RUB), minimizes a provable upper bound of the adversarial loss specifically for L_1 norm bounded uncertainty sets. This method shows the best performance for larger uncertainty sets, and more importantly, it provides security guarantees against L_1 norm bounded adversarial attacks.

- We design HDL, a novel framework that jointly optimizes for neural network robustness, stability, and sparsity metrics by appropriately modifying the objective function. Through extensive ablation experiments across tabular and image sets, we analyze the individual performance of each metric as well as the interactions and trade-offs between them.

- We propose a prescriptive approach to provide recommendations on selecting the appropriate loss function for a classification task depending on the practitioner's metric of interest.

1.2 Large Language Models for Tabular Data Representations

Accurate predictions of patient outcomes can facilitate resource allocation and enhance personalized care. In collaboration with a large hospital network, we developed machine learning models that predict short-term and long-term outcomes for all inpatients across their seven hospitals using electronic medical records. We implemented an automated pipeline displaying our daily predictions with user-friendly software. Over 200 medical staff currently use our tool, resulting in a significant reduction in length of stay and projected annual benefits of millions of dollars for the healthcare system (Na et al. 2023).

Given this successful implementation, the question arises: how could we extend these tools for the benefit of hospitals with limited resources, small patient populations, and/or non-standardized healthcare records? Even though electronic medical records are widely available for most digitized healthcare systems, tabular data in healthcare is generally disorganized, not standardized across institutions and scarce in small healthcare systems. We then identified two significant limitations in the existing approaches to handling tabular data: they require labor-intensive data processing, and they ignore contextual information such as column headers and meta content descriptions which could be used for data augmentation.

To address these limitations, in Chapter 3 we present TabText, a systematic framework that leverages Large Language Models to extract contextual information from tabular structures, resulting in more complete and flexible data representations. These new representations can then be used to train any standard machine learning model for downstream prediction tasks. Although deep learning models often perform poorly on classification tasks with structured data, off-the-shelf and pre-trained neural networks can be remarkably helpful for enhancing pre-processing pipelines. We demonstrated the flexibility of our approach compared to traditional labor-consuming processing techniques, and we showed that TabText can significantly improve performance across all patient outcome classification tasks considered, especially those with small data sizes and high variability (Carballo et al. 2022).

Contributions

- We develop and implement machine learning models that predict several inpatient outcomes at a healthcare system. We show that after utilizing our user-friendly software the hospitals observe significant reduction in length of stay.

- We develop TabText, a systematic framework that leverages language to extract contextual information from tabular structures. Our experiments demonstrate that augmenting electronic medical records with our TabText representations can significantly improve the AUC score, especially when trained with small-size datasets. We also show that TabText enables the generation of high-performing predictive models for patient outcomes with minimal data processing.

1.3 Sparse Reproducing Kernels for Stochastic Multistage Optimization with Covariates

Multistage stochastic optimization arises in numerous applications and remains an important research area in the optimization community. Recent work has focused on using predictive analytics to leverage available side information and historical data to make better decisions. However, these dynamic methods are affected by the curse of dimensionality; they require scenario tree enumeration and can require many hours for solving problems with only a few stages. More recently, kernel methods have been used for data-driven, single-period optimization problems with auxiliary information. This approach overcomes the curse of dimensionality; however, the number of parameters per decision grows linearly with the number of observations, resulting in function representations that are as complex as the size of the data and that become potentially intractable especially in multistage settings.

In Chapter 4 we develop a non-parametric, data-driven, tractable approach for solving multistage stochastic optimization problems in which decisions do not affect the uncertainty. The proposed framework represents the decision variables as elements of a reproducing kernel Hilbert space and performs functional stochastic gradient descent to minimize the empirical regularized loss. By incorporating sparsification techniques based on function subspace projections we are able to overcome the computational complexity that standard kernel methods introduce as the data size increases. We prove that the proposed approach is asymptotically optimal for multistage stochastic optimization with side information. Across various computational experiments on stochastic inventory management problems, our method performs well in multidimensional settings and remains tractable when the data size is large. Lastly, by computing lower bounds for the optimal loss of the inventory control problem, we show that the proposed method produces decision rules with near-optimal average performance (Bertsimas and Carballo 2023).

Contributions

- We propose a novel data-driven approach for multistage stochastic optimization problems with side information based on reproducing kernel Hilbert spaces and sparse projections. To the best of our knowledge, this is the first tractable application of reproducing kernel Hilbert spaces to multistage optimization problems with large data sizes.
- We prove that under standard convexity and smoothness conditions on the loss function, the expected loss achieved with our algorithm achieves asymptotic optimality.
- We demonstrate across several instances of inventory management problems that the proposed method finds near-optimal solutions using only a few parameters and with very low computational times even for large instances of the problem.

Chapter 2

Robust Deep Learning

2.1 Introduction

Robustness of neural networks for classification problems has received increasing attention in the past few years, since it was exposed that these models could be easily fooled by introducing some small perturbation in the input data. These perturbed inputs, which are commonly referred to as *adversarial examples*, are visually indistinguishable from the natural input, and neural networks simply trained to maximize accuracy often assign them to an incorrect class (Szegedy et al. 2013). This problem becomes particularly relevant when considering applications related to self-driving cars or medicine, in which adversarial examples represent an important security threat (Kurakin et al. 2016).

The machine learning community has recently developed multiple heuristics to make neural networks robust. The most popular ones are perhaps those based on training with adversarial examples, a method first proposed by Goodfellow et al. (2015) and which consists in training the neural network using adversarial inputs instead of or in addition to the standard data. The defense by Madry et al. (2019), which finds the adversarial examples with bounded norm using iterative projected gradient descent (PGD) with random starting points, has proved to be one of the most effective methods (Tjeng et al. 2019), although it comes with a high computational cost. Another more efficient defense was proposed by Wong et al. (2020), which uses instead fast gradient sign methods (FGSM) to find the attacks. Other heuristic defenses rely on preprocessing or projecting the input space (Lamb et al. 2018, Kabilan et al. 2018, Ilyas et al. 2017), on randomizing the neurons (Prakash et al. 2018, Xie et al. 2017) or on adding a regularization term to the objective function (Ross and Doshi-Velez 2017, Hein and Andriushchenko 2017, Yan et al. 2018). There is a plethora of heuristics for adversarial robustness by now. Yet, these defenses are only effective to adversarial attacks of small magnitude and are vulnerable to attacks of larger magnitude or to new attacks (Athalye et al. 2018).

Given the lack of an exact and tractable reformulation of the adversarial loss with norm-bounded perturbations, a recent strand of research has been to leverage upper bounds to improve adversarial robustness. These upper bounds provide security guarantees against adversarial attacks, even new ones, by finding a mathematical proof that a network is not susceptible to any attack, e.g. (Dathathri et al. 2020, Raghunathan et al. 2018b, Katz et al. 2017, Tjeng et al. 2019, Bunel et al. 2017, Anderson et al. 2020, Singh et al. 2018, Zhang et al. 2018, Weng et al. 2018, Gehr et al. 2018, Dvijotham et al. 2018b, Lecuyer et al. 2019, Cohen et al. 2019). Replacing the standard loss with these upper bounds during training is a common technique for obtaining adversarial defenses. Wong and Kolter (2018) for instance, find an upper bound for the adversarial loss by applying linear relaxations in the network and computing a convex polytope that contains all possible values for the last layer given adversarial examples with bounded norm. An upper bound on the adversarial loss is also computed in Raghunathan et al. (2018a) by solving instead a semidefinite program. Other more scalable and effective methods based on minimizing an upper bound of the adversarial loss have been introduced (Gowal et al. 2019, Balunovic and Vechev 2019, Mirman et al. 2018, Dvijotham et al. 2018a, Wong et al. 2018, Zhang et al. 2019).

While these methods can provide security guarantees against adversarial examples, most of them rely on convex relaxations to recursively compute upper and lower bounds for each layer, which introduces gaps that propagate and can affect the final bound for the last layer. For instance, the approach proposed in Gowal et al. (2019) computes bounds for each layer by assuming that the worst-case bounds for all previous layers can be achieved simultaneously. This often yields a loose upper bound of the adversarial loss whose minimization can be sensitive to hyperparameters (Zhang et al. 2019). Another example is the aforementioned defense from Wong and Kolter (2018), where bounds at each layer are computed by solving a linear program that uses the bounds from previous layers for the linear ReLU relaxations. Unlike these approaches, the method proposed in Raghunathan et al. (2018a) does not require computation of intermediate bounds, however, their proposed upper bound only works for neural networks with two layers.

Upper bounds for the adversarial loss have also been explored in the context of Distributionally Robust Optimization, in which the data distribution is perturbed within a Wasserstein ball (Sinha et al. 2017, Shafieezadeh-Abadeh et al. 2019). These works find upper bounds for the worst-case expected loss and provide generalization guarantees under certain assumptions. However, it is not evident how to apply these methods for norm-bounded input perturbations given the different nature of the uncertainty.

A promising yet under-explored approach is the application of state-of-the-art Robust Optimization (RO) tools (Bertsimas and den Hertog 2022). RO has proven to be effective in handling uncertainty in parameters that may result from rounding or implementation errors. Recently, it has also been applied to provide robustness against input perturbations in some machine learning models, such as Support Vector Machines and Optimal Classification Trees (Bertsimas et al. 2019), and it could be similarly leveraged for deep learning. While previous works on robustness of neural networks generally formulate the problem in the context of RO, they do not utilize the more advanced tools available in this field. Instead, they mostly depend on linear or convex relaxations and heuristic methods to simplify the original non-convex problem. In this paper, we use state-of-the-art RO tools to derive a new closed-form solution of an upper bound of the adversarial loss. Our approach is based on a holistic expansion of the network; it does not rely on convex relaxations or separate computation of bounds for each layer of the network, and it can still be effectively trained with backpropagation.

We develop two new methods for training deep learning models that are robust against input perturbations. The first method (*Approximated Robust Upper Bound* or aRUB), minimizes an empirical upper bound of the adversarial loss for L_p norm bounded uncertainty sets, for general p. It is simple to implement and performs similar to state-of-the-art defenses on small uncertainty sets. The second method (*Robust Upper Bound* or RUB), minimizes a provable upper bound of the adversarial loss specifically for L_1 norm bounded uncertainty sets. This method shows the best performance for larger uncertainty sets, and more importantly, it provides security guarantees against L_1 norm bounded adversarial attacks. More concretely, we introduce the following robustness methods:

- *Approximated Robust Upper Bound* or aRUB: We develop a simple method to approximate an upper bound of the adversarial loss by adding a regularization term for each target class separately. As an alternative to standard adversarial training (which relies on linear approximations to find good adversarial attacks), we use the first order approximation of the network to estimate the worst case scenario for each individual class. We then apply standard results from Linear Robust Optimization to obtain a new objective that behaves like an upper bound of the adversarial loss and which can be tractably minimized for robust training. This method can be easily implemented and performs very well when the uncertainty set radius ρ is small.

- *Robust Upper Bound* or RUB: We extended state-of-the-art tools from RO to functions that like neural networks are neither convex nor concave. By splitting each layer of the network as the sum of a convex function and a concave function, we are able to obtain an upper bound of the adversarial loss for the case in which the uncertainty set is the L_1 sphere. Since the dual function of the L_1 norm is the L_∞ norm, we convert the maximum over the uncertainty set into a maximum over a finite set. In the end, instead of minimizing the worst case loss over an infinite uncertainty set, the new objective minimizes the worst case loss over a discrete set whose cardinality is twice the dimension of the input data. While this represents a significant increase in memory for high dimensional inputs, we show that this approach remains tractable for multiple applications. The main advantage of this method is that it provides security guarantees against adversarial examples bounded in the L_1 norm. Additionally, we also show experimentally that this method generally achieves the highest adversarial accuracies for larger uncertainty sets.

Also, we show that these methods consistently achieve higher standard accuracy (i.e., non adversarial accuracy), than the nominal neural networks trained without robustness. While this result is not true for a general choice of uncertainty set (see for example Ilyas et al. (2019)), we observe that when the uncertainty set has the appropriate size it can significantly improve the classification performance of the network, which is consistent with the results obtained for other classification models like Support Vector Machines, Logistic Regression and Classification Trees (Bertsimas et al. 2019).

The paper is organized as follows: Section 2.2 revisits previous works on RO, Section 2.3 defines the robust problem, Section 2.4 presents the first method (Approximate Robust Upper Bound), and Section 2.5 contains the second method (Robust Upper Bound). Lastly, Section 5.7 contains the results for the computational experiments.

2.2 Previous Works on Robust Optimization

Over the last two decades, RO has become a successful approach to solve optimization problems under uncertainty. For an overview of the primary research in this field we refer the reader to Bertsimas et al. (2011). Areas like mathematical programming and engineering have long applied these tools to develop models that are robust against uncertainty in the parameters, which may arise from rounding or implementation errors. For many applications, the robust problem can be reformulated as a tractable optimization problem, which is referred to as the *robust counterpart*. For instance, for several types of uncertainty sets, the robust counterpart of a linear programming problem can be written as a linear or conic programming problem (Ben-Tal et al. 2009), which can be solved with many of the current optimization software. While there is not a systematic way to find robust counterparts for a general nonlinear uncertain problem, multiple techniques have been developed to obtain tractable formulations in some specific nonlinear cases.

As shown in Ben-Tal et al. (2009), the exact robust counterpart is known for Conic Quadratic problems and Semidefinite problems in which the uncertainty is finite, an interval or an unstructured norm-bounded set. More generally, it is shown in Ben-Tal et al. (2015) that for problems in which the objective function or the constraints are concave in the uncertainty parameters, Fenchel duality can be used to exactly derive the corresponding robust counterpart. While the result does not necessarily have a closed-form, the authors show that it yields a tractable formulation for the most common uncertainty sets (e.g. polyhedral and ellipsoidal uncertainty sets).

The problem becomes significantly more complex when the functions in the objective or in the constraints are instead convex in the uncertainty (Chassein and Goerigk 2019). Since obtaining provable robust counterparts in these cases is generally infeasible, safe approximations are considered instead (Bertsimas et al. 2023). For instance, Zhen et al. (2017) develop safe approximations for the specific cases of second order cone and semidefinite programming constraints with polyhedral uncertainty. These techniques are generalized in Roos et al. (2020), where the authors convert the robust counterpart to an adjustable RO problem that produces a safe approximation for any problem that is convex in the optimization variables as well as in the the uncertain parameters.

Even though the approaches mentioned above consider uncertainty in the parameters of the model as opposed to uncertainty in the input data, the same techniques can be utilized for obtaining robust counterparts in the latter case. In fact, the robust optimization RO methodologies have recently been applied to develop machine learning models that are robust against perturbations in the input data. In Bertsimas et al. (2019), for example, the authors consider uncertainty in the data features as well as in the data labels to obtain tractable robust counterparts for some of the major classification methods: support vector machines, logistic regression, and decision trees. However, due to the high complexity of neural networks as well as the large dimensions of the problems in which they are often utilized, robust counterparts or safe approximations for this type of models have not yet been developed. There are two major challenges with applying RO tools for training robust neural networks:

(i) *Neural networks are neither convex nor concave functions:* As mentioned earlier, robust counterparts are difficult to find for a general problem. Although plenty of work has been done to find tractable reformulations as well as safe approximations, all of them rely on the underlying function being a convex or a concave function of the uncertainty parameters. Unfortunately, neural networks don't satisfy either condition, which makes it really difficult to apply any of the approaches discussed above.

(ii) *The robust counterpart needs to preserve the scalability of the model:* Neural networks are most successful in problems involving vision data sets, which often imply large input dimensions and enormous amount of data. For the most part, they can still be successfully trained thanks to the fact that back propagation algorithms can be applied to solve the corresponding unconstrained optimization problem. However, the RO techniques for both convex and concave cases often require the addition of new constraints and variables for each data sample, increasing significantly the number of parameters of the network and making it very difficult to use standard machine learning software for training.

A straightforward way to overcome both of these difficulties would be to replace the loss function of the network with its first order approximation. However, this loss function is usually highly nonlinear and therefore the linear approximation is very inaccurate. Our method aRUB explores a slight modification of this approach that significantly improves adversarial accuracy by considering only the linear approximation of the network's output layer.

Alternatively, a more rigorous approach to overcome problem (i) would be to piece-wise analyze the convexity of the network and apply the RO techniques in each piece separately, but this approach would introduce additional variables that are in conflict with requirement (ii). For the proposed RUB method we then develop a general framework to split the network by convexity type, and we show that in the specific case in which the uncertainty set is the L_1 norm bounded sphere, we can solve for the extra variables and obtain an unconstrained problem that can be tractably solved using standard gradient descent techniques.

2.3 The Robust Optimization problem

We consider a classification problem over data points $\boldsymbol{x} \in \mathbb{R}^M$ labeled with one of K different classes in [K], where we use the notation [n] to denote the set $\{1, \ldots, n\}$. Given weight matrices $\boldsymbol{W}^\ell \in \mathbb{R}^{r_{\ell-1} \times r_\ell}$ and bias vectors $\boldsymbol{b}^\ell \in \mathbb{R}^{r_\ell}$ for $\ell \in [L]$, such that $r_0 = M, r_L = K$, the corresponding feed forward neural network with L layers and ReLU activation function is defined by the equations

$$\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) = \boldsymbol{W}^1 \boldsymbol{x} + \boldsymbol{b}^1, \tag{2.1}$$

$$\boldsymbol{z}^\ell(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) = \boldsymbol{W}^\ell [\boldsymbol{z}^{\ell-1}(\boldsymbol{\mathcal{W}}, \boldsymbol{x})]^+ + \boldsymbol{b}^\ell, \quad \forall\, 2 \leq \ell \leq L, \tag{2.2}$$

where $\boldsymbol{\mathcal{W}}$ denotes the set of parameters $(\boldsymbol{W}^\ell, \boldsymbol{b}^\ell)$ for all $\ell \in [L]$ and $[\boldsymbol{x}]^+$ is the result of applying the ReLU function ($[x]^+ = \max\{x, 0\}$) to each coordinate of $\boldsymbol{x}$. For fixed parameters $\boldsymbol{\mathcal{W}}$, the network assigns a sample $\boldsymbol{x}$ to the class $\hat{y} = \arg\max_k \ \boldsymbol{z}_k^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})$. And given a data set $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where $y_n \in [K]$ is the target class of $\boldsymbol{x}_n$, the optimal parameters $\boldsymbol{\mathcal{W}}$ are usually found by minimizing the empirical loss

$$\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)), \tag{2.3}$$

with respect to a specific loss function $\mathcal{L} : [K] \times \mathbb{R}^K \to \mathbb{R}_{\geq 0}$.

In the RO framework, however, we want to find the parameters $\boldsymbol{\mathcal{W}}$ by minimizing the worst case loss achieved over an uncertainty set of the input. More specifically, instead of optimizing the nominal loss in Eq. (2.3), we want to optimize the adversarial loss:

$$\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \max_{\boldsymbol{\delta} \in \mathcal{U}} \ \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\ \boldsymbol{x}_n + \boldsymbol{\delta})\right), \tag{2.4}$$

for some uncertainty set $\mathcal{U} \subset \mathbb{R}^M$.

Unfortunately, a closed-form expression for the inner maximization problem above is unknown and solving the min-max problem is notoriously difficult. RO provides multiple tools for solving such problems when the loss function is either convex or concave in the input variables. For example, in the case of concave loss functions, a common approach would be to take the dual of the maximization problem so that the problem can be formulated as a single minimization problem (Bertsimas and den Hertog 2022). If the loss function is instead convex, Fenchel's duality as well as conjugate functions can be used to find upper bounds and lower bounds of the maximization problem. However, there is no general framework developed for loss functions that do not fall into those categories, like in the case of neural networks.

In this paper, we will focus on the specific case in which the uncertainty set is the ball of radius ρ in the L_p space; i.e. $\mathcal{U} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \rho\}$, and we make the following assumptions about the loss function:

Assumption 2.3.1. *The loss function $\mathcal{L}$ is translationally invariant; i.e. for all $y \in [K], \boldsymbol{z} \in \mathbb{R}^K$, it satisfies*

$$\mathcal{L}(y, \boldsymbol{z}) = \mathcal{L}(y, \boldsymbol{z} - c\boldsymbol{e}) \quad \forall\, c \in \mathbb{R}, \tag{2.5}$$

where $\boldsymbol{e} \in \mathbb{R}^K$ denotes the vector with value 1 in all the coordinates.

Assumption 2.3.2. *The loss function $\mathcal{L}$ is monotonic; i.e. for all $y \in [K],\ \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^K$ it satisfies*

$$\Big(\forall k \in [K], \boldsymbol{z}_k - \boldsymbol{z}_y \leq \boldsymbol{z}'_k - \boldsymbol{z}'_y\Big) \implies \Big(\mathcal{L}(y, \boldsymbol{z}) \leq \mathcal{L}(y, \boldsymbol{z}')\Big). \tag{2.6}$$

Although some loss functions utilized in deep learning like the squared error or the absolute error do not satisfy these assumptions, the most popular losses for classification problems (softmax with cross-entropy loss, multiclass hinge loss, and hardmax with zero-one loss) do satisfy both of them. Intuitively, Assumption 2.3.1 implies that the loss function $\mathcal{L}$ takes into account the differences between the coordinates of $\boldsymbol{z}$ but not the exact value at each coordinate; while Assumption 2.3.2 means that a larger difference between the coordinates of the incorrect classes and the correct class results in a larger loss. These assumptions allow us to obtain the following result:

Lemma 2.3.3. *If the loss function $\mathcal{L}$ satisfies Assumptions 2.3.1 and 2.3.2, then for all $\boldsymbol{x} \in \mathbb{R}^M, y \in [K]$ the adversarial loss can be upper bounded as*

$$\min_{\boldsymbol{\mathcal{W}}} \max_{\boldsymbol{\delta}\in\mathcal{U}} \mathcal{L}(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta})) \tag{2.7}$$

$$\leq \min_{\boldsymbol{\mathcal{W}}} \mathcal{L}\bigg(y, \Big(\max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}^L_1(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta}) - \boldsymbol{z}^L_y(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta}), \dots, \max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}^L_K(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta}) - \boldsymbol{z}^L_y(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta})\Big)\bigg). \tag{2.8}$$

Proof. See Appendix A.1.1. □

The robustness methods we propose in the next sections generate upper bounds for the adversarial differences $\max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_k^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})$ for all $k \in [K]$, and then apply the previous lemma to upper bound the adversarial loss.

2.4 Approximate Robust Upper Bound for small $\boldsymbol{\rho}$

Perhaps the most intuitive approach to tackle problem (2.4) is to consider the first order approximation of the loss function

$$\mathcal{L}\left(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})\right) \approx \mathcal{L}\left(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\, \boldsymbol{x})\right) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{x}} \mathcal{L}\left(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\, \boldsymbol{x})\right),$$

since the right hand side is a linear function of $\boldsymbol{\delta}$ and the maximization problem can be more easily solved for linear functions of the uncertainty. For example, it is not hard to see that the first order approximation reaches its maximum value in $\mathcal{U} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \le \rho\}$ exactly at $\boldsymbol{\delta}^\star = \rho\, sign\left(\nabla_{\boldsymbol{x}} \mathcal{L}\big(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}\, \boldsymbol{x})\big)\right)$. This approach is referred to as *fast gradient sign method* and it was first explored in Goodfellow et al. (2015), where the networks are trained with adversarial examples generated as $\boldsymbol{x} + \boldsymbol{\delta}^\star$. A similar approach was proposed in Huang et al. (2016), where the authors considered the cross entropy loss and use the linear approximation of the softmax layer instead of the approximation of the entire loss function. In these methods, linear approximations are used to find near optimal perturbations that can produce strong adversarial examples for training, but not to approximate the adversarial loss.

An alternative to these methods would then be to train the network with the natural data and replace the loss function with its linear approximation, transforming the problem into

$$\begin{aligned}
&\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \max_{\boldsymbol{\delta}\in\mathcal{U}} \ \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)\right) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{x}} \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)\right) \\
=&\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)\right) + \rho \|\nabla_{\boldsymbol{x}} \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)\right)\|_q, \qquad (2.9)
\end{aligned}$$

where $\| \, \|_q$ is the dual norm of $\| \, \|_p$, satisfying $\frac{1}{p} + \frac{1}{q} = 1$. However, since the loss function is highly nonlinear, this approach (which we call Baseline-L_p) generally performs worse than training with adversarial examples (see the Baseline method in Section 5.7).

A more promising approach can be derived by noting that each component of the network $\boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})$ is in fact a continuous piecewise linear function (see the network definitions in Section 2.3), which suggests that the first order approximation of $\boldsymbol{z}^L$ is more precise than that of $\mathcal{L}(y, \boldsymbol{z}^L)$ for small neighborhoods. In fact, we expect the outputs $\boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})$ and $\boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})$ to be in the same linear piece when $\boldsymbol{x}+\boldsymbol{\delta}$ is close to $\boldsymbol{x}$. In other words, the linear approximation

$$\boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) \approx \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})^\top \boldsymbol{\delta} \tag{2.10}$$

is exact for small enough $\boldsymbol{\delta}$. We can then approximately solve the adversarial problem for each class k as

$$\begin{aligned}\max_{\boldsymbol{\delta} \in \mathcal{U}} \boldsymbol{z}_k^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) &\approx \max_{\boldsymbol{\delta} \in \mathcal{U}} (\boldsymbol{e}_k - \boldsymbol{e}_y)^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) + (\boldsymbol{e}_k - \boldsymbol{e}_y)^\top \nabla_x \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})^\top \boldsymbol{\delta} \\ &= (\boldsymbol{e}_k - \boldsymbol{e}_y)^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) + \rho \|(\boldsymbol{e}_k - \boldsymbol{e}_y)^\top \nabla_x \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})\|_q,\end{aligned} \tag{2.11}$$

where $k, y \in [K]$ and $\boldsymbol{e}_k$ (respectively $\boldsymbol{e}_y$) is the one-hot vector with a 1 in the k^{th} (respectively in the y^{th}) coordinate and 0, everywhere else. Applying the result from Lemma 2.3.3 and defining $\Delta \boldsymbol{e}_k^y \coloneqq \boldsymbol{e}_k - \boldsymbol{e}_y$ we obtain the approximate robust upper bound:

$$\begin{aligned}&\min_{\boldsymbol{\mathcal{W}}} \max_{\boldsymbol{\delta} \in \mathcal{U}} \mathcal{L}(y, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) \\ &\lessapprox \min_{\boldsymbol{\mathcal{W}}} \mathcal{L}\Big(y, \big(\Delta \boldsymbol{e}_1^{y\top} \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) + \rho \|\Delta \boldsymbol{e}_1^{y\top} \nabla_x \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})\|_q, \qquad (2.12) \\ &\qquad \ldots, \Delta \boldsymbol{e}_K^{y\top} \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}) + \rho \|\Delta \boldsymbol{e}_K^{y\top} \nabla_x \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})\|_q\big)\Big). \qquad (2.13)\end{aligned}$$

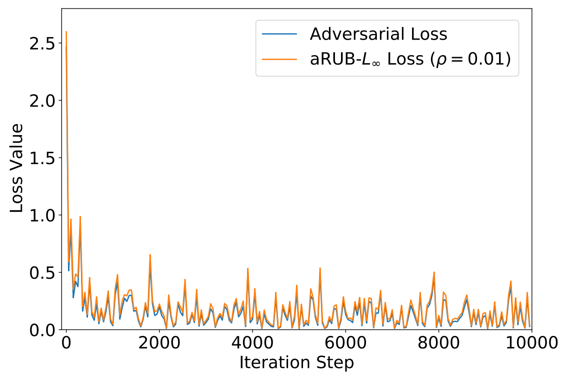
Table 2.1: Percentage of times when the aRUB approach yields an upper bound of the adversarial loss with respect to PGD attacks, i.e., percentage of times when Eq. (2.14) is larger than Eq. (2.4) evaluated using PGD-L_p attacks. For each row, aRUB-L_p and the PGD-L_p attacks use the L_p norm indicated on the first column. The loss function utilized is the cross entropy with softmax. Percentages are computed across all networks trained (ie., for all tested hyperparameters) in the 46 UCI data sets, every 500 training steps (see subsections 2.6.1 and 2.6.2 for more details about the networks and data sets). In this way, the approximate bound is evaluated in a large number of different conditions, including data sets, training steps and hyperparameters.

	$\boldsymbol{\rho} = \mathbf{0.0}$	**0.0008**	**0.001**	**0.0015**	**0.002**	**0.003**	**0.01**	**0.1**	**0.3**	**0.5**	**1.0**
L_∞	94%	99%	99%	99%	99%	99%	99%	95%	86%	81%	79%
L_1	93.3%	99%	99%	99%	99%	99%	99%	99%	99%	99%	98%

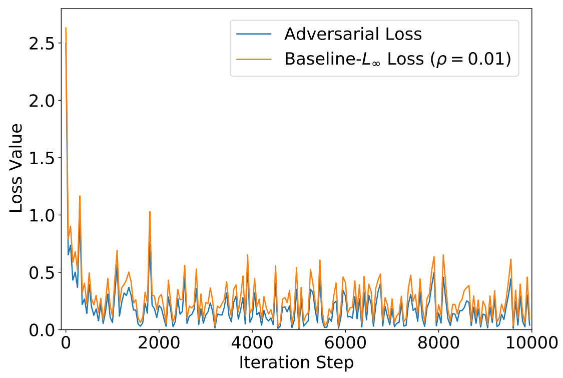
Therefore, we propose to train the network by minimizing Eq. (2.13) instead of the standard average loss, and we refer to this defense as aRUB-L_p. For the particular case of the cross entropy loss with softmax activation function in the output layer, the exact optimization problem to be solved would be the following:

$$\min_{\boldsymbol{\mathcal{W}}} \frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x}) + \rho \|(\Delta \boldsymbol{e}_k^{y_n})^\top \nabla_{\boldsymbol{x}} \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})\|_q}\right), \tag{2.14}$$

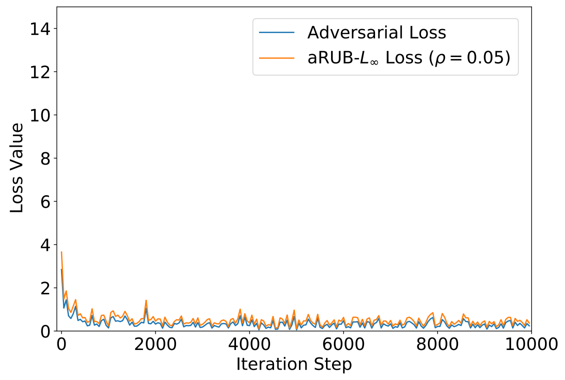
This expression may not always be an upper bound of the adversarial loss (Eq. (2.4)); however, we observe across a variety of experiments that usually it is indeed an upper bound (see Table 2.1). This suggests that the upper bound provided by Lemma 1 compensates for the errors introduced by the first order approximation of $\boldsymbol{z}^L$. Additionally, in Figure 2.1 we empirically show that Eq. (2.13) is much closer than Eq. (2.9) to the adversarial loss in Eq. (2.4) for small values of ρ.



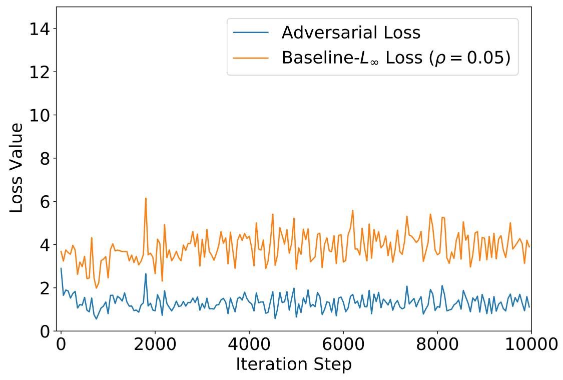
(a) aRUB-L_∞, $\rho = 0.01$



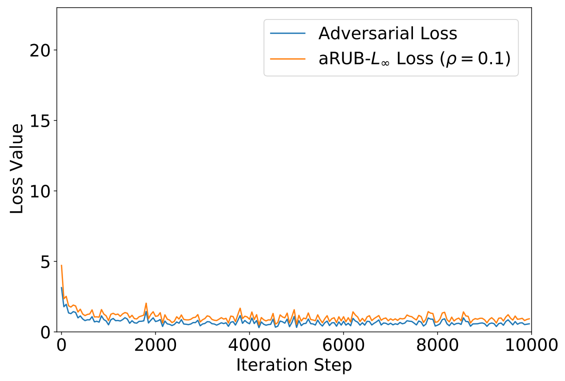
(b) Baseline-L_∞, $\rho = 0.01$



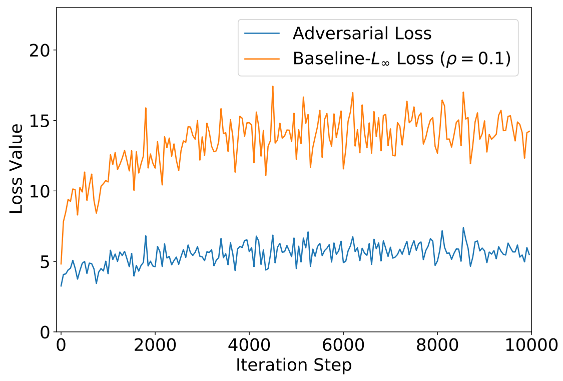
(c) aRUB-L_∞, $\rho = 0.05$



(d) Baseline-L_∞, $\rho = 0.05$



(e) aRUB-L_∞, $\rho = 0.1$



(f) Baseline-L_∞, $\rho = 0.1$

Figure 2.1: Adversarial loss (cross entropy loss evaluated at adversarial images bounded in L_∞ norm) vs the loss function being minimized across training iterations (aRUB-L_∞ on the left and Baseline-L_∞ on the right); the value of ρ increases for lower rows. Experiments are in the MNIST data set with the infinity norm ($p = \infty$). We use a feed forward neural network with three hidden layers with the softmax-cross-entropy loss (see subsection 2.6.1 for details). We used a learning rate of 10^{-3} and a batch size of 32, which we found to work best across the experiments in this figure.

2.5 Robust Upper bound for the $\boldsymbol{L_1}$ norm and general $\boldsymbol{\rho}$.

In this section we derive a provable upper bound for the robust counterpart of the inner maximization problem in Eq. (2.4) for the specific case in which the uncertainty set is the ball of radius ρ in the L$_1$ space; i.e. $\mathcal{U} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 \leq \rho\}$.

In the RO framework, finding a provable upper bound for a problem that is convex (or concave) in the uncertainty parameters relies on using convex (or concave) conjugate functions to make the problem linear in the uncertainty (Bertsimas and den Hertog 2022). The following lemmas are a generalization of this approach for the case in which the objective involves the composition of two functions, with the goal of making the problem linear not in the uncertainty but in the inner most function. Lemma 2.5.1 makes this generalization when the outer function is convex while Lemma 2.5.2 focuses on the concave case. The proofs of these lemmas rely on the definition of conjugate functions as well as the Fenchel duality theorem (Rockafellar 1970), and can be found in Appendix A.1. Together, these lemmas are the core of the methodology presented in this section.

Lemma 2.5.1. *If $f : A \to B$ is a convex and closed function then for any function $\boldsymbol{z} : \mathcal{U} \to A$, and any function $g : A \to B$ we have*

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} f(\boldsymbol{z}(\boldsymbol{\delta})) + g(\boldsymbol{z}(\boldsymbol{\delta})) = \sup_{\boldsymbol{u}\in dom(f^\star)} \; \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}(\boldsymbol{\delta})^T\boldsymbol{u} - f^\star(\boldsymbol{u}) + g(\boldsymbol{z}(\boldsymbol{\delta})),$$

where the convex conjugate function $f^\star$ is defined by $f^\star(\boldsymbol{z}) = \sup_{\boldsymbol{x}\in dom(f)} \boldsymbol{z}^T\boldsymbol{x} - f(\boldsymbol{x})$.

Proof. See Appendix A.1.2. □

Lemma 2.5.2. *Let $g : A \to B$ be a concave and closed function. If a function $\boldsymbol{z} : \mathcal{U} \to A$ satisfies $g(\boldsymbol{z}(\boldsymbol{\delta})) < \infty$ for all $\boldsymbol{\delta} \in \mathcal{U}$, then*

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} g(\boldsymbol{z}(\boldsymbol{\delta})) = \inf_{\boldsymbol{v}\in dom(g_\star)} \; \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}(\boldsymbol{\delta})^T\boldsymbol{v} - g_\star(\boldsymbol{v}),$$

where the concave conjugate function is defined by $g_\star(\boldsymbol{z}) = \inf_{\boldsymbol{x}\in dom(g)} \boldsymbol{z}^T\boldsymbol{x} - g(\boldsymbol{x})$.

Proof. See Appendix A.1.3. □

From the lemmas above we can observe that to apply them we will need to compute convex and concave conjugate functions. The next lemma facilitates these computations for neural networks with ReLU activation functions.

Lemma 2.5.3. *If* $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q} \geq \boldsymbol{0}$*, then the functions* $f(\boldsymbol{x}) = \boldsymbol{p}^\top[\boldsymbol{x}]^+$ *and* $g(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{u} - \boldsymbol{q}^\top[\boldsymbol{x}]^+$ *satisfy*

$$a)\ f^\star(\boldsymbol{z}) = \begin{cases} 0 & \text{if } \boldsymbol{0} \leq \boldsymbol{z} \leq \boldsymbol{p}, \\ \infty & \text{otherwise,} \end{cases} \quad \text{and} \quad b)\ g_\star(\boldsymbol{z}) = \begin{cases} 0 & \text{if } \boldsymbol{u} - \boldsymbol{q} \leq \boldsymbol{z} \leq \boldsymbol{u}, \\ -\infty & \text{otherwise.} \end{cases}$$

Proof. See Appendix A.1.4. □

As observed in Lemma 2.3.3, we can obtain an upper bound of the min-max problem in Eq. (2.4) by finding instead upper bounds for $\max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_k^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})$ for each class k. We will find these upper bounds in the following three steps:

- *Step #1 - Linearize the uncertainty:* We first make the maximization problem over the uncertainty set $\mathcal{U}$ linear in the first layer of the network (and therefore also linear in the uncertainty $\boldsymbol{\delta}$). Starting from the last layer we recursively split the objective function as the sum of a convex function and a concave function, and we then apply Lemmas 2.5.1 and 2.5.2 to make the maximization problem over $\mathcal{U}$ linear in the previous layer.

- *Step #2 - Optimize over the uncertainty set:* Then, we solve the maximization problem over $\mathcal{U}$. This problem can be exactly solved because the first layer of the network is linear in the uncertainty and the dual function of the L_1 norm is the L_∞ norm (a maximum over a finite set).

- *Step #3 - Backtrack:* Finally, we backtrack to solve for the variables $\boldsymbol{u}, \boldsymbol{v}$ that Lemmas 2.5.1 and 2.5.2 introduce.

For simplicity, we develop and prove the upper bound for the robust counterpart assuming that the neural network has only two layers; however, the results can be extended to the general case as shown in Appendix A.2. In addition, all the theorems can be generalized for residual and convolutional neural networks, since convolutions are a special case of matrix multiplication.

Step #1 - Linearize the uncertainty

The following theorem shows how the maximization problem over $\mathcal{U}$ can be transformed from a linear problem in the second layer to a linear problem in the first layer of the network. The proof relies on Lemma 2.5.1 and Lemma 2.5.2.

Theorem 2.5.4. *The maximum difference between the output of the correct class and the output of any other class k can be written as*

$$\begin{aligned}\sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_k^2(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^2(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) = \quad & \sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) && (2.15)\\ = & \sup_{0\le\boldsymbol{s}\le 1}\ \inf_{0\le\boldsymbol{t}\le 1}\ \sup_{\boldsymbol{\delta}\in\mathcal{U}} (\boldsymbol{p}-\boldsymbol{q})^\top \boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) + (\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{b}^2 \\ & \quad s.t. \quad \boldsymbol{p} = [(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s} \\ & \qquad\qquad \boldsymbol{q} = [-(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}, \\ & && (2.16)\end{aligned}$$

where $\odot$ corresponds to entry-wise multiplication.

Proof. By definition of the two layer neural network $\boldsymbol{z}^2$, we have

$$\begin{aligned}(\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) &= (\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{W}^2[\boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta})]^+ + (\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{b}^2 \\ &= f_+(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta})) - f_-(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta})) + (\Delta\boldsymbol{e}_k^y)^\top \boldsymbol{b}^2,\end{aligned}$$

where f_+, f_- are the convex functions defined by

$$f_+(\boldsymbol{x}) = [(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^2]^+[\boldsymbol{x}]^+, \quad \text{and} \quad f_-(\boldsymbol{x}) = [-(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^2]^+[\boldsymbol{x}]^+.$$

Applying Lemma 2.5.1 to the function f_+ we then have

$$\begin{aligned}
&\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}), \\
=&\sup_{\boldsymbol{\delta}\in\mathcal{U}} f_+(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) - f_-(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2, \\
=& \sup_{\boldsymbol{u}\in\mathrm{dom}(f_+^\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{u}^\top \boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - f_+^\star(\boldsymbol{u}) - f_-(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2, \quad (2.17)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(By Lemma 2.5.1)}\\
=& \sup_{\boldsymbol{u}\in\mathrm{dom}(f_+^\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{u}^\top \boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - f_-(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2. \quad (2.18)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(By Lemma 2.5.3a)}
\end{aligned}$$

Defining the concave function $g(\boldsymbol{x}) = \boldsymbol{u}^\top \boldsymbol{x} - f_-(\boldsymbol{x})$, and applying Lemma 2.5.2 to the function g we obtain

$$\begin{aligned}
&\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) \\
=& \sup_{\boldsymbol{u}\in\mathrm{dom}(f_+^\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} g(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta})) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2 \quad (2.19)\\
=& \sup_{\boldsymbol{u}\in\mathrm{dom}(f_+^\star)} \inf_{\boldsymbol{v}\in\mathrm{dom}(g_\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{v}^\top \boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) - g_\star(\boldsymbol{v}) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2 \qquad \text{(By Lemma 2.5.2)}, \quad (2.20)\\
=& \sup_{\boldsymbol{u}\in\mathrm{dom}(f_+^\star)} \inf_{\boldsymbol{v}\in\mathrm{dom}(g_\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{v}^\top \boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}+\boldsymbol{\delta}) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2 \qquad \text{(By Lemma 2.5.3b)}. \quad (2.21)
\end{aligned}$$

Lastly, by Lemma 2.5.3a and 2.5.3b, we know that the variables $\boldsymbol{u}$ and $\boldsymbol{v}$ can be parameterized as

$$\begin{aligned}\boldsymbol{u} &= [(\boldsymbol{W}^2)^\top(\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s},\\ \boldsymbol{v} &= [(\boldsymbol{W}^2)^\top(\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s} - [-(\boldsymbol{W}^2)^\top(\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}\end{aligned}$$

with $0 \leq \boldsymbol{s}, \boldsymbol{t} \leq 1$. Substituting these values in Eq. (2.21) we obtain Eq. (2.16), as desired. □

Step #2 - Optimize over the uncertainty set

Notice that the objective in Eq. (2.16) is linear in $\boldsymbol{z}^1$ and therefore it is also linear in $\boldsymbol{\delta}$, which facilitates the computation of the exact value of the supremum over $\mathcal{U}$, as shown in the next corollary.

Corollary 2.5.5. *If* $\mathcal{U} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \rho\}$, *then:*

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta}) \tag{2.22}$$

$$\begin{aligned}= \sup_{0\leq \boldsymbol{s}\leq 1} \inf_{0\leq \boldsymbol{t}\leq 1} \; & \rho\|(\boldsymbol{p}-\boldsymbol{q})^\top \boldsymbol{W}^1\|_q + (\boldsymbol{p}-\boldsymbol{q})^\top(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^2\\ s.t. \quad & \boldsymbol{p} = [(\boldsymbol{W}^2)^\top(\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s}\\ & \boldsymbol{q} = [-(\boldsymbol{W}^2)^\top(\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t},\end{aligned} \tag{2.23}$$

where $\|\cdot\|_q$ *is the dual norm of* $\|\cdot\|_p$, *with* $\frac{1}{p} + \frac{1}{q} = 1$.

Before proceeding to the proof of the corollary, notice that we can recover the approximation method developed in the previous section by setting

$$\boldsymbol{s} = \boldsymbol{t} = [sign(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}))]^+ \tag{2.24}$$

in the objective of problem (2.23) to obtain

$$\rho\|((\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}_k^y)\odot[sign(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x}))]^+)^\top\boldsymbol{W}^1\|_q+(\Delta\boldsymbol{e}_k^y)^\top\boldsymbol{W}^2[\boldsymbol{z}^1(\boldsymbol{\mathcal{W}},\boldsymbol{x})]^++(\Delta\boldsymbol{e}_k^y)^\top\boldsymbol{b}^2, \tag{2.25}$$

which is the same as the linear approximation of $(\Delta\boldsymbol{e}_k^y)^\top\boldsymbol{z}^2(\boldsymbol{\mathcal{W}}+\boldsymbol{\delta})$ obtained in Eq. (2.11).

Proof. The proof follows directly after applying Theorem 2.5.4 and using the fact that for all vectors $\boldsymbol{c}$ we have

$$\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_p\le\rho}\boldsymbol{c}^\top\boldsymbol{\delta}=\rho\|\boldsymbol{c}\|_q. \tag{2.26}$$

□

Step #3 - Backtrack

Since neural networks are trained by minimizing the empirical loss over the parameters $\boldsymbol{\mathcal{W}}$, we want to avoid the computation of supremums in the objective. While the previous corollary shows how to solve the supremum over the uncertainty set, a new supremum was introduced in Theorem 2.5.4 over the variables $\boldsymbol{s}$. The next theorem tells us how we can remove this new supremums for the specific case $p=1$.

Theorem 2.5.6. *The maximum difference between the output of the correct class and the output of any other class k can be upper bounded by*

$$\begin{aligned}&\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_1\le\rho}(\Delta\boldsymbol{e}_k^y)^\top\boldsymbol{z}^2(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta})\\&\qquad\le\inf_{0\le\boldsymbol{t}\le1}\max_{m\in[M]}\max\left\{g_{k,m}^2(\boldsymbol{\mathcal{W}},\boldsymbol{t},\boldsymbol{x},\rho),g_{k,m}^2(\boldsymbol{\mathcal{W}},\boldsymbol{t},\boldsymbol{x},-\rho)\right\},\end{aligned} \tag{2.27}$$

where the new network g is defined by the equations

$$g^1_m(\boldsymbol{\mathcal{W}}, a) = a\boldsymbol{W}^1\boldsymbol{e}_m + \boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1,$$
$$g^2_{k,m}(\boldsymbol{\mathcal{W}}, \boldsymbol{t}, \boldsymbol{x}, a) = [(\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{W}^2]^+[g^1_m(\boldsymbol{\mathcal{W}}, a)]^+ - [-(\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{W}^2]^+[g^1_m(\boldsymbol{\mathcal{W}}, a)] \odot \boldsymbol{t} + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2,$$

for $a = \rho, -\rho$.

Proof. Applying Corollary 2.5.5 with $p = 1$ and using the min-max inequality we obtain

$$\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_1 \le \rho} (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta}) \tag{2.28}$$
$$\le \inf_{0\le \boldsymbol{t} \le 1} \sup_{0 \le \boldsymbol{s} \le 1} \rho\|(\boldsymbol{p} - \boldsymbol{q})^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p} - \boldsymbol{q})^\top(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2$$
$$\text{s.t.} \quad \boldsymbol{p} = [(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}^y_k)]^+ \odot \boldsymbol{s} \tag{2.29}$$
$$\boldsymbol{q} = [-(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}^y_k)]^+ \odot \boldsymbol{t}.$$

Defining $\boldsymbol{p}(\boldsymbol{s}) = [(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}^y_k)]^+ \odot \boldsymbol{s}$ and $\boldsymbol{q}(\boldsymbol{t}) = [-(\boldsymbol{W}^2)^\top(\Delta\boldsymbol{e}^y_k)]^+ \odot \boldsymbol{t}$, we have that for fixed $\boldsymbol{t}$ it holds

$$\begin{aligned}
&\sup_{0\le \boldsymbol{s} \le 1} \rho\|(\boldsymbol{p}(\boldsymbol{s}) - \boldsymbol{q}(\boldsymbol{t}))^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p}(\boldsymbol{s}) - \boldsymbol{q}(\boldsymbol{t}))^\top(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2 \\
&= \max_{m\in[M]} \max\Bigg\{ \sup_{0\le \boldsymbol{s} \le 1} (\boldsymbol{p}(\boldsymbol{s}) - \boldsymbol{q}(\boldsymbol{t}))^\top(\boldsymbol{W}^1(\boldsymbol{x} + \rho\boldsymbol{e}_m) + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2, \\
&\qquad\qquad \sup_{0\le \boldsymbol{s} \le 1} (\boldsymbol{p}(\boldsymbol{s}) - \boldsymbol{q}(\boldsymbol{t}))^\top(\boldsymbol{W}^1(\boldsymbol{x} - \rho\boldsymbol{e}_m) + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2 \Bigg\} \\
&= \max_{m\in[M]} \max\Big\{[(\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{W}^2]^+[\boldsymbol{W}^1(\boldsymbol{x} + \rho\boldsymbol{e}_m) + \boldsymbol{b}^1]^+ - \boldsymbol{q}(\boldsymbol{t})^\top(\boldsymbol{W}^1(\boldsymbol{x} + \rho\boldsymbol{e}_m) + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2, \\
&\qquad\qquad [(\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{W}^2]^+[\boldsymbol{W}^1(\boldsymbol{x} - \rho\boldsymbol{e}_m) + \boldsymbol{b}^1]^+ - \boldsymbol{q}(\boldsymbol{t})^\top(\boldsymbol{W}^1(\boldsymbol{x} - \rho\boldsymbol{e}_m) + \boldsymbol{b}^1) + (\Delta\boldsymbol{e}^y_k)^\top \boldsymbol{b}^2 \Big\} \\
&= \max_{m\in[M]} \max\{g^2_{k,m}(\boldsymbol{\mathcal{W}}, \boldsymbol{t}, \boldsymbol{x}, \rho), g^2_{k,m}(\boldsymbol{\mathcal{W}}, \boldsymbol{t}, \boldsymbol{x}, -\rho)\}.
\end{aligned}$$

The theorem then follows after applying the inf over $0 \leq \boldsymbol{t} \leq 1$. $\square$

Notice that in the previous proof it was important to use $p = 1$, since the dual of the L_1 norm is the L_∞ norm, which can be written as a maximum over a finite set. With a different p, the solution for the variables $\boldsymbol{s}$ would be more challenging to find. However, for the chosen uncertainty set we obtain an upper bound of Eq. (2.4) by applying the result from the previous Theorem to Lemma 2.3.3. While we could include the variables $\boldsymbol{t}$ in the minimization problem over $\boldsymbol{\mathcal{W}}$, we instead use fixed values $\boldsymbol{t} = [sign(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}))]^+$ based on the linear approximation of $(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^2(\boldsymbol{\mathcal{W}}, \boldsymbol{x} + \boldsymbol{\delta})$, as described in Eq. (2.24). Notice that setting specific values for $\boldsymbol{t}$ does not affect the inequalities: since the upper bound includes the infimum over $\boldsymbol{t}$, any $0 \leq \boldsymbol{t} \leq 1$ yields an upper bound of the robust problem. For the specific case of the cross entropy loss function, the proposed upper bound for the min-max robust problem is

$$\min_{\boldsymbol{\mathcal{W}}} \frac{1}{N} \sum_{n=1}^{N} \log \left(\sum_k e^{\left(\max_{m \in [M]} \max\{ g^2_{k,m}(\boldsymbol{\mathcal{W}}, [sign(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}))]^+, \boldsymbol{x}, \rho), g^2_{k,m}(\boldsymbol{\mathcal{W}}, [sign(\boldsymbol{z}^1(\boldsymbol{\mathcal{W}}, \boldsymbol{x}))]^+, \boldsymbol{x}, -\rho) \} \right)} \right). \tag{2.30}$$

2.6 Experiments

In this section, we demonstrate the effectiveness of the proposed methods in practice. We first introduce the experimental setup and then we compare the robustness of several defenses.

2.6.1 Experimental details

Data sets. We use 46 data sets from the UCI collection (Dua and Graff 2017), which correspond to classification tasks with a diverse number of features that are not categorical. For each data set we do a 80%/20% split for training/testing sets, and we further reserve 25% of each training set for validation. In addition, we use three popular computer vision data sets, namely the MNIST (Deng 2012a), Fashion MNIST (Xiao et al. 2017a) and CIFAR (Krizhevsky et al. 2009a) data sets.

Pre-processing. All input data has been previously scaled, which facilitates the comparison of the adversarial attacks across data sets. For the UCI data sets, each feature is standarized using the statistics of the training set, while for the vision data sets each image channel is normalized to be between 0 and 1, or standarized, depending on what leads to best robustness.

Attacks. We use the implementation provided by the foolbox library (Rauber et al. 2017, 2020) using the default parameters. We evaluate attacks using projected gradient descent and fast gradient methods. More specifically, we use the following adversarial attacks:

- PGD-L_p: Attack bounded in L_p norm and found using Projected Gradient Descent.
- FGM-L_p: Attack bounded in L_p norm and found using Fast Gradient Method for $p = 1, 2$ and Fast Gradient Sign Method for $p = \infty$.

Defenses. Our comparisons include different defenses denoted as follows:

- aRUB-L_p: Approximate Robust Upper Bound method described in section 2.4 using the L_1 or L_∞ sphere as the uncertainty set.
- RUB: Robust Upper Bound method described in section 2.5 using the L_1 sphere as the uncertainty set.
- PGD-L_∞: Adversarial training method in which the network is trained using attacks that are bounded in the L_∞ norm and found using Projected Gradient Descent.
- Baseline-L_∞: Simple approximation method resulting from minimizing Eq. (2.9) using the L_p sphere as the uncertainty set.
- Nominal: Standard vanilla training with no robustness ($\rho = 0$).

Architecture. We evaluate a neural network with three dense hidden layers with 200 neurons in each hidden layer. For the vision data sets, we also provide results with Convolutional Neural Networks (*CNNs*) in Appendix A.3. The architecture has two convolutional layers alternated with pooling operations, and two dense layers, as in Madry et al. (2019). The parameters of the networks were initialized with the Glorot initialization (Glorot and Bengio 2010a).

Hyperparameter Tuning. Each network and defense is trained for different learning rates ($\{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$). For the UCI data set we use a batch size of 256 and for the vision data sets we try a batch size of 32 and 256. For the L_∞ based training methods we try all values of ρ from the set ($\{10^{-4}, 10^{-5}, 10^{-3}, 10^{-2}, 0.1, 0.3, 0.5, 1, 3, 5, 10\}$). For the methods based on the L_1 norm, we scale those values of ρ by a factor of $\sqrt{m}$, since $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2$ and $\|\boldsymbol{x}\|_1 \leq \sqrt{m}\|\boldsymbol{x}\|_2$ for any $\boldsymbol{x} \in \mathbb{R}^m$. In this way we ensure that the L_1 spheres and the L_∞ spheres contain the same L_2 spheres, allowing for a fair comparison of all methods in terms of adversarial attacks that are bounded in the L_2 norm. All networks trained using the UCI data sets are trained for 5000 iterations, and all vision data sets are trained for 10000 iterations. For each network, data set, batch size, and defense radius ρ, we have verified that for at least one of the learning rates the validation accuracy converges with the aforementioned number of training iterations. Finally, for each attack type with radius ρ, we select on the validation set the best hyperparameters for each defense, i.e., given a data set, an attack type and its radius ρ, the hyperparameters of a defense (network, learning rate, batch size, normalization and defense radius ρ) are the ones that lead to the highest adversarial robustness in the validation set. In total we trained more than $40,000$ networks across all tested data sets and defenses.

2.6.2 UCI data sets

We run experiments on the 46 UCI data sets using different methods for robust training and compare the adversarial accuracies achieved with multiple types of adversarial attacks. For each data set we rank every training method, where the method with rank 1 corresponds to the one with highest adversarial accuracy. The average ranks and the corresponding 95% confidence intervals are shown in Figure 2.2, where we observe a similar pattern across all types of attacks, namely, we see that the best ranks are achieved with aRUB-L_∞ and PGD-L_∞ when ρ is smaller than 10^{-1}; next there is a small range in which PGD-L_∞ does best and finally for larger values of ρ the best rank is that of RUB. In addition, for large values of ρ we observe better results with Baseline-L_∞ than with aRUB-L_∞, suggesting that the linear approximation of the network becomes inaccurate and leads to a large change in the loss function. We also highlight that looking at $\rho = 0$, it is clear that robust training methods achieve better natural accuracy than the Nominal training method.

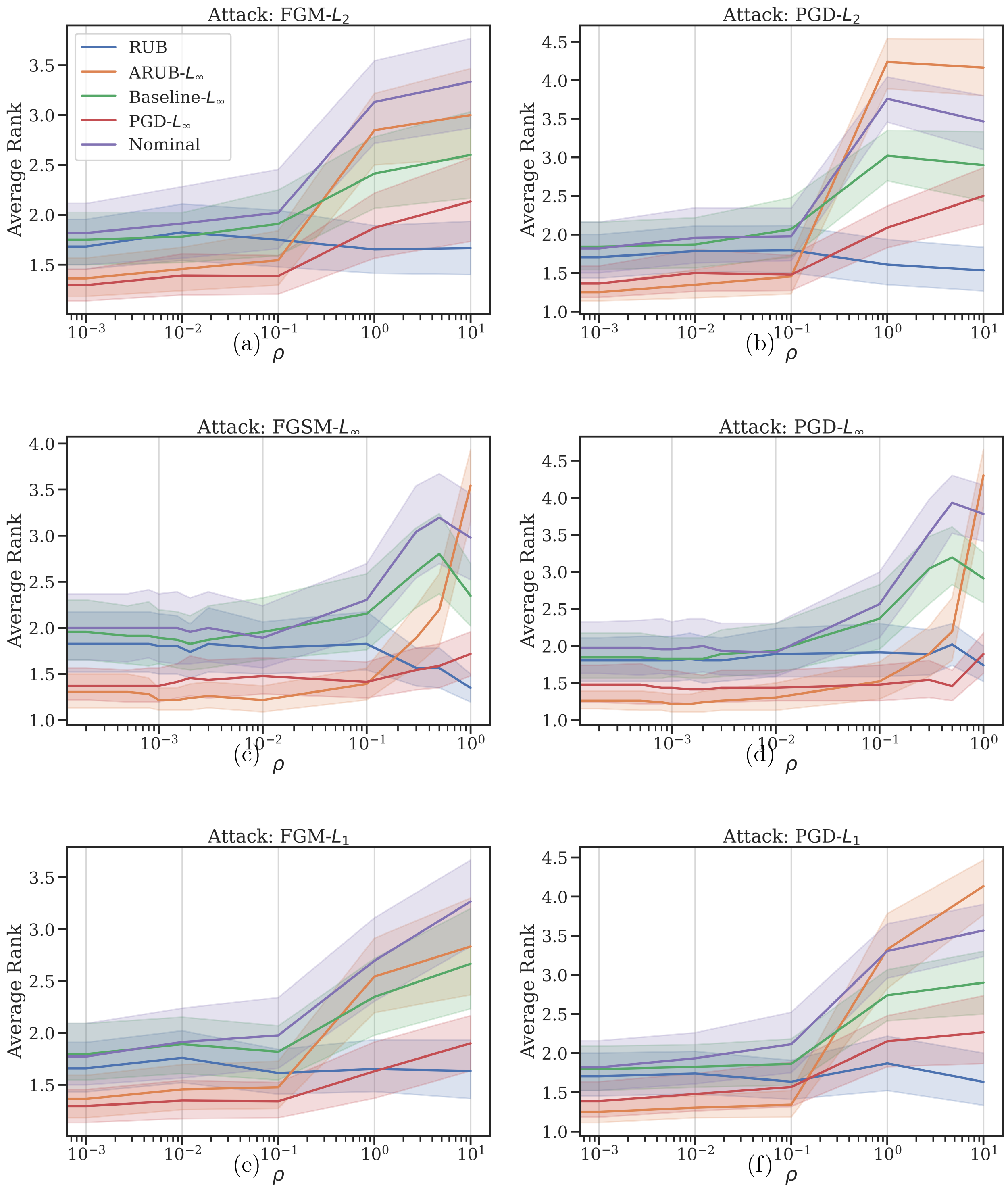
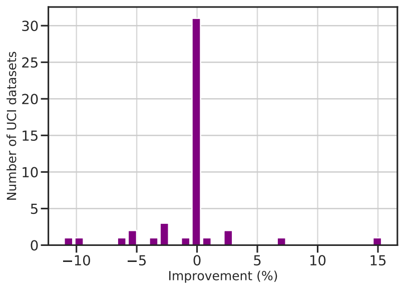
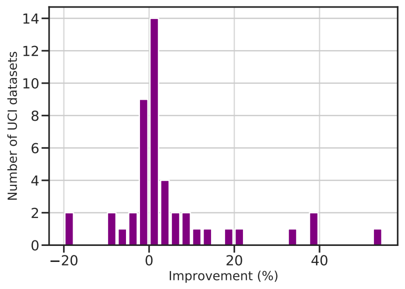


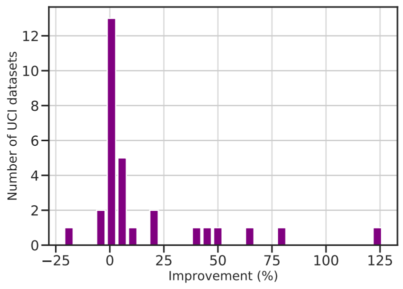
Figure 2.2: Average rank and the corresponding 95% confidence interval for each method across the 46 UCI data sets for adversarial attacks bounded in L_2, L_∞ and L_1 norm, respectively from top to bottom. The figures on the left use attacks based on Fast Gradient methods, and the figures on the right use instead attacks based on Projected Gradient Descent.



(a) $\rho = 0.01$

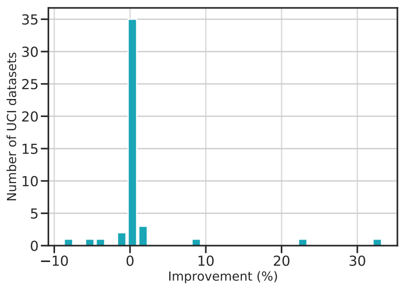


(b) $\rho = 1$

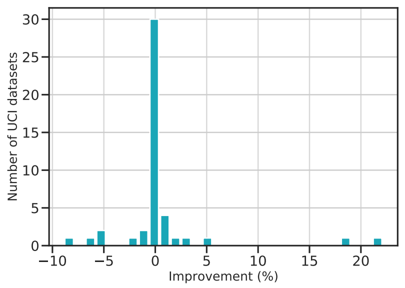


(c) $\rho = 10$

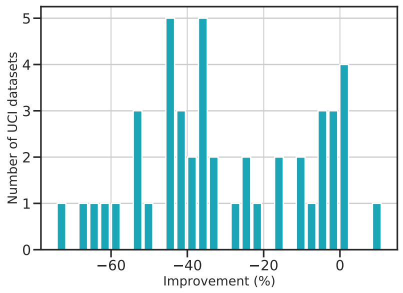
Figure 2.3: Number of UCI data sets for which RUB-L_1 improves adversarial accuracy over PGD-L_∞ by a specific percentage. Figures a), b) and c) show the corresponding plots for PGD-L_2 adversarial examples with different values of ρ.



(a) $\rho = 0.001$



(b) $\rho = 0.1$



(c) $\rho = 1$

Figure 2.4: Number of UCI data sets for which aRUB-L_∞ improves adversarial accuracy over PGD-L_∞ by a specific percentage. Figures a), b) and c) show the corresponding plots for PGD-L_∞ adversarial examples with different values of ρ.

To better compare the performances of RUB and aRUB against PGD-L_∞, we also analyze the percentage by which each method improves adversarial accuracy across data sets. In Figure 2.3 we show the number of data sets for which RUB improves L_2 adversarial accuracy over PGD-L_∞ by a specific percentage. We observe that the improvement becomes larger as ρ increases, and in particular, for $\rho = 10$ we observe that RUB only lowers adversarial accuracy for 3 data sets, while it shows more than 15% improvement over PGD-L_∞ for 8 of the data sets. Similarly, Figure 2.4 displays the number of data sets for which aRUB-L_∞ improves adversarial accuracy over PGD-L_∞ by some percentage; we observe that for small perturbations ($\rho = 0.001, \rho = 0.1$) aRUB seems to slightly improve over PGD-L_∞, while this last defense has a clear advantage for larger perturbations ($\rho = 1$).

Lastly, in Table 2.2 we display the average number of batches processed per second as well as the corresponding standard deviation for each method across the 46 UCI data sets. As expected, we see that Nominal is the method that processes the largest number of batches per second, and all defense methods except Baseline-L_∞ are much slower than Nominal. However, RUB, RUB-L_1 and aRUB-L_∞ all process more batches per second compared to PGD-L_∞.

Table 2.2: Average number of batches processed per second across the 46 UCI data sets, as well as the corresponding standard deviations.

	Avg no. batches per second	**Standard Deviation**
RUB	65.1	12.6
aRUB -L_1	17.5	0.5
aRUB -L_∞	18.8	0.4
Baseline-L_∞	465	56.4
PGD -L_∞	4.5	0.1
Nominal	712.6	87.8

2.6.3 Vision Data Sets

We next show experiment results for the three vision data sets. Specifically, we compare for different training methods their performance against adversarial attacks as well as the security guarantees obtained from applying the upper bound from Eq. (2.30). Since the proposed RUB method significantly increases memory requirements for inputs with large dimensions, we compare this method against other defenses using the feed forward architecture with three hidden layers. We do include results using a CNN architecture for all other methods in Appendix A.3, where we also explain how to extend the theory of the RUB method for convolutional layers with ReLU and MaxPool activation functions.

Performance Against Adversarial Attacks. We evaluate adversarial accuracy for all the aforementioned methods (Nominal, RUB, aRUB-L_1, aRUB-L_∞, Baseline-L_∞, PGD-L_∞), and we add two other state-of-the-art defenses to make our evaluation even more comprehensive. These two methods were proposed in Wong and Kolter (2018) and Wong et al. (2020); which we call COAP-L_∞ (Convex Outer Adversarial Polytope with L_∞ norm) and FGSM-L_∞ (Fast Gradient Sign Method) respectively, and are representative of state-of-the-art defenses in terms of robustness and training computational cost, respectively. For each L_p norm we report the minimum adversarial accuracy achieved using both Projected Gradient Descent attacks and Fast Gradient Method attacks.

We observe that for the Fashion MNIST data set (Table 2.3), aRUB-L_1 and RUB achieve the best accuracies for small values of ρ; we then observe a small range in which PGD-L_∞ does best and lastly for larger values of ρ we see that RUB takes the lead, which is similar to the average results observed for the UCI data sets. For the MNIST data set, all defenses achieve similar results when the input perturbations are small, with RUB showing again better performance with larger radius ρ. In the CIFAR data set we observe a different behavior; PGD-L_∞, FGSM-L_∞ and aRUB methods achieve better accuracies at various radius regimes, although we observe that all methods perform very poorly overall as this is a notoriously more difficult data set.

Lastly, we again observe that in all three data sets Baseline-L_∞ outperforms aRUB-L_∞ when ρ is large, and the robust training methods achieve a higher natural accuracy than the accuracy resulting from standard nominal training.

Table 2.3: Adversarial Accuracy (%) for Fashion MNIST. For each choice of L_p norm, defense method and noise radius ρ, we report the minimum accuracy achieved with PGD and FGM attacks. Colored cells correspond to accuracies that are within 0.5 percentage units of the best result in each column, which has bold font.

	$\rho =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00	56.00
L_2 Attacks	RUB	89.88	89.22	88.95	**86.80**	60.51	**55.47**	**55.47**	**55.27**	**49.41**
	aRUB-L_1	**90.31**	**90.04**	**89.45**	85.98	63.98	29.10	18.24	15.04	9.96
	aRUB-L_∞	89.18	89.06	88.75	85.98	65.43	24.18	18.24	16.02	9.96
	Baseline-L_∞	89.38	89.02	88.32	85.04	48.20	29.88	28.71	26.52	18.55
	PGD-L_∞	89.61	89.38	88.01	85.23	**68.20**	31.60	25.90	22.85	19.10
	FGSM-L_∞	89.41	87.81	87.19	84.06	67.81	35.23	27.11	25.39	19.06
	COAP-L_∞	89.14	88.48	87.11	82.58	30.51	20.94	19.69	19.69	19.69
	Nominal	87.70	88.40	88.01	85.35	46.60	19.92	16.84	15.39	15.39

	$\rho =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00	140.00
L_1 Attacks	RUB	**89.84**	**89.80**	**89.73**	**89.41**	**87.93**	84.18	81.72	75.70	**55.51**
	aRUB-L_1	89.02	89.02	88.98	88.71	87.34	**84.57**	**82.34**	**75.86**	41.21
	aRUB-L_∞	89.02	89.02	89.02	88.79	87.03	82.85	79.77	71.25	30.63
	Baseline-L_∞	88.83	88.20	88.20	87.85	85.74	82.11	78.09	68.28	30.59
	PGD-L_∞	89.02	89.02	88.95	88.95	87.11	**84.57**	81.37	73.16	39.69
	FGSM-L_∞	89.80	89.77	**89.73**	89.26	85.00	82.19	78.44	73.36	35.62
	COAP-L_∞	89.22	89.22	89.22	88.67	82.58	73.01	60.47	48.59	22.19
	Nominal	86.99	86.95	86.91	86.91	85.00	82.54	78.83	68.98	21.88

	$\rho =$	0.000	0.001	0.003	0.010	0.100	0.300	0.500	1.00	5.00
L_∞ Attacks	RUB	89.38	89.18	88.52	86.45	65.08	56.17	56.13	**55.78**	**37.77**
	aRUB-L_1	**90.00**	**89.73**	**89.06**	86.05	65.82	28.95	11.02	10.00	10.00
	aRUB-L_∞	89.77	89.22	88.71	86.68	79.80	67.93	52.81	12.15	10.00
	Baseline-L_∞	89.14	88.63	86.99	85.55	56.80	30.20	29.06	27.19	19.06
	PGD-L_∞	89.02	89.02	88.24	**87.77**	80.43	**73.09**	**65.74**	31.21	18.52
	FGSM-L_∞	89.61	89.10	87.97	86.91	**80.55**	62.93	53.16	25.74	18.12
	COAP-L_∞	89.18	88.24	86.76	85.66	67.15	23.87	17.19	17.19	16.21
	Nominal	88.12	87.93	87.03	85.27	58.32	21.80	18.79	16.37	10.94

Table 2.4: Adversarial Accuracy (%) for MNIST. For each choice of L_p norm, defense method and noise radius ρ, we report the minimum accuracy achieved with PGD and FGM attacks. Colored cells correspond to accuracies that are within 0.5 percentage units of the best result in each column, which has bold font.

	$\rho =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00	56.00
L_2 Attacks	RUB	97.93	97.85	97.07	97.07	79.30	**66.48**	62.89	**55.00**	**38.20**
	aRUB-L_1	**98.63**	**98.52**	97.73	97.46	80.82	57.58	55.04	46.60	29.96
	aRUB-L_∞	97.97	97.97	97.81	97.81	91.17	51.80	49.53	43.40	31.13
	Baseline-L_∞	97.58	97.50	97.50	97.11	72.85	64.18	61.29	52.81	34.80
	PGD-L_∞	98.24	98.24	**98.24**	**98.09**	92.70	66.05	62.46	54.26	35.94
	FGSM-L_∞	97.81	97.81	97.50	97.50	**93.01**	64.96	**63.20**	53.40	36.52
	COAP-L_∞	98.01	97.97	97.54	94.88	54.73	31.29	31.29	28.98	28.48
	Nominal	97.62	97.46	97.34	96.41	69.61	19.57	15.90	15.35	11.25

	$\rho =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00	140.00
L_1 Attacks	RUB	98.01	97.97	97.93	97.89	97.03	97.03	96.25	91.84	**66.95**
	aRUB-L_1	98.28	98.28	98.28	98.28	97.89	97.46	96.52	94.49	58.20
	aRUB-L_∞	98.40	98.40	98.40	98.24	**98.24**	**97.58**	**96.91**	93.16	51.05
	Baseline-L_∞	98.05	98.05	98.05	98.01	97.58	96.52	95.47	90.16	63.83
	PGD-L_∞	98.20	98.20	98.20	98.20	98.20	97.50	96.68	93.87	66.84
	FGSM-L_∞	**98.44**	**98.44**	**98.44**	**98.44**	97.19	97.19	95.00	**94.53**	64.02
	COAP-L_∞	97.81	97.81	97.81	97.81	96.99	92.50	83.12	69.30	30.47
	Nominal	97.58	97.58	97.58	97.54	97.30	95.94	94.49	89.96	19.38

	$\rho =$	0.000	0.001	0.003	0.010	0.100	0.300	0.500	1.00	5.00
L_∞ Attacks	RUB	98.44	98.32	97.93	97.70	86.09	68.52	67.15	61.56	**27.15**
	aRUB-L_1	98.52	98.36	98.28	98.09	86.84	61.88	59.61	54.77	21.64
	aRUB-L_∞	**98.71**	**98.59**	98.36	98.36	96.95	89.96	78.16	49.69	23.24
	Baseline-L_∞	98.24	97.81	97.81	97.66	84.69	65.08	63.24	59.49	24.73
	PGD-L_∞	98.20	98.20	98.20	98.20	96.99	**93.16**	**86.80**	**64.57**	25.35
	FGSM-L_∞	98.59	**98.59**	**98.59**	**98.59**	**97.50**	91.17	74.10	63.12	26.36
	COAP-L_∞	98.12	98.12	98.12	96.56	90.78	37.81	31.72	32.62	24.88
	Nominal	97.62	97.62	97.46	96.80	81.91	23.28	16.48	14.61	9.69

Table 2.5: Adversarial Accuracy (%) for CIFAR. For each choice of L_p norm, defense method and noise radius ρ, we report the minimum accuracy achieved with PGD and FGM attacks. Colored cells correspond to accuracies that are within 0.5 percentage units of the best result in each column, which has bold font.

	$\boldsymbol{\rho} =$	0.00	0.01	0.03	0.06	0.08	0.11	0.17	0.55	5.54
L_2 Attacks	RUB	49.69	48.67	46.33	48.59	48.36	48.12	47.58	44.14	17.73
	aRUB-L_1	52.42	51.41	51.37	**51.13**	**51.02**	**50.74**	**50.20**	**46.56**	21.88
	aRUB-L_∞	53.83	52.89	51.84	47.77	47.11	46.48	44.45	41.37	20.66
	Baseline-L_∞	53.12	52.07	50.66	45.82	42.50	42.97	42.42	35.86	9.30
	PGD-L_∞	**53.91**	**53.32**	**52.27**	48.83	48.63	48.48	48.09	45.12	20.16
	FGSM-L_∞	53.52	52.07	50.82	49.34	47.97	47.30	47.30	44.53	**24.69**
	COAP-L_∞	51.45	50.86	49.45	48.59	47.07	45.35	41.76	35.86	11.76
	Nominal	46.02	45.78	44.84	44.10	42.89	42.15	40.43	37.11	10.59

	$\boldsymbol{\rho} =$	0.00	0.01	0.03	0.06	0.11	0.55	5.54	16.63	27.71
L_1 Attacks	RUB	50.86	50.86	50.86	50.70	50.55	48.98	47.03	44.84	42.58
	aRUB-L_1	51.56	51.56	51.56	51.56	51.56	51.37	**50.35**	**47.85**	**45.98**
	aRUB-L_∞	53.55	53.48	53.44	53.40	53.24	52.30	44.02	40.59	40.62
	Baseline-L_∞	53.32	53.28	53.28	53.24	52.03	50.94	41.72	38.79	34.53
	PGD-L_∞	52.97	52.97	52.97	52.97	52.93	52.30	47.89	45.62	43.52
	FGSM-L_∞	**53.71**	**53.71**	**53.67**	**53.59**	**53.55**	**53.20**	47.62	45.39	43.36
	COAP-L_∞	50.86	50.86	50.78	50.74	50.66	49.84	43.59	29.96	29.96
	Nominal	46.02	46.02	46.02	45.98	45.98	45.66	43.87	39.69	36.25

	$\boldsymbol{\rho} =$	0.000	0.001	0.003	0.010	0.100	0.300	0.500	1.00	5.00
L_∞ Attacks	RUB	50.86	46.02	47.58	45.08	21.64	9.38	9.30	9.30	8.12
	aRUB-L_1	53.28	**50.94**	**50.23**	**47.07**	26.91	10.66	10.12	10.12	10.62
	aRUB-L_∞	53.52	46.76	45.23	42.54	29.06	10.43	10.78	10.78	9.69
	Baseline-L_∞	51.48	47.54	40.90	39.22	9.34	9.34	9.34	9.34	10.00
	PGD-L_∞	**54.10**	49.34	48.05	46.33	25.31	**15.90**	**12.93**	**12.89**	**13.16**
	FGSM-L_∞	54.06	50.39	43.12	40.31	**30.63**	13.55	10.94	10.94	11.88
	COAP-L_∞	51.76	48.52	44.92	41.76	11.56	11.56	11.56	11.56	10.00
	Nominal	46.72	45.31	43.44	39.18	9.92	9.92	9.92	9.92	10.00

Table 2.6: Average number of batches processed per second across the 3 vision data sets, as well as the corresponding standard deviations.

	Avg no. batches per second	**Standard Deviation**
RUB	4.8	0.2
aRUB -L_1	53.1	2.4
aRUB -L_∞	56.4	0.2
Baseline -L_∞	343.5	33.4
PGD -L_∞	3.7	0.2
FGSM -L_∞	86	4.1
COAP -L_∞	12.2	0.3
Nominal	473	45.2

In Table 2.6 we present the average number of batches processed per second as well as the corresponding standard deviation for each method across the 3 vision data sets. We observe that FGSM-L_∞ is the fastest defense after Baseline-L_∞, followed by the aRUB methods. We highlight that contrary to the results obtained with the UCI data sets, the RUB method is slower than the aRUB defenses. This is attributed to the increased memory requirements for RUB with high dimensional inputs, which prevented full parallelization during training.

Security Guarantees against L_1 Norm Bounded Attacks. Finally, we use the upper bound of the adversarial loss derived in section 2.5 to find lower bounds for the adversarial accuracy with respect to attacks bounded in the L_1 norm by ρ. Specifically, the RUB-L_1 defense finds an upper bound for $\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_1\leq\rho} \boldsymbol{z}^L_k(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}^L_y(\boldsymbol{\mathcal{W}},\boldsymbol{x}+\boldsymbol{\delta})$, and therefore when this upper bound is nonpositive for all $k \in [K]$ we know that the network $\boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\cdot)$ correctly classifies all adversarial attacks $\boldsymbol{x}+\boldsymbol{\delta}$ for which $\|\boldsymbol{\delta}\|_1 \leq \rho$. In other words, the nonpositivity of this upper bound gives the network a security guarantee against the attacks considered. The percentage of images in the testing set for which this guarantee exists is therefore a lower bound of the adversarial accuracy achieved by network. In Tables 2.7, 2.8 and 2.9 we report the lower bounds for each method by selecting the hyperparameters that lead to the best lower bound in the validation set. In particular, notice that for a given choice of radius ρ and defense method, the selected network might not be the same as the one selected in the previous results for adversarial accuracy.

We observe that, as expected, for all three data sets, CIFAR, Fashion MNIST and MNIST, the best security guarantees are the ones for the RUB method. While these results are only lower bounds for the adversarial accuracy and we cannot claim a better accuracy for RUB than for the rest of the methods, the lower bound for the RUB method shows that this method indeed performs very well against L_1 attacks bounded by large values of ρ. For instance, in Table 2.7 we can see that for the Fashion MNIST data set the RUB method guarantees 86.02% adversarial accuracy (less than 5% decrease from the best natural accuracy) against attacks with L_1 norm smaller or equal to $\rho = 2.8$. Similarly, in Table 2.8 we observe that for this same attacks RUB has at least 97.11% adversarial accuracy (less than 1% decrease over natural accuracy) for the MNIST data set. And finally, for the CIFAR data set, we can see in Table 2.9 that RUB achieves 45.66% adversarial accuracy (less than 5% decrease over natural accuracy) against attacks whose L_1 norm is upper bounded by $\rho = 5.54$.

Table 2.7: Fashion MNIST: Lower bound of adversarial accuracy with uncertainty bounded in L_1 norm by ρ.

$\boldsymbol{\rho} =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00
RUB	90.27	90.16	90.16	89.49	**86.02**	**80.55**	**76.21**	**66.95**
aRUB-L_1	**90.51**	**90.51**	**90.39**	**89.73**	85.59	73.98	69.61	47.54
aRUB-L_∞	89.96	89.92	89.84	88.75	76.60	21.37	10.16	9.84
Baseline-L_∞	89.49	89.38	89.38	87.50	77.42	46.41	20.04	15.23
PGD-L_∞	89.92	89.92	89.92	87.85	78.75	40.35	19.22	15.55
Nominal	88.59	88.55	88.48	87.93	77.50	40.98	15.04	9.88

Table 2.8: MNIST: Lower bound of adversarial accuracy with uncertainty bounded in L_1 norm by ρ.

$\boldsymbol{\rho} =$	0.00	0.01	0.06	0.28	2.80	8.40	14.00	28.00
RUB	98.01	97.93	97.93	97.46	**97.11**	**93.67**	**89.77**	**74.96**
aRUB-L_1	**98.52**	**98.48**	**98.48**	98.20	96.48	89.96	80.86	26.05
aRUB-L_∞	98.40	98.40	98.28	97.54	94.69	68.52	16.56	12.19
Baseline-L_∞	98.05	98.05	98.05	97.81	95.23	57.23	20.74	10.35
PGD-L_∞	98.48	**98.48**	98.44	**98.36**	95.55	63.59	15.43	11.60
Nominal	97.73	97.73	97.58	97.54	93.87	44.80	10.31	10.23

Table 2.9: CIFAR: Lower bound of adversarial accuracy with uncertainty bounded in L_1 norm by ρ.

$\boldsymbol{\rho} =$	0.00	0.01	0.06	0.55	5.54	16.63	27.71	55.43
RUB	50.62	50.62	49.96	48.91	**45.66**	**37.81**	**32.85**	**23.28**
aRUB-L_1	53.40	53.16	51.99	**51.33**	43.36	33.83	27.19	15.16
aRUB-L_∞	53.67	53.52	53.20	47.07	37.30	14.26	9.65	9.65
Baseline-L_∞	53.32	53.24	52.66	46.09	36.99	13.71	9.61	9.61
PGD-L_∞	**54.88**	**54.80**	**53.98**	49.26	37.42	27.30	14.02	12.46
Nominal	46.88	46.88	46.76	44.69	37.19	14.88	9.02	8.95

2.7 Conclusions

We developed two new methods for adversarial training of neural networks, both of which provide an upper bound of the adversarial loss by considering the whole network at once instead of applying convex relaxations and propagating bounds for each layer separately as in previous works. First, we found an empirical upper bound by incorporating the first order approximation of the network's output layer. This method does not provide security guarantees against adversarial attacks but it performs very well across a variety of data sets when the uncertainty set is small and it stands out for its simplicity. Second, by extending state-of-the-art tools from RO to non-convex and non-concave functions, we were able to construct a provable upper bound of the adversarial loss. Experimental results show that this method has a performance edge for larger uncertainty sets, and importantly, this method can certify the non-existence of adversarial attacks bounded in L_1 norm. The two proposed upper bounds are in closed-form and can be effectively minimized with backpropagation. Lastly, we provide evidence that adding robustness can improve the natural accuracy of neural networks for classification problems with tabular or vision data.

For future work we are interested in extending the RUB approach for other types of norms as well as understanding how the tightness of the proposed upper bounds change across layers in order to facilitate further improvements. Adversarial robustness is crucial in the development of more secure machine learning systems, and we hope that our work will inspire further research in this important area.

Chapter 3

Holistic Deep Learning

3.1 Introduction

Neural networks have become increasingly popular due to their remarkable achievements in computer vision and natural language processing. Their generalization power has been demonstrated in wide-ranging applications, from classifying photos to recommending products. However, neural networks face challenges in real-world applications for high-stakes decision-making, including healthcare, policy-making, and autonomous driving.

First, many standard neural networks are not robust – they can be easily fooled by natural or artificial noise in the input data (Szegedy et al. 2014), making them vulnerable to perturbations that may arise in real-world applications. Moreover, neural networks, similar to other machine learning models, often suffer from instability during the training process – different train-validation splits could generate models with very different performance (May et al. 2010, Xu and Goodacre 2018). This reduces the policymakers' trust in these models and hinders post-hoc interpretations. Another critical difficulty is that neural networks are not sparse – the high number of parameters utilized for neural networks prevents efficient computation and storage (Thompson et al. 2020). Most neural networks have millions of non-zero parameters to be stored and accessed for evaluation. This is problematic in many decision-making settings with limitations or restrictions on hardware capabilities. Reducing the number of parameters could make them more applicable in a broader range of scenarios (Changpinyo et al. 2017, Narang et al. 2017).

The questions around improving robustness, stability, and sparsity metrics have all been previously studied in the neural network literature. However, they have been almost exclusively studied in isolation, with a limited understanding of the tradeoffs between these desired qualities and their effect on natural accuracy (accuracy with respect to the unperturbed data samples). This paper aims to simultaneously address all these objectives through a novel comprehensive methodology named Holistic Deep Learning (HDL). In particular, HDL carefully combines state-of-the-art techniques that address these individual challenges and demonstrates their collective efficacy through extensive experiments on diverse data sets. Our findings provide a promising pathway toward developing efficient and reliable machine learning models across many dimensions for real-world applications.

Specifically, our contributions are as follows:

1. We design HDL, a novel framework that jointly optimizes for neural network robustness (adversarial accuracy), stability (worst accuracy across train-val splits), and sparsity (parameters with value zero) metrics by appropriately modifying the objective function.

2. Through extensive ablation experiments and SHAP value analysis (Lundberg and Lee 2017) across 45 UCI data sets (Dua and Graff 2017) and 3 image data sets (MNIST (Deng 2012b), Fashion MNIST (Xiao et al. 2017b) and CIFAR10 (Krizhevsky et al. 2009b)), we analyze the individual performance of each metric as well as the interactions and trade-offs between them. We corroborate that imposing robustness, stability, and sparsity improves the corresponding metrics across all data sets. In addition, we show that:

    - imposing stability and sparsity further improves robustness,

    - imposing stability and robustness further improves sparsity,

- imposing robustness further improves stability,
- imposing stability and robustness further improves natural accuracy.

The effect of sparsity on natural accuracy is more complex and highly varies across data sets. However, we show that it is often possible to simultaneously improve robustness, stability, and sparsity without sacrificing performance on natural accuracy.

3. We propose a prescriptive approach to provide recommendations on selecting the appropriate loss function depending on the practitioner's objective. In particular, simultaneously imposing robustness, stability and sparsity in the loss function leads to the best results when jointly optimizing for all the metrics.

The paper is organized as follows: Section 3.2 outlines the current literature of robust, sparse, and stable methods; Section 3.3 describes the Holistic Deep Learning framework, and Section 3.4 shows the results of the computational experiments.

3.2 Related Work

3.2.1 Robust Neural Networks

Many state-of-the-art deep neural networks are highly vulnerable to small perturbations in the input data (Szegedy et al. 2014). Adversarial robustness evaluates a neural network's resistance against these altered inputs intentionally designed to worsen the network's performance (Goodfellow et al. 2014, Carlini and Wagner 2017, Madry et al. 2017).

Multiple methods have been developed in recent years to enhance the adversarial robustness of neural networks. One of the most popular heuristics is augmenting the data set during training with adversarial examples (Madry et al. 2017, Goodfellow et al. 2014). Others include neuron randomization (Prakash et al. 2018, Xie et al. 2017), input space projections (Lamb et al. 2018, Kabilan et al. 2018, Ilyas et al. 2017) and regularization (Bertsimas et al. 2021a, Ross and Doshi-Velez 2017, Hein and Andriushchenko 2017, Yan et al. 2018). A less common but more theoretically rigorous approach is to minimize a provable upper bound of the loss achieved with adversarial examples (Raghunathan et al. 2018b, Singh et al. 2018, Zhang et al. 2018, Weng et al. 2018, Dvijotham et al. 2018b, Lecuyer et al. 2019, Cohen et al. 2019, Anderson et al. 2020, Bertsimas et al. 2021a).

While these methods successfully improve the network's robustness, the extent to which they do so often depends on the data set, the network size, and the magnitude of the input perturbations. In particular, heuristic methods generally work well for small perturbations, while the upper bound methods yield better results when the input noise is larger (Bertsimas et al. 2021a, Athalye et al. 2018). However, there is a trade-off between effectiveness and efficiency. The methods providing the strongest adversarial robustness are often computationally demanding, making it challenging to implement them for large data sets or complex network architectures.

3.2.2 Sparse Neural Networks

In machine learning, sparse models make predictions based on a limited number of parameters. Sparsity is often desirable, as it may save memory, enhance model interpretability, and reduce overfitting (Bertsimas et al. 2020).

There are two typical approaches to sparsity in deep learning. The first one, *train-then-sparsify*, consists of removing unnecessary neurons or connections after training the network, sometimes followed by retraining (Janowsky 1989, LeCun et al. 1989). This approach has been widely investigated, and several schemes exist to choose which connections to prune (Hoefler et al. 2021). Han et al. (2015), for example, propose to prune the connections with the smallest weights. Other methods include formulating a convex optimization problem (Aghasi et al. 2020), removing filters for which the total absolute sum is low (Li et al. 2016), and eliminating channels that have limited impact on the network's discriminatory ability (Zhuang et al. 2018). The second approach, *sparsify-during-training*, is achieved by learning a sparse architecture while training the network. Multiple methodologies exist (Bellec et al. 2017, Mocanu et al. 2017, Mostafa and Wang 2019), including the method to approximate the ℓ_0 norm with continuous functions and add a regularization term to the loss function (Louizos et al. 2017, Savarese et al. 2020). We refer the reader to Gale et al. (2019) and Hoefler et al. (2021) for more comprehensive surveys on sparsity.

3.2.3 Stable Neural Networks

The stochastic nature of data samples can lead to instability and high dependence of machine learning models on the specific train-validation split. This can negatively impact the interpretability of the resulting model and its ability to make reliable predictions (Bertsimas and Paskov 2020), a key factor to establishing trust in any algorithm.

The sensitivity of machine learning models to the choice of training split has mostly been studied through the lens of cross-validation and distributionally robust optimization. Cross-validation can be used to measure the variability from the selection of training split but at a significant increase in computational cost (Krogh and Vedelsby 1994, Hastie et al. 2001) that is often intractable for deep learning settings. Distributionally robust optimization has been used to quantify the worst-case generalization error in the presence of shifts in distribution or regime (Staib and Jegelka 2019, Goldwasser et al. 2020, Sagawa et al. 2019), but it often requires pre-defined groups over the training data and expensive group annotations for each data sample to avoid overly pessimistic uncertain distributions (Sagawa et al. 2019, Liu et al. 2021). A different approach has been studied by Bertsimas and Paskov (2020) and Bertsimas et al. (2022a), who instead optimize over the worst training set of fixed size without making any probabilistic assumptions. Although their method was presented in the context of linear and tree-based models, their framework also applies to neural networks.

3.3 The Holistic Deep Learning Approach

3.3.1 The HDL Framework

We introduce the HDL framework for a classification problem with cross-entropy loss using the same notation as in the previous chapter. We illustrate our approach over a fully-connected neural network for simplicity of notation, but the framework remains the same for convolutional neural networks.

The nominal DL approach is to minimize the loss of the network $\boldsymbol{z}^L$ described in Eq. (2.2), which can be written as:

$$\min_{\boldsymbol{\mathcal{W}}} \frac{1}{N} \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x})} \right), \tag{3.1}$$

In our HDL framework, we propose instead to minimize the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{s},\theta,\boldsymbol{\mathcal{W}}} \underbrace{\lambda \sum_{j=1}^{|\boldsymbol{\mathcal{W}}|} \sigma(\beta s_j)}_{\text{Sparsity}} + \underbrace{\theta}_{\text{Stability}} + \\ &\frac{1}{a} \sum_{n=1}^{N} \left[\log \sum_{k=1}^{K} e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}} \odot \sigma(\beta \boldsymbol{s}), \boldsymbol{x}) + \overbrace{\rho \| \nabla_{\boldsymbol{x}} (\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}} \odot \sigma(\beta \boldsymbol{s}), \boldsymbol{x}) \|_1}^{\text{Robustness}}} - \underbrace{\theta}_{\text{Stability}} \right]^+, \end{aligned} \tag{3.2}$$

where $\odot$ corresponds to the element-wise product, σ is the standard sigmoid function, $\boldsymbol{z}^L(\cdot, \boldsymbol{x})$ was defined in (2.2), λ (resp. ρ) is the regularization coefficient corresponding to the sparsity (resp. robustness) loss component, and a is the size of the data subsets used for the stability requirement (see Section 3.2.3). We observe that robustness adds a term to the output, while stability and sparsity add new parameters (θ and $\boldsymbol{s}$ respectively) to be optimized. This loss function allows us to simultaneously train robust, sparse, and stable feed-forward neural networks at scale. In the next section, we provide more details about each metric.

3.3.2 Robustness

This section describes our method to introduce the robust component into neural network training. Since our ultimate goal is to incorporate the sparsity, robustness, and stability of neural networks together in a tractable way, we avoid algorithms that improve robustness at the expense of a significant increase in the training time or the algorithm's complexity (for instance, the algorithms that perform training with adversarial examples usually require significantly longer times to optimally find such examples at each gradient descent iteration (Madry et al. 2017, Bertsimas et al. 2021a)). We follow the approach from Section 2.4 of using a linear approximation of the neural network to estimate the robust objective. This approach is simple to implement, produces good adversarial accuracy, and does not require the extensive training time of other algorithms.

We then minimize the loss function in Eq. (2.14). As shown in Section 5.7, for small ρ this approach achieves competitive results with state-of-the-art methods while requiring significantly less computational time across various tabular and image data sets. However, we emphasize that the methodology developed in this paper could also be performed with other methods for robust training, like adversarial training or upper bound minimization, which might be more appropriate for large uncertainty sets.

3.3.3 Sparsity

In this work, we use the specific retraining procedure proposed by Savarese et al. (2020), which deterministically approximates the ℓ_0 regularization utilizing a sequence of sigmoid functions and adding them as a penalty term in the loss function. Notably, the implementation is easily compatible with our robustness and stability requirements, since this methodology relies on a penalty term added in the loss function. Therefore, we can use gradient descent to simultaneously optimize the objective function comprising the robustness, stability, and sparsity penalties.

Adding ℓ_0 regularization explicitly penalizes the number of non-zero weights in the model to induce sparsity. However, the ℓ_0-norm induces a priori a non-convex and non-differentiable loss function $\mathcal{R}(\boldsymbol{\mathcal{W}})$, as follows:

$$\mathcal{R}(\boldsymbol{\mathcal{W}}) = \frac{1}{N}\left(\sum_{n=1}^{N} \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)\right)\right) + \lambda \|\boldsymbol{\mathcal{W}}\|_0, \quad \|\boldsymbol{\mathcal{W}}\|_0 = \sum_{j=1}^{|\boldsymbol{\mathcal{W}}|} \mathbb{I}\left[w_j \neq 0\right], \tag{3.3}$$

where $|\boldsymbol{\mathcal{W}}|$ is the number of parameters, w_j is the j^{th} coordinate of $\boldsymbol{\mathcal{W}}$, λ is the regularization weight and $\mathcal{L}$ a loss function (e.g., cross-entropy loss).

The goal is to relax the discrete nature of the ℓ_0 penalty to preserve an efficient continuous optimization while allowing for exact zeros in the neural network weights. To do this, Savarese et al. (2020) propose to first parameterize the weights $w_j = H(s_j)$ where $H(\cdot)$ is the Heaviside step function, and then approximate the non-differentiable step function with the sigmoid function: $\sigma(\beta s_j) \to H(s_j)$ when $\beta \to \infty$. Therefore, β is the hardness parameter that controls how close the approximation is to the ℓ_0 regularization, and the final loss function can be written as:

$$\mathcal{R}(\boldsymbol{\mathcal{W}}) \approx \frac{1}{N}\left(\sum_{n=1}^{N} \mathcal{L}\left(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}} \odot \sigma(\beta \boldsymbol{s}), \boldsymbol{x}_n)\right)\right) + \lambda \sum_{j=1}^{|\boldsymbol{\mathcal{W}}|} \sigma(\beta s_j). \tag{3.4}$$

To achieve a sparse network, we use this loss function (3.4) over multiple training rounds to gradually reach a sparse initialization before training the final sparse neural network. To obtain each initialization before a new training round, we start with our initialized auxiliary sparsity $\boldsymbol{s}_0$ and hardness $\beta = 1$ parameters. Over the T training iterations, we gradually increase β until it reaches a maximum value $\bar{\beta}$ when the training procedure is completed with sparsity $\boldsymbol{s}_T$. Then, we take $\boldsymbol{s}_0' = \min(\bar{\beta}\boldsymbol{s}_T, \boldsymbol{s}_0)$ to generate the new initialization for the next round of training. This minimization function essentially keeps the information of the suppressed weights, i.e., $\sigma(\beta s_j) \approx 0$, while reverting those not suppressed to their starting position. This process is completed over multiple rounds to find better and sparser initializations for the neural network.

We implement the methodology as suggested by Savarese et al. (2020). In the results section, we measure sparsity in terms of the percentage of neuron connections (weights) set to 0.

3.3.4 Stability

Using the measure of stability defined in Section 3.2.3, we apply the methodology developed in Bertsimas et al. (2022a) for building stable neural networks. At a high level, this corresponds to constructing a model that is robust to the specific subset of data used to train it. One way to think about this is to view the training data set as a sample from the true data distribution and then require the model to be robust to the specific sample. Considering the partition of the data into training/validation sets as a sampling mechanism from this true data distribution (each split choice gives a different training set), we desire to build models that are robust to every partition.

To achieve this, we first associate each observation $(\boldsymbol{x}_n, y_n)$ with a binary variable z_n, $n \in [N]$ that indicates whether or not $(\boldsymbol{x}_n, y_n)$ is part of the training set. We then choose the network's parameters as to minimize the worst-case loss over all possible allocations of these z_n's, resulting in a model that is explicitly built to do well not just over one training set, but over all possible training sets. We start from the same minimization problem introduced in Section 3.3.1, i.e.,

$$\min_{\boldsymbol{\mathcal{W}}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)).$$

To obtain network stability we require the model to be robust to every training set of fixed size a, which results in the following optimization problem:

$$\begin{aligned}
&\min_{\boldsymbol{\mathcal{W}}} \max_{z \in \mathcal{Z}} \frac{1}{a} \sum_{n=1}^{N} z_n \mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)), \\
&\text{where } \mathcal{Z} = \left\{ z : \sum_{n=1}^{N} z_n = a, \quad z_n \in \{0,1\},\ n \in [N] \right\}.
\end{aligned} \tag{3.5}$$

The value of a indicates the desired proportion between the size of the training and validation sets. For example, by setting $a = 0.7N$ we recover the typical 70/30 training/validation split. Since the inner maximization problem is linear in z, the problem is equivalent to optimizing over the convex hull of $\mathcal{Z}$. This implies that the binary constraints on z_n can be relaxed to $0 \leq z_n \leq 1$, and the inner maximization problem becomes linear in the variables z_n. Computing its dual problem we obtain that the value of the inner maximization problem is equivalent to:

$$\min_{\theta, u_n} \ \theta + \frac{1}{a} \sum_{n=1}^{N} u_n \quad \text{subject to} \quad \theta + u_n \geq \mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)), \quad u_n \geq 0,\ n \in [N].$$

Therefore, the stability problem becomes

$$\min_{\substack{\boldsymbol{\mathcal{W}}, \\ \theta, u_n \in \mathbb{R}}} \ \theta + \frac{1}{a} \sum_{n=1}^{N} u_n \quad \text{subject to} \quad \theta + u_n \geq \mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)),\ u_n \geq 0,\ n \in [N].$$

Note that the variables u_n can be solved in closed form as $u_n = [\mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)) - \theta]^+$. The final minimization problem with stability then becomes:

$$\min_{\boldsymbol{\mathcal{W}}, \theta} \ \theta + \frac{1}{a} \sum_{n=1}^{N} \left[\mathcal{L}(y_n, \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}, \boldsymbol{x}_n)) - \theta \right]^+,$$

which is now an unconstrained problem that can be solved with standard gradient descent optimization algorithms.

3.4 Experiments

This section presents extensive computational experiments comparing the nominal DL approach (abbreviated DL) with 7 other models resulting from our holistic methodology. We showcase the merit of our HDL framework and investigate the influence of each studied component – robustness, sparsity, and stability – on the overall performance across 4 evaluation metrics:

- *Natural accuracy*: Average accuracy on the testing set across the 10 different train-validation splits with respect to the original input data.
- *Adversarial robustness*: Average adversarial accuracy on the testing set across the 10 different train-validation splits with respect to adversarial attacks resulting from perturbations of the original input data. We consider only attacks bounded in the L_∞ norm by some radius ρ using Projected Gradient Descent as in Madry et al. (2017).
- *Stability*: Worst accuracy on the testing set across the 10 different train-validation splits with respect to the original input data.
- *Sparsity*: Percentage of network parameters with value 0.

The exact optimization problem solved for each model results from combinations of the loss functions described in the previous section, and the specific formulations can be found in Table 3.1 above.

Data We computed experiments on classification tasks with 45 UCI data sets from the UCI Machine Learning Repository (Dua and Graff 2017). These data sets give various problem sizes and difficulties to form a representative sample of real-world tabular problems, with the largest data set having 245,056 observations and the highest number of features being 856. We also benchmarked our methodologies on three image data sets: MNIST, Fashion-MNIST, and CIFAR10.

Method	Optimization Problem
DL	$\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x}_n)}\right)$
Robust	$\min_{\boldsymbol{\mathcal{W}}} \ \frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})+\rho\Vert\nabla_{\boldsymbol{x}}(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})\Vert_1}\right)$
Stable	$\min_{\boldsymbol{\mathcal{W}},\theta} \ \theta + \frac{1}{a}\sum_{n=1}^{N}[\log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x}_n)}\right) - \theta]^+$
Sparse	$\min_{\boldsymbol{\mathcal{W}},\boldsymbol{s}} \ \lambda\sum_{j=1}^{\vert\boldsymbol{\mathcal{W}}\vert}\sigma(\beta s_j) + \frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}\odot\sigma(\beta\boldsymbol{s}),\boldsymbol{x}_n)}\right)$
Robust + Sparse	$\min_{\boldsymbol{\mathcal{W}},\boldsymbol{s}} \ \lambda\sum_{j=1}^{\vert\boldsymbol{\mathcal{W}}\vert}\sigma(\beta s_j) + \frac{1}{N}\times$ $\sum_{n=1}^{N}\log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}\odot\sigma(\beta\boldsymbol{s}),\boldsymbol{x})+\rho\Vert\nabla_{\boldsymbol{x}}(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})\Vert_1}\right)$
Stable + Sparse	$\min_{\boldsymbol{\mathcal{W}},\theta,\boldsymbol{s}} \ \lambda\sum_{j=1}^{\vert\boldsymbol{\mathcal{W}}\vert}\sigma(\beta s_j) + \theta + \frac{1}{a}\sum_{n=1}^{N}[\log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}\odot\sigma(\beta\boldsymbol{s}),\boldsymbol{x}_n)}\right) - \theta]^+$
Stable + Robust	$\min_{\boldsymbol{\mathcal{W}},\theta} \ \theta + \frac{1}{a}\sum_{n=1}^{N}[\log\left(\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})+\rho\Vert\nabla_{\boldsymbol{x}}(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})\Vert_1}\right) - \theta]^+$
HDL	$\min_{\boldsymbol{\mathcal{W}},\theta,\boldsymbol{s}} \lambda\sum_{j=1}^{\vert\boldsymbol{\mathcal{W}}\vert}\sigma(\beta s_j) + \theta + \frac{1}{a}\times$ $\sum_{n=1}^{N}[\log\sum_k e^{(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}}\odot\sigma(\beta\boldsymbol{s}),\boldsymbol{x})+\rho\Vert\nabla_{\boldsymbol{x}}(\Delta \boldsymbol{e}_k^{y_n})^\top \boldsymbol{z}^L(\boldsymbol{\mathcal{W}},\boldsymbol{x})\Vert_1} - \theta]^+$

Table 3.1: Loss functions used for DL and all methods in the HDL framework.

Implementation Our code is written in Python 3.8 (Van Rossum and Drake 2009a). Neural networks are coded using Tensorflow v1 (Abadi et al. 2015). We trained each model on a system equipped with an Intel Xeon Gold 6248 processor, which included 4 CPU cores and one Nvidia Volta V100 GPU.

Training Methodology For each data set, we used 20% of the data to obtain a fixed test set, and we randomly generated 10 different 80%-20% train-validation splits with the remaining data points. The same 10 train-validation partitions were used across all methods for a fair comparison. Given a choice of model and evaluation metric, we selected the hyperparameters that led to the best average performance in the validation set for the metric in question. We then reported the average performance of the chosen parameter configuration on the test set with respect to the given metric. For all evaluation metrics, the average performance is computed over the 10 training-validation splits initially generated.

Neural network architectures For our experiments on UCI data sets, we used a feed-forward neural network architecture with 2 hidden layers, each with 128 neurons and ReLU activations. For our experiments on the image data sets, we used a convolutional neural network with the AlexNet architecture (Krizhevsky et al. 2012). We used the Glorot uniform initialization (Glorot and Bengio 2010b) for the network weights $\boldsymbol{\mathcal{W}}$ and $\mathbf{0}$ as initialization for the sparsity variable $\boldsymbol{s}_0$. The choice of architecture and initialization was made to reflect typical settings utilized in the machine learning community (e.g. Madry et al. (2017), Savarese et al. (2020), Bertsimas et al. (2021a)) while maintaining moderate size networks that facilitate exhaustive experimentation across dozens of data sets. Importantly, the same architecture is used across all methods been evaluated.

Hyperparameter search For each model, we cross-validated the values of the following hyperparameters:

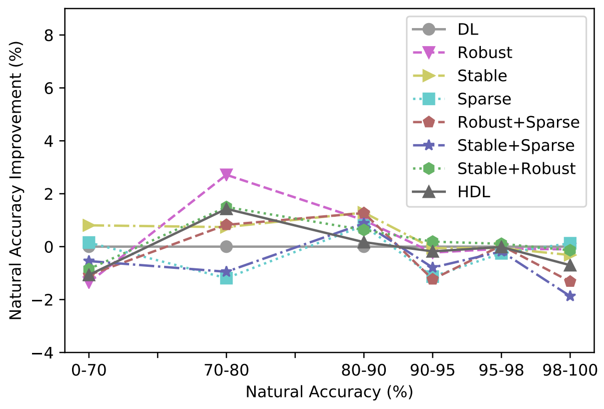
- Adam learning rate: $\{1e^{-2}, 1e^{-3}\}$ for UCI data sets, $\{1e^{-3}, 1e^{-4}\}$ for image data sets.
- Number of epochs: 150 for UCI data sets, 50 for vision data sets.
- Batch Size: 32 for UCI data sets, 64 for image data sets.
- Robustness radius $\rho : \{1e^{-1},\ 1e^{-2},\ 1e^{-3},\ 1e^{-4},\ 1e^{-5}\}$.
- Sparsity regularization parameter λ: $\{1e^{-6},\ 1e^{-8},\ 1e^{-10}\}$.
- Sparsity temperature parameter $\bar{\beta} : \{200, 1000\}$.
- Stability parameter a: $\{0.7,\ 0.8,\ 0.9,\ 1\}$.

3.4.1 UCI Data sets

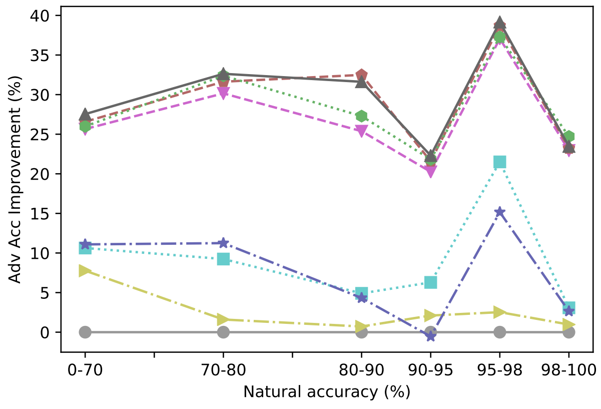
We split the 45 UCI data sets into 6 roughly even-sized groups based on their difficulty level. Specifically, we consider the ranges 0%-70%, 70%-80%, 80%-90%, 90%-95%, 95%-98% and 98%-100% of natural accuracy achieved by the nominal DL approach. We first investigate the performance of the HDL framework with respect to a single evaluation metric. In Figure 3.1, we evaluate all methods in terms of natural accuracy, adversarial accuracy with $\rho = 0.1$, stability, and sparsity.

Figures 3.1a and 3.1c show that those data sets for which the nominal approach achieves accuracy in the 70%-90% range are the ones that benefit the most from the HDL framework (especially the Robust, Stable, and Stable+Robust models) when the evaluation metric corresponds to natural accuracy or stability. For the data sets with natural accuracy above 90%, none of the models significantly improve over the natural accuracy or stability achieved by the nominal DL model. However, for data sets in the 98-100% range sparsity slightly improves accuracy and robustness slightly helps for stability.

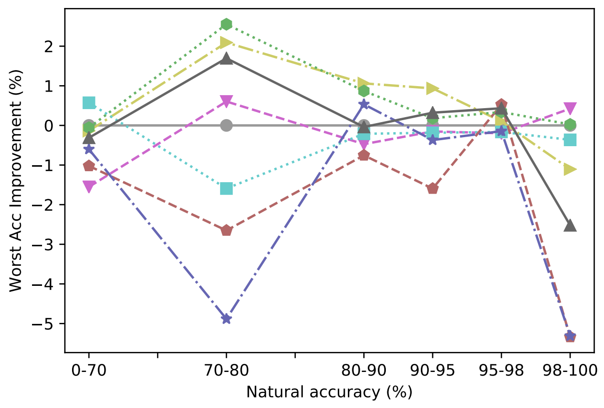
Figure 3.1b shows the adversarial robustness achieved with perturbation parameter $\rho = 0.1$. We see a substantial adversarial robustness improvement in all methods that included the robust component. Moreover, combining robustness with stability and/or sparsity leads to higher adversarial accuracy than that achieved with robustness alone. In terms of parameter sparsity, Figure 3.1d shows that all models with imposed sparsity (Sparse, Stable+Sparse, Robust+Sparse, and HDL) have a much lower percentage of nonzero parameters compared to the models without it. And importantly, both robustness and stability help achieve sparser models when combined with sparsity.



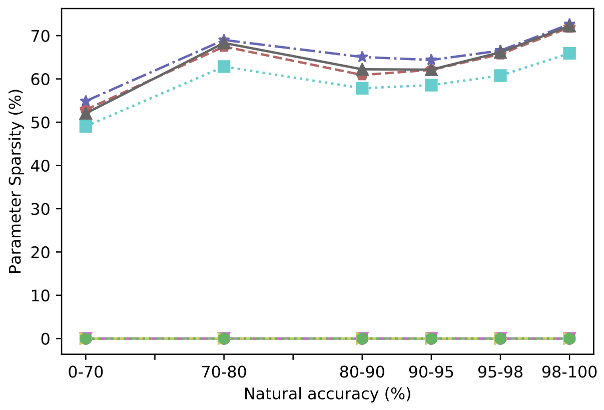
(a) Natural accuracy.



(b) Adversarial accuracy with $\rho = 0.1$.



(c) Stability.



(d) Sparsity.

Figure 3.1: Evaluation of the different methods depending on the natural accuracy of the nominal DL approach on the UCI data sets.

Since we are also interested in models that are simultaneously accurate, sparse, robust, and stable, we consider a multi-objective metric using the rank of each method (ranks start at 1, with lower ranks corresponding to better performance). For each method, we use the natural accuracy, adversarial accuracy, stability, and sparsity achieved in the validation set respectively to rank all its hyperparameter configurations 4 times. Then for each hyperparameter configuration, we compute the average rank across the 4 metrics and select the configuration that leads to the method's highest average rank. Finally, we rank the 8 selected models (for the 8 different methods) with respect to each evaluation metric on the testing set to obtain their out-of-sample average rank.

As shown in Figure 3.2, all 7 models from the HDL framework outperform the nominal DL approach with respect to this holistic metric. Moreover, the HDL model typically achieves the best results across data set complexities.

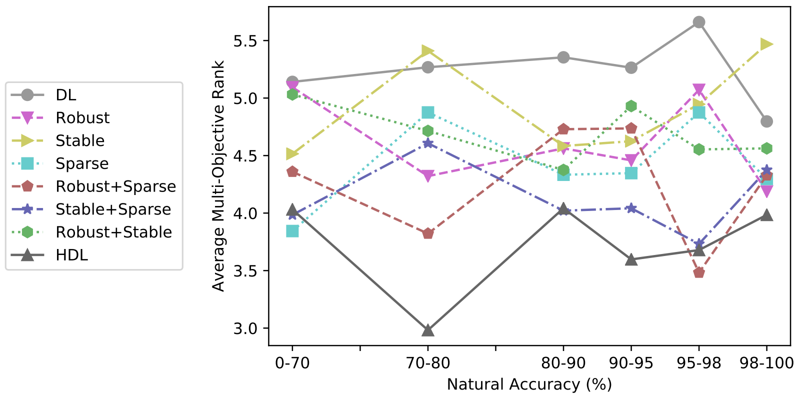


Figure 3.2: Average multi-objective rank.

3.4.2 Image Data Sets

In this section, we evaluate all methods using the MNIST, Fashion-MNIST, and CIFAR10 data sets. For each method, we select the parameters based on the multi-objective metric utilized for the UCI data sets in the validation set and report the performance across metrics. In Tables 3.2 and 3.3, we see that for MNIST and Fashion-MNIST, the HDL model outperforms the DL model for all objectives. In particular, HDL achieves higher accuracy using only around 70% of the parameters. The results for the CIFAR10 data set (Table 3.4) are a bit different since adding sparsity slightly hurts natural accuracy. However, the accuracy achieved by the HDL model is comparable to those achieved by the non-sparse models and the number of parameters is reduced by 47%.

Method	**DL**	**Rob.**	**Stab.**	**Sparse**	**Rob. + Sparse**	**Stab. + Sparse**	**Stab. + Rob.**	**HDL**
Avg. Accuracy	91.8	92.0	91.9	91.4	**92.1**	91.4	92.0	**92.1**
Adv. Acc. $\rho = 0.01$	78.7	81.1	78.3	80.8	86.9	80.2	86.8	**87.1**
Stability	91.5	91.7	**91.8**	91.3	91.9	91.2	91.7	**91.8**
Sparsity	0	0	0	36.2	26.6	**48.4**	0	26.8

Table 3.2: Results for the Fashion-MNIST data set. For each method, the parameters with the highest average rank in the validation set were chosen.

Method	**DL**	**Rob.**	**Stab.**	**Sparse**	**Rob. + Sparse**	**Stab. + Sparse**	**Stab. + Rob.**	**HDL**
Avg. Accuracy	99.2	**99.3**	99.2	99.1	99.2	99.2	**99.3**	**99.3**
Adv. Acc. $\rho = 0.1$	49.6	78.4	51.5	42.6	74.7	27.7	**79.5**	76.0
Stability	99.1	99.2	99.2	99.1	99.1	99.0	**99.3**	99.2
Sparsity	0	0	0	16.1	27.9	**31.7**	0	28.0

Table 3.3: Results for the MNIST data set. For each method, the parameters with the highest average rank in the validation set were chosen.

Method	**DL**	**Rob.**	**Stab.**	**Sparse**	**Rob. + Sparse**	**Stab. + Sparse**	**Stab. + Rob.**	**HDL**
Avg. Accuracy	**70.1**	**70.1**	**70.1**	69.8	69.2	69.3	69.8	69.3
Adv. Acc. $\rho = 0.01$	26.7	27.1	26.6	27.3	27.4	27.7	29.1	**30.6**
Stability	69.7	69.7	**69.8**	69.3	68.9	68.5	69.4	68.8
Sparsity	0	0	0	28.7	47.2	**47.8**	0	**47.8**

Table 3.4: Results for the CIFAR10 data set. For each method, the parameters with the highest average rank in the validation set were chosen.

3.4.3 Computational Times

Since modifying the loss function often affects the training computational time, we quantify the slowdown effect for all the methods in the HDL framework. Specifically, for each of the 45 UCI data sets as well as the 3 image data sets introduced in the previous section, we calculate how many times slower each method is when compared to the nominal DL approach in terms of batches per second as well as number of iterations needed. The average slowdown factors are shown in Table 3.5.

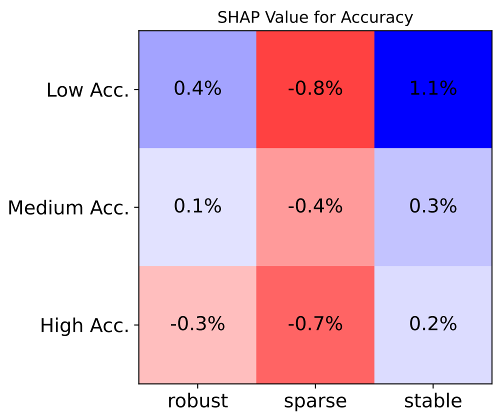
We observe that robustness and sparsity both decrease the number of batches per second by approximately a factor of 3, while stability preserves the same speed as the DL approach. In addition, since we used 5 training rounds for the methods incorporating sparsity, they require 5 times as many training iterations as the other methods. On average, the HDL method is only 16 times slower, and methods that don't optimize for sparsity only increase the computational time by less than 3 times.

Method	**Batches/sec Slowdown Factor**	**No. Iterations Increase Factor**	**Total Slowdown factor**
Robust	2.7	1	2.7
Stable	1.0	1	1.0
Sparse	1.2	5	5.9
Robust+Sparse	3.2	5	16.1
Stable+Sparse	1.1	5	5.5
Stable+Robust	2.7	1	2.7
HDL	3.2	5	16.2

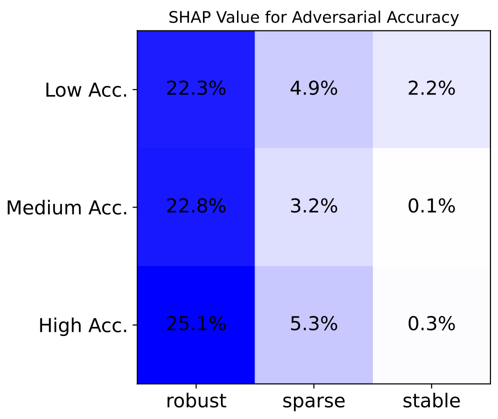
Table 3.5: Average slowdown factors of computational time with respect to the nominal DL method.

3.4.4 SHAP Values

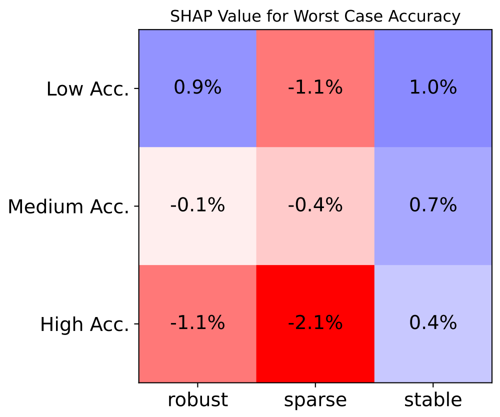
To gain a deeper understanding of the interplay between individual loss components (robustness, stability, sparsity) and the metrics we measure, we employ the SHAP values method (Lundberg and Lee 2017). We compute the SHAP values for each UCI data set and average the results over three data set categories: Low Accuracy (< 80%), Medium Accuracy (80%-95%), and High Accuracy (> 95%), with 15 data sets each. The results are shown in Figure 3.3.



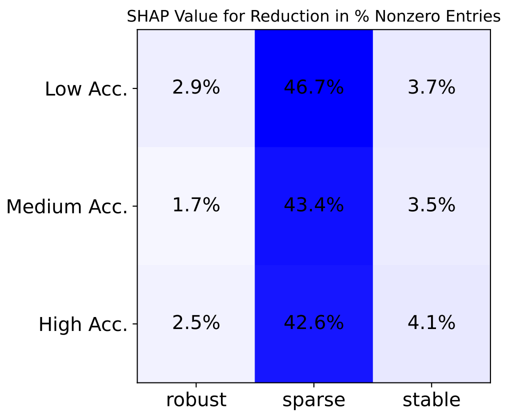
(a) Accuracy.



(b) Adversarial accuracy with $\rho = 0.1$.



(c) Stability.



(d) Sparsity.

Figure 3.3: SHAP values on various metrics across different UCI data set categories. Blue/red indicates that the feature has a positive/negative SHAP value on a specific category of UCI data set.

Our findings confirm that robustness, stability, and sparsity techniques improve the corresponding metrics across all data set categories. More intriguingly, these techniques also positively impact metrics beyond their intended purposes. For example, sparsity and stability enhance adversarial accuracy, while robustness and stability yield sparser networks. This indicates that combining techniques does not necessarily result in any adverse effects and that it is feasible to attain networks with good performance across all metrics. Additionally, the benefits of these techniques are more pronounced in data sets with low initial accuracy, particularly for the accuracy and stability metrics. Lastly, we observe that sparsity generally hurts accuracy and stability, although this highly varies across data sets, as observed in Section 3.4.1.

3.4.5 Prescriptive Approach

In this section, we develop a prescriptive approach that allows users to choose a training loss function based on the specific objective they wish to maximize, which can be a single evaluation metric or a weighted combination of several metrics. Depending on the data set characteristics and the performance scores of the nominal DL model, we propose a tree-based recommendation model to suggest the most suitable HDL loss function for optimal results with respect to the desired objective.

We train our models using an Optimal Policy Tree (OPT) algorithm (Amram et al. 2022), which uses observational data of the form $(\mathbf{x}_i, y_i, z_i)$. While it is possible to include variability and complexity indicators of the data set as part of $\mathbf{x}_i$ (Lorena et al. 2019), given the extensive and diverse range of data sets in consideration, we choose to capture complexity using the performance metrics achieved by the nominal DL approach on the corresponding classification tasks. In our case, each observation (i.e., data set) i encompasses:

- Data set features $\mathbf{x}_i \in \mathbb{R}^8$: number of features, number of target classes, nominal DL accuracy, nominal DL adversarial accuracy with $\rho = 0.001$, nominal DL adversarial accuracy with $\rho = 0.01$, nominal DL adversarial accuracy with $\rho = 0.1$, nominal DL stability, nominal DL worst case accuracy.

- Prescriptions $z_i \in 1, \ldots, 8$: DL, Robust, Stable, Sparse, Robust+Sparse, Stable+Sparse, Stable+Robust, HDL.

- Outcomes $\boldsymbol{y}_i \in \mathbb{R}^8$, which represent the performance improvement of each method compared to the nominal DL model with respect to the metric set by the user.

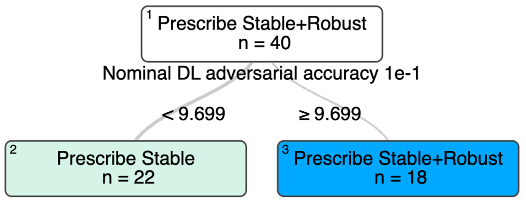

1 Prescribe Stable+Robust
n = 40
Nominal DL adversarial accuracy 1e-1
< 9.699
≥ 9.699
2 Prescribe Stable
n = 22
3 Prescribe Stable+Robust
n = 18

Figure 3.4: Optimal policy tree for maximizing natural accuracy.

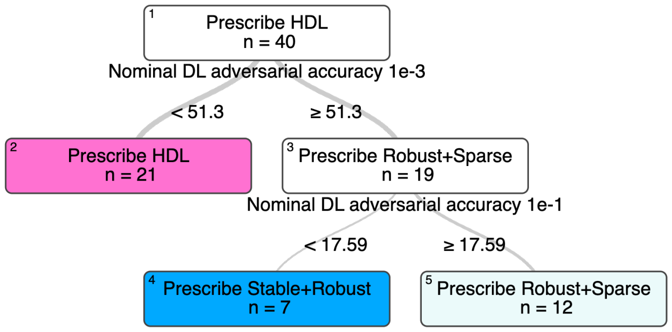

1 Prescribe HDL
n = 40
Nominal DL adversarial accuracy 1e-3
< 51.3
≥ 51.3
2 Prescribe HDL
n = 21
3 Prescribe Robust+Sparse
n = 19
Nominal DL adversarial accuracy 1e-1
< 17.59
≥ 17.59
4 Prescribe Stable+Robust
n = 7
5 Prescribe Robust+Sparse
n = 12

Figure 3.5: Optimal policy tree for maximizing robustness ($\rho = 0.1$).

Our prescriptive task is to find the optimal policy that, given the information $\mathbf{x}$ of a data set, prescribes the method z leading to the best metric score y. We randomly split the 45 UCI data sets into a training set (40 data sets) and a test set (5 data sets from different difficulty levels). We cross-validated the optimal tree depth and complexity using the training set.

Figures 3.4 and 3.5 represent the OPTs obtained for maximizing two different objectives: natural accuracy and adversarial accuracy. The tree in Figure 3.4 highlights that the Stable and Stable+Robust methods are the best suited to obtain high natural accuracy, with the former being preferred when the nominal DL approach has very low adversarial accuracy ($\rho = 0.1$). To maximize robustness, the tree in Figure 3.5 prescribes HDL, Stable+Robust, or Robust+Sparse depending on the adversarial accuracy achieved by the nominal DL method.

In addition, we obtained single-leaf trees when maximizing the stability and sparsity objectives. The recommended methods are Stable+Robust for optimizing stability and Stable+Sparse for maximizing sparsity. Lastly, HDL was always the prescribed method when the desired objective was the equally weighted average of all 4 previous metrics.

Finally, Table 3.6 reports the out-of-sample performance of these prescription trees on the 5 UCI data sets from the test set (cnae-9, hill-valley, libras-movement, magic-gamma, and thyroid-ann). We emphasize that the performance of the prescribed methods is higher than that of the nominal DL approach across all objectives and data sets, and it often matches the performance of the best method.

Objective	**Method**	**Test Data Sets Objective Value**				
		cnae-9	hill-valley	libras-move	magic-gamma	thyroid-ann
Nat. Acc.	DL	93.70	47.21	79.44	87.11	98.86
	Prescribed	94.07	53.61	80.00	87.28	99.05
	Optimal	94.07	53.61	82.50	87.28	99.05
Robustness ($\rho = 0.1$)	DL	0.00	7.54	0.00	15.07	48.42
	Prescribed	3.80	36.39	2.50	64.59	91.79
	Optimal	3.80	40.16	4.72	64.59	91.79
Stability	DL	91.20	43.44	75.00	86.65	98.28
	Prescribed	93.06	45.08	81.94	87.01	98.54
	Optimal	93.06	49.18	81.94	87.01	98.81
Sparsity	DL	0.00	0.00	0.00	0.00	0.00
	Prescribed	73.43	34.89	71.00	57.52	53.22
	Optimal	73.43	41.94	71.00	57.52	53.22
(Nat. Acc. +Robustness +Stability +Sparsity)/4	DL	46.25	24.55	38.61	47.21	61.39
	Prescribed	57.75	39.35	52.76	58.06	73.03
	Optimal	62.07	40.66	52.82	59.62	73.03

Table 3.6: Performance of prescription trees on the testing set.

3.4.6 Significance Analysis

To further validate the improvements achieved by the HDL framework, we analyze the significance of our results with one-sided Welch's t-tests with different variance groups. Specifically, for each evaluation metric and each leaf of the corresponding optimal prescriptive tree, we consider all the UCI and image data sets that fall within that leaf. For those data sets, we test the null hypothesis that the average performance achieved by the prescribed method is equal to that one achieved by the nominal DL approach, with alternative hypothesis corresponding to the average performance achieved by the prescribed method being higher. As shown in Table 3.7, all p-values are below the 0.05 significance level, concluding that the prescribed methods have statistically significant higher performance than the nominal DL approach across all performance metrics.

Objective	**Leaf Prescription**	**p-value**
Nat. Acc.	Stable	0.025
	Stable+Robust	0.0462
Robustness ($\rho = 0.1$)	HDL	$1.508e^{-6}$
	Stable+Robust	0.001
	Robust+Sparse	$1.727e^{-5}$
Stability	Stable+Robust	0.0161
Sparsity	Stable+Sparse	$1.188e^{-38}$
$\frac{\text{Nat. Acc.+Robustness+Stability+Sparsity}}{4}$	HDL	$4.472e^{-26}$

Table 3.7: Significance results for the null hypothesis that the average performance achieved by the prescribed method is equal to that one achieved by the nominal DL approach, with alternative hypothesis corresponding to the average performance achieved by the prescribed method being higher.

3.5 Conclusions

This paper presents a unifying methodology to obtain deep learning models that are accurate, robust, stable, and sparse by appropriately modifying the objective function to be minimized. Across multiple computational experiments, we show how these 4 metrics interact and demonstrate that we can often train models that simultaneously improve adversarial accuracy, worst-case accuracy, and parameter sparsity without sacrificing natural accuracy. Finally, we provide prescriptive trees that use general features of the data set (e.g. dimension, number of target classes, nominal accuracy, etc.) to recommend which method is more appropriate depending on the desired objective to be maximized, and we show that the improvements achieved by the prescribed methods are statistically significant.

For future research we aim to explore how HDL performs with respect to other data set indicators like variability and complexity, as this could offer further guidance on which method to select. We would also like to test our framework in real world applications; for instance in the area of healthcare, where trustworthy models are crucial and memory constraints are often required for practical use. Consequently, improving the interpretability of the HDL framework would be essential to make it more suitable for such applications. We deem adversarial robustness, stability and sparsity as critical qualities in the development of more reliable machine learning algorithms, and we hope this work will motivate further research in this important field.

Chapter 4

Large Language Models for Patient Flow Predictions

4.1 Introduction

Increasing data availability from Electronic Medical Records (EMR) combined with advances in machine learning (ML) generates new opportunities for enhancing decision-making within healthcare institutions. For instance, anticipating short-term discharges informs about bed availability and can facilitate resource utilization and personalized delivery of care. Furthermore, detecting patients with high mortality or ICU risk can alert the medical team and call their attention to those who need it the most.

Despite a growing literature on data-driven approaches for healthcare, deploying these models in practice remains a difficult task. Not only there is a need for close relationships between academics and medical teams, but also there are several data challenges that can make such collaborations difficult. For instance, unstructured healthcare data like images and notes are often difficult to access for model development due to privacy concerns and high computational costs. Structured healthcare data, or Electronic medical records (EMRs), are often available for most digitalized healthcare systems, but the tables are generally disorganized, not standardized across institutions and very scarce for small healthcare systems.

These challenges highlight two significant limitations in the existing approaches to handling EMRs and tabular data in general: 1) they require labor-intensive data processing that is unique to each institution, and 2) they ignore contextual information such as column headers and meta content descriptions which could be used for data augmentation.

In contrast to the standard tabular approaches, language is a very flexible data modality that can easily represent information about different data points without imposing any structural similarity between them. Furthermore, recent developments on off-the-shelf large language models (LLMs) based on the Transformer architecture (Vaswani et al. 2017) offer state-of-the-art performances on a wide range of language tasks, including translation, sentence completion, and question answering. These pre-trained models are often developed with very large and diverse data sets, allowing them to exploit prior knowledge and make accurate predictions with very few new training samples. Some LLMs are trained to target specific domain knowledge and technical challenges, making them particularly useful in the corresponding applications. For example, LLMs fine-tuned on clinical notes and biomedical corpora such as ClinicalBERT (Alsentzer et al. 2019), BioBERT(Lee et al. 2019), and BioGPT(Luo et al. 2022) offer substantial advantages for medical learning tasks, and LLMs that specifically target long token sequences unveil opportunities for dealing with data that contains long texts (Beltagy et al. 2020, Li et al. 2022).

These successful language models offer a natural solution to represent and process contextual information from tabular structures. Standard machine learning models only utilize the explicit table contents, disregarding all accompanying context like column headers and table descriptions. Incorporating these metadata into the model via language could give meaning to the data values within the broader context. For example, a numerical value might be very relevant for disease prediction if it represents a person's age but not so much if it corresponds to the ward census. Moreover, LLMs could save significant manual labor for selecting, encoding, and imputing data (Sweeney 2017, Geneviève et al. 2019, Nan et al. 2022). Missing data, in particular, is a challenging and frequent problem that requires attentive processing and expert knowledge. Current predictive models either exclude such attributes, potentially ignoring rare-occurring but valuable data or impute missing values with very few recorded instances. Additional processing challenges arise when units of measurement or data types are inconsistent across tabular data systems. By leveraging language, these difficulties could be addressed, for instance, by simply writing that particular values are missing and converting inconsistent values into text.

Previous works using LLMs have shown the potential of using natural language processing (NLP) models to systematically and efficiently process tabular data in the form of language (Herzig et al. 2020, Yin et al. 2020, Padhi et al. 2021, Somepalli et al. 2021). However, they have mainly relied on training fixed BERT-based models that are not flexible to changes in tabular structures. These works have mostly assumed that encoding data using LLMs leads to better performance than traditional data processing methods, but concrete evidence has not been provided. Other works augment tabular data with external unstructured data (Harari and Katz 2022) but do not leverage contextual data from the original tabular source. In addition, language models are considered sensitive to their input representations (Miyajiwala et al. 2022), and most previous works do not thoroughly investigate how the choice of language affects their results. Hegselmann et al. (2023) do investigate different language variants, but in the context of zero-shot and few-shot classification as opposed to feature representation. Thus, guidance on the best way to construct the language data remains in need.

In this chapter we first present a successful real-world implementation of machine learning models at a large healthcare system, and then we build and evaluate a new feature extraction methodology that leverages LLMs to improve and generalize these models. The main contributions of this work are as follows:

1. We develop and implement machine learning models that predict several inpatient outcomes for a large hospital network. We show that after utilizing our user-friendly

software the hospitals observe significant reduction in length of stay and millions of dollars in financial benefits.

2. We create TabText, a systematic framework that leverages language to extract contextual information from tabular structures. Our experiments demonstrate that augmenting electronic medical records with our TabText representations can significantly improve the AUC score of patient flow predictions, especially when trained with small-size datasets.

3. We investigate the impact of several language syntactic parsing schemes on the performance of TabText representations and demonstrate that TabText enables the generation of high-performing predictive models for patient outcomes with minimal data processing.

4.2 Patient Flow Predictions

Access to accurate predictions of patients' outcomes can enhance decision-making within healthcare institutions. In collaboration with a large hospital network, we develop machine learning models that predict short-term and long-term patient outcomes such us discharge, intensive care unit transfers and end-of-stay mortality. We implement an automated pipeline that extracts data and updates predictions every morning, as well as user-friendly software and a color-coded alert system to communicate these patient-level predictions to clinical teams. Since its deployment, over 200 doctors, nurses, and case managers across seven hospitals have been using the tool in their daily patient review process.

Collaboration HHC is the largest healthcare network in Connecticut; it contains 7 hospitals ranging from Hartford Hospital (HH), one of the largest teaching hospitals in New England (867 beds), to small and medium sized (130–520 beds) community / teaching hospitals. We have been collaborating with HHC since 2020, starting with daily data extraction from their EMR system. In 2021, we constituted physician pilot users who iteratively collected feedback and helped improve the models and the software tool. We extended models for the other 6 hospitals at HHC and deployed them in production between May 2022 and January 2023.

4.2.1 Data and Feature Extraction

First, we build EMR data extracts containing medical records of all inpatients from HHC over a four-year period. The data set includes tables for demographics, patient status (e.g., oxygen device), clinical measurements (e.g., blood pressure), laboratory results, diagnoses, orders, procedures, notes, and others. We create a feature space where each row represents each patient day (in total 1,375,215 patient days). Given these raw tabular data files, we perform several data processing steps to obtain the final feature space, as described below.

String Parsing Some columns in string format require string parsing to extract numerical features as continuous variables. For instance, the normal ranges of laboratory tests in forms such as "50–70" are replaced with two columns: one with a value of 50 for the lower bound and another with a value of 70 for the upper bound.

Categorical Encoding Categorical columns (e.g., department, mobility level, the reason for visit) must be converted to ordered numerical levels (consecutive integers) using label-encoding or binary categories using one-hot encoding. Due to the large number of categories, we use label encoding for all categorical variables.

Feature Engineering To better capture the clinical information, we compute various auxiliary variables:

1) Current conditions extracted from records (e.g., whether the patient is in ICU or IV)

2) Normal indicators (whether the clinical measurement is within the normal/critical range) instead of the ranges themselves.

3) Counts (e.g., number of days in ICU, number of attending physicians).

4) Pending procedures/results (time until surgery, whether MRI is pending, etc.).

5) Historical record linked to the patient (e.g., number of days since the previous admission and length of the previous stay).

6) Non-patient-specific operational variables (e.g., day of the week, ward census and utilization, hospital admission volume on the previous day).

Missing Data Imputation Since the raw data comes from a hospital system, it contains many missing values. We impute most missing entries with 0, except for a few cases. From communications with the hospital, we impute certain variables with prior knowledge of the meaning of missingness (e.g., missing Do Not Resuscitate (DNR) means the patient did not sign a DNR form). For some auxiliary variables, we apply some rules, such as imputing counts with 0 if no record exists and imputing the number of days since previous admission with a large number (e.g., 9999) if no previous admission exists.

4.2.2 Machine Learning Models

Prediction Tasks We consider several binary classification tasks related to the length of stay, ICU, and mortality for each inpatient and on each day in the hospital. Two discharge-related outcomes are whether patients are discharged or not within the next 24 hours (resp. 48 hours). Four ICU-related outcomes include whether the patient will enter (resp. leave) the intensive care unit (ICU) for patients currently not in the ICU (resp. in the ICU) within the next 24 hours (resp. 48 hours). Two short-term expiration outcomes concern whether each patient will die in the next 24 hours (resp. 48 hours). One end-of-stay mortality outcome indicates whether patients die or not at the end of their stay.

Models We evaluate a variety of machine learning models to make predictions, including Optimal Classification Trees (Bertsimas and Dunn 2017), sparse classification (Bertsimas et al. 2021c), XGBoost, LightGBM (Ke et al. 2017), and Tabnet (Arik and Pfister 2021).

With the highest performance, XGBoost (multi-class and binary class) classification models are trained for each prediction task for each hospital and tuned with hyper-parameters using a validation set (chronologically splitter). Furthermore, we ensure that all our models are well-calibrated, by using the isotonic regression method (Zadrozny and Elkan 2002) to calibrate the models on the first half of the testing set and assessing the final calibration on the second half.

Predictive Analytics for Decision-Making It can be difficult to grasp the implications of raw probability scores and use them efficiently for decision-making. To turn these predictive analytics into a decision support tool that is sustainably used by practitioners, we complement the predictions with a color-coded alert system. We send green alerts for patients who are ready for short-term discharge (probability of 24hr or 48hr discharge is above certain thresholds). On the other hand, we send red alerts to warn about patients who have a high risk or are exacerbating (mortality risk or the increase of mortality risk from the previous day reaches certain thresholds).

Model Evaluation All models achieve high out-of-sample AUC (75.7%–92.5%) across all prediction tasks and are well calibrated for all seven hospitals. After threshold tuning and discussions with doctors, we select to send a green alert when the 24-hr discharge probability exceeds 0.25 or the 48-hr probability exceeds 0.4, which gives 0.621 precision and 0.746 recall. On the other hand, we raise a red alert when the mortality risk exceeds 0.2 or when its absolute change compared with the previous day exceeds 0.1, which gives 0.477 precision and 0.705 recall.

4.2.3 Results

To evaluate the empirical impact of the tool, we consider a treatment group of 15 HHC units that used our tool at varying degrees between 2022 and 2023, and a control group of 12 units that had not yet fully incorporated the tool as of April 15, 2023. To estimate the effect of

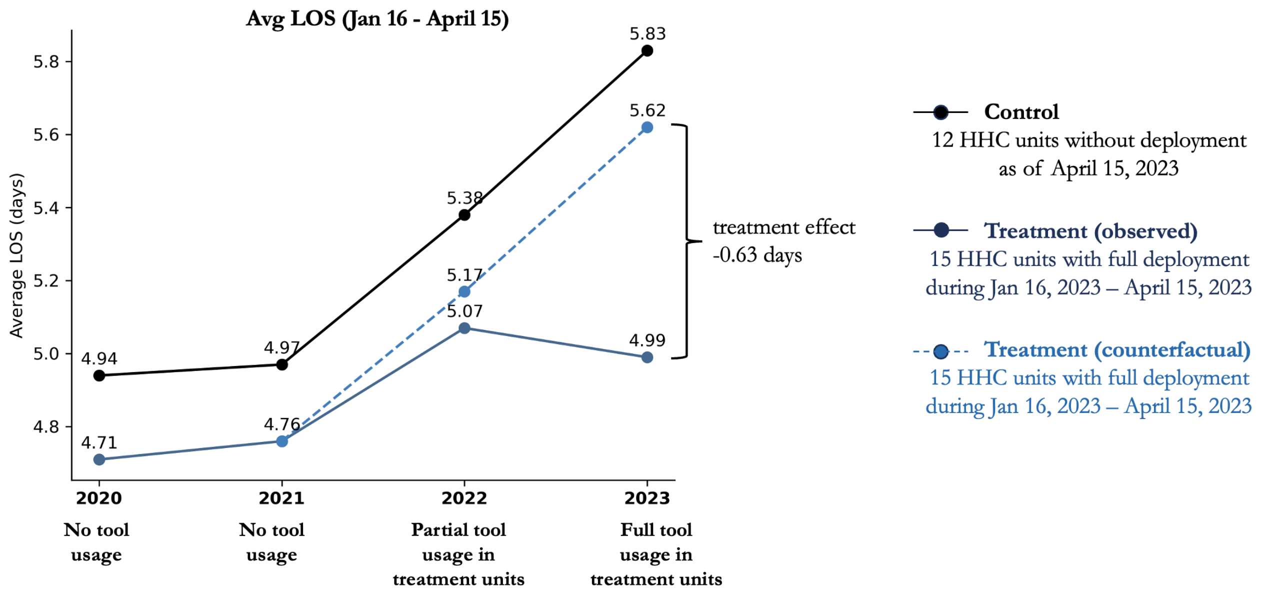


Figure 4.1: Empirical Analysis for Treatment Effect on Length of Stay. All the units in the control and treatment groups are medicine or cardiology units offering general level of care.

our tool, we use a Difference-in-Differences (DiD) technique (Abadie 2005, Bertrand et al. 2004) and compare the average change in LOS among patients discharged in the treatment group to that in the control group. We control for similar population fixed effects and time non-stationarity effects, as we cover units of the same level of care and specialty, and the same January 16 - April 15 period over the past three years. We assume that the difference in LOS over time would have been the same between the two groups if the tool had not been used (parallel trend assumption). We use this assumption to impute the counterfactual average LOS for the treatment group, had there been no treatment (light green dashes in Figure 4.1).

The LOS of the control group showed a steady increase after 2020, rising from 4.97 in 2021 to 5.38 in 2022 and eventually reaching 5.83 in 2023. Between 2021 and 2022, the treatment group's LOS increased from 4.76 to 5.07, which was in line with the parallel trend but slightly lower, potentially due to the pilot's partial treatment effect. After full deployment, the treatment group's LOS dropped to 4.99 in 2023, while the control group's LOS continued to rise from 4.96 to 5.85. The difference between the parallel counterfactual and actual treatment group's LOS resulted in an estimated benefit of reducing the average

LOS by 0.63 days per patient.

By reducing the average LOS by 0.63 days among patients in the 15 treatment units (49,424 patients annually), we can save 31,137.12 patient days. If all beds are backfilled, this would make room for an additional 6,239.9 patients per year, leading to a projected total annual contribution margin increase of $67,365,60.4. Alternatively, under a no backfill scenario, at an average direct cost for a medical/surgical inpatient of $1,661 per patient day, the LOS reduction would be translated into estimated annual savings of $51,718,76. Therefore, in practice, HHC is projected to obtain annual financial benefits in the range of $51–$67 million, with some beds backfilled and others not.

To support these observations, we conduct a DiD regression analysis with variations in treatment times (Callaway and Sant'Anna 2021, Goodman-Bacon 2021) in Appendix Section C.3, which confirms a significant reduction in average LOS (and its logarithm) of similar magnitude (see Table C.4). In addition, we observe a significant reduction in the time between the green alert and the discharge order placed by physicians, supporting the hypothesis that the reduction in LOS in the treated units is partially due to physicians better anticipating the administrative process associated with discharges.

The main strength of our empirical validation is its multi-center nature, unlike the simple before-and-after evaluation of prior work from the literature (Bertsimas et al. 2021b). We also control for time trends and seasonality (time fixed effects) and unit/hospital heterogeneity. However, the main limitation comes from the staged roll-out design over which we had no control and which can be lead to estimation biases (Baker et al. 2022). In addition, a small number of physicians in the control units had access and used the tool. We considered these units as control nonetheless, which could lead to a conservative estimation of the effect.

4.3 TabText

As observed in the previous section, traditional machine learning approaches using tabular data typically require thorough data cleaning before data is input into the models. Standard model development requires a series of data pre-processing steps, including merging raw data tables, parsing string columns, encoding categorical variables, constructing features, and imputing missing data, as described in Section 4.2.1. In fact, in our collaboration with HHC it took approximately one year of joint effort by machine learning researchers and hospital specialists to obtain, clean, and process the data.

TabText is a new feature extraction methodology to represent contextual information from tabular sources, which can replace data cleaning techniques or serve as a method for data augmentation. We process tabular data by creating a text representation for each data sample. This text contains the column attribute with its corresponding value and potentially other available contextual information. We then use this text as input for a finetuned pre-trained model that generates TabText embeddings of a fixed dimension. Finally, we augment the tabular features with these TabText embeddings to train any standard machine learning model for downstream prediction tasks.

The overall TabText framework can be visualized in Figure 4.2.

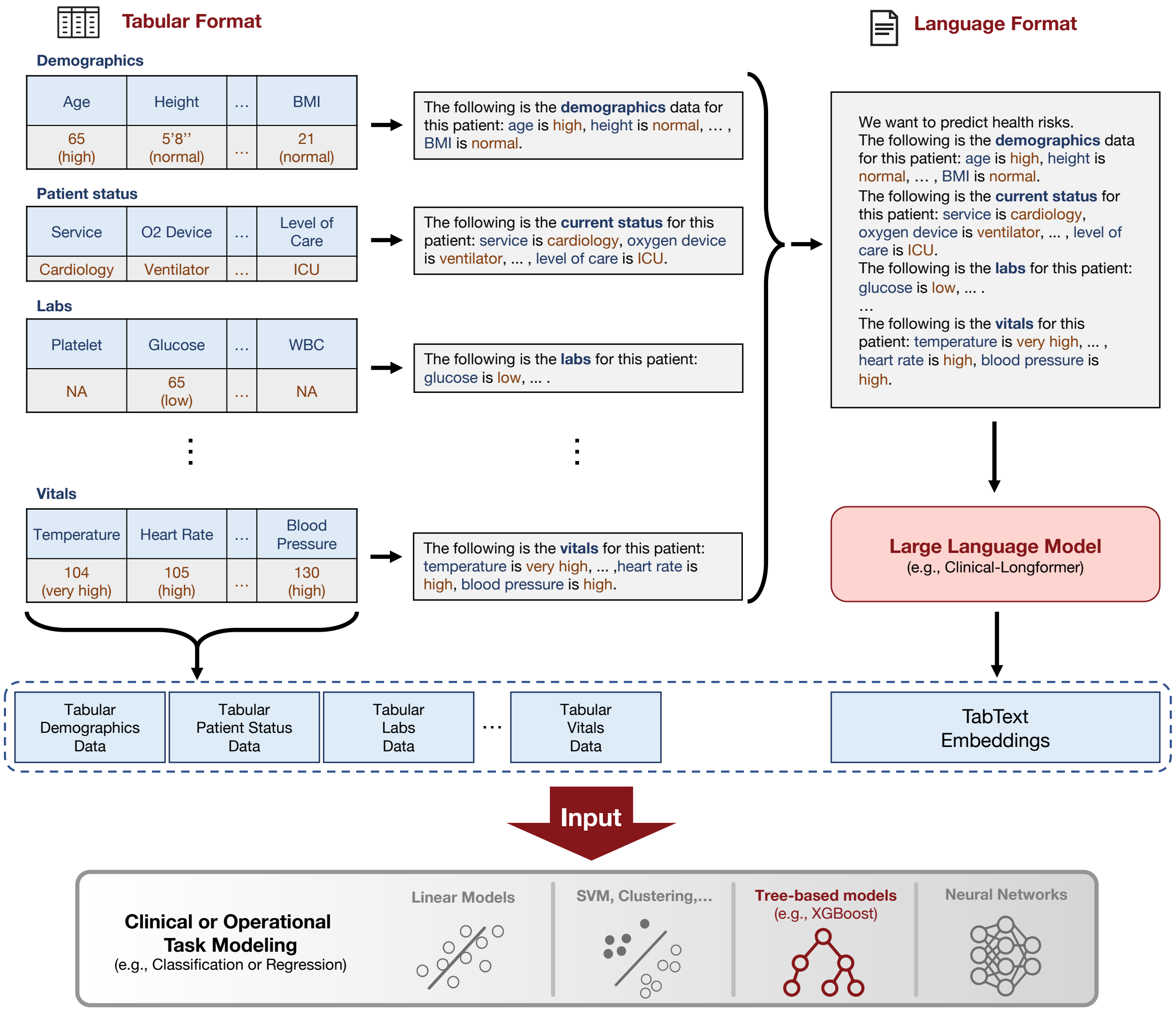

Tabular Format
Demographics
Age | Height | … | BMI
65 (high) | 5'8'' (normal) | … | 21 (normal)
Patient status
Service | O2 Device | … | Level of Care
Cardiology | Ventilator | … | ICU
Labs
Platelet | Glucose | … | WBC
NA | 65 (low) | … | NA
Vitals
Temperature | Heart Rate | … | Blood Pressure
104 (very high) | 105 (high) | … | 130 (high)
The following is the demographics data for this patient: age is high, height is normal, … , BMI is normal.
The following is the current status for this patient: service is cardiology, oxygen device is ventilator, … , level of care is ICU.
The following is the labs for this patient: glucose is low, … .
The following is the vitals for this patient: temperature is very high, … ,heart rate is high, blood pressure is high.
Language Format
We want to predict health risks. The following is the demographics data for this patient: age is high, height is normal, … , BMI is normal. The following is the current status for this patient: service is cardiology, oxygen device is ventilator, … , level of care is ICU. The following is the labs for this patient: glucose is low, … . … The following is the vitals for this patient: temperature is very high, … , heart rate is high, blood pressure is high.
Large Language Model
(e.g., Clinical-Longformer)
Tabular Demographics Data
Tabular Patient Status Data
Tabular Labs Data
…
Tabular Vitals Data
TabText Embeddings
Input
Clinical or Operational Task Modeling
(e.g., Classification or Regression)
Linear Models
SVM, Clustering,…
Tree-based models
(e.g., XGBoost)
Neural Networks

Figure 4.2: End-to-end TabText framework.

4.3.1 Methodology

As part of our methodology, we need to answer three main questions: 1) which LLM we are using, 2) how we are constructing the language data, and 3) if we are fine-tuning the pretrained model or not.

To address each one of those questions we use a data set from a large teaching hospital over a three-month period, where each data point represents a patient day. There are 160 columns of different patient attributes on demographics, patient status, vital signs, laboratory results, diagnoses, treatments, and other information. The summary of the tables utilized can be found in Table 4.1.

Table #	Table Meta Information	Example Columns
1	Lab values	Platelet, Sodium
2	Chart measurements	Respiratory rate, oxygen concentration
3	Counting statistics	Number of medications, number of orders
4	Current condition	Oxygen device, is in ICU
5	Historical patient record	Previous admission, previous length of stay
6	Non-patient-specific data	Day of the week, ward census

Table 4.1: Summary of tabular data, which contains different aspects of a patient's admission stay from patient's high-level demographics to precise lab measurements.

LLM Selection We consider two different Transformer models, BioGPT and Clinical-Longformer (Li et al. 2022), both of which were pre-trained with MIMIC-III clinical notes (Johnson et al. 2016). Following the TabText framework, we convert the tables into simple text: for each row, the cell from column "attribute" with value X is transformed into "attribute: X" and the texts from all columns are concatenated into a single sentence with the comma character. We next create TabText embeddings and finally train gradient-boosted tree models. We use $60,000$ data samples for training and validation, and $10,000$ for testing. Figure 4.3a shows the boxplots for the out-of-sample AUC over 10 random 75%-25% train-validation splits for each task and each model. Both NLP models achieve similar performance across tasks, and we therefore choose the Clinical-Longformer model, as it allows for input text of larger size.

Language Construction The versatility of language creates a challenge for consistency, as multiple textual expressions can convey the same information. Moreover, tabular data in healthcare is often split across multiple tabular sources (e.g., vitals table, medications table), some of which include information only for a particular group of patients. This results in even more possibilities for textual representation.

The TabText framework creates a single paragraph for each data sample (e.g., for each patient day) as follows: we first create a sentence for each column in each table. Next, for each table, we concatenate contextual information and the sentences of its columns using the colon (":") and comma (",") characters, respectively. We then merge the text from all tables into a single paragraph using the period (".") character. While the exact punctuation doesn't significantly impact BERT-based transformers (Ek et al. 2020), the exact text chosen to build each sentence might have a larger impact on the final embedding. We therefore investigate different ways to construct sentences for each column attribute.

Descriptiveness: We consider whether or not to use descriptive language to construct text sentences. Specifically, consider a cell from column "attribute" that has value "X". If the column is non-binary, we consider the following options:

- Non-Descriptive Sentence: "attribute: X";
- Descriptive Sentence: "attribute is X".

For binary columns, we consider the verb associated with the specific attribute. For instance, if the column attribute is associated with the verb "to have" we consider

- Non-Descriptive Sentence: "has X: yes" or "has X: no";
- Descriptive Sentence: "has X" or "does not have X".

Missing Values: When the value for a column "attribute" is missing, we consider two options, to explicitly mention in the text that this information is not available ("attribute is missing"), or to simply skip this column when building the text representation.

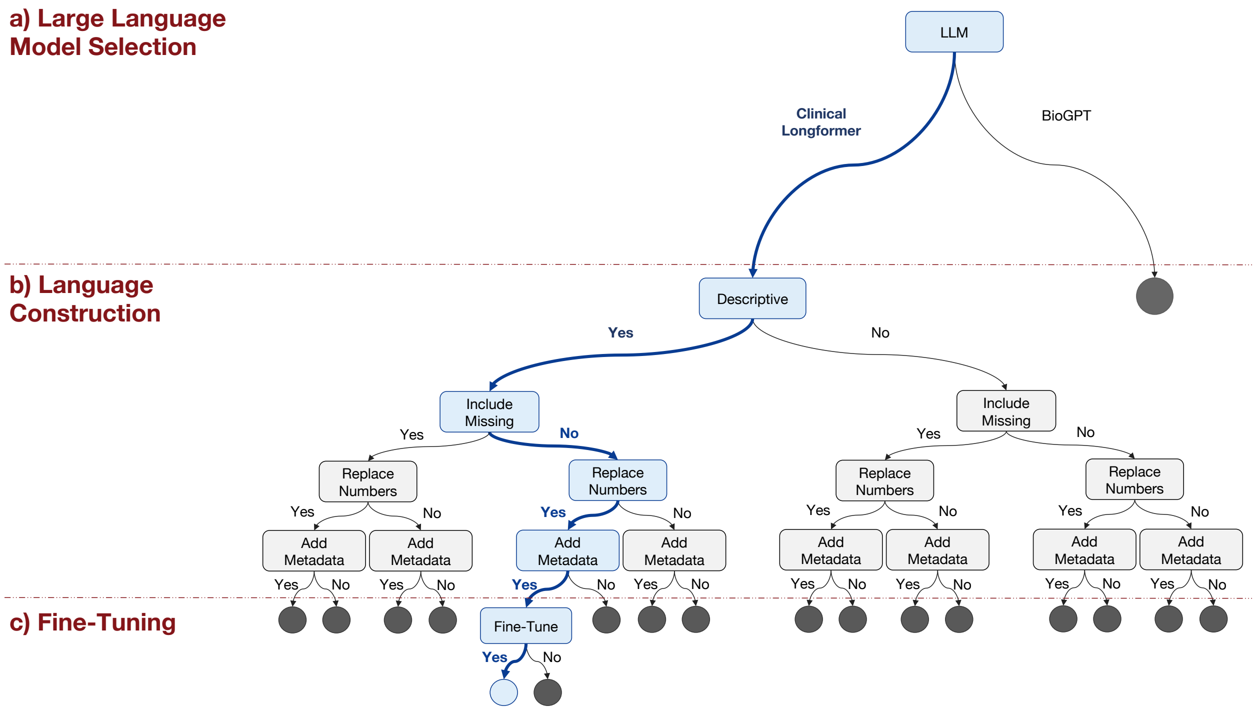
Numerical Data: Transformer models often struggle to represent language with numerical data (Gorishniy et al. 2022). Therefore, we also consider whether or not to replace numerical values with text. For replacement, we compute the average (AVG) and standard deviation (SD) of the corresponding column with respect to the training data. We then replace a given cell value X as follows:

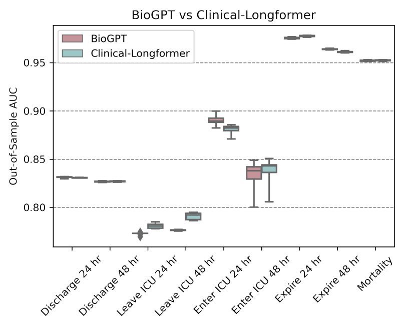
- "very low" if $X <$ AVG $-$ 2SD;
- "low" if AVG $-$ 2SD $\leq X <$ AVG $-$ SD;
- "normal" if AVG $-$ SD $\leq X <$ AVG $+$ SD;
- "high" if AVG $+$ SD $\leq X <$ AVG $+$ 2SD;
- "very high" if AVG $+$ 2SD $< X$.

Including Metadata: We investigate the added value of including metadata as part of the text representation. This corresponds to descriptions of table content (e.g., "This table contains information about the medications administered to this patient") or the prediction task of interest (e.g., "We want to predict mortality risk").

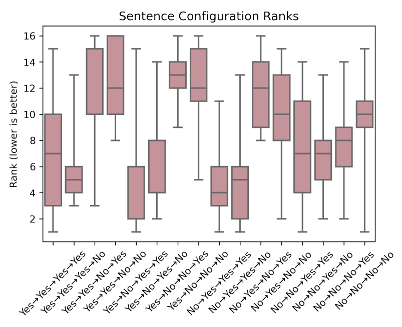
For each possible sentence configuration, we use default values of the Clinical-Longformer model to obtain TabText embeddings that are given as input to a gradient-boosted tree model. For this small experiment, we utilize 63 data features corresponding to laboratory results. We use $60,000$ data samples for training and validation, and $10,000$ for testing. In Figure 4.3b, the Language Construction results show the boxplot for the rank achieved with each configuration across tasks, where lower numbers correspond to better ranking. We choose the sentence configuration with the lowest median ranking; specifically, we use descriptive language, omit missing values from the text, replace numerical values with text, and include metadata.

Fine-Tuning Although Clinical-Longformer was pre-trained with large language data sets, we can further improve its performance with a few more training iterations using our training data. Specifically, we convert our training data into language following the sentence configuration selected in Section 4.3.1, and we use it to fine-tune Clinical-Longformer following the original BERT training methodology, which includes self-supervised masked word prediction. We fine-tune for 3 epochs and with the default values for all hyperparameters. We then generate embedings that are given as input to a gradient-boosted tree model, using $60,000$ data samples for training and validation, and $10,000$ for testing. We show in Figure 4.3 the boxplots for the out-of-sample AUC over 10 random 75%-25% train-validation splits for each task. We see that fine-tuning the model with our local data slightly improves performance for eight out of the nine classification tasks of interest.

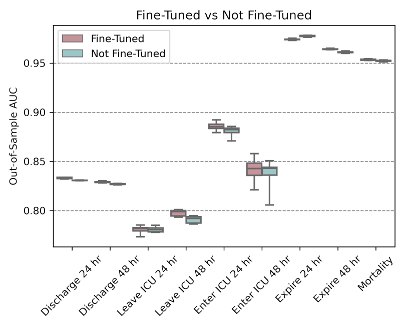




(a) LLM Selection.



(b) Language Construction.



(c) Fine-Tuning.

Figure 4.3: Overview of our overall methodology. We start with the selection of an LLM. Figure (a) shows that BioGPT and Clinical Longformer achieve similar results, and we therefore choose Clinical Longformer, as it allows for input texts of larger sizes. Notice that the specific LLM can flexibly be replaced as novel models become available. Then, we look for the best language representations of the original patient data. In Figure (b) we observe the boxplots of the ranks for different sentence configurations, which were tested across different prediction tasks (lower rank is better). We select the configuration with the lowest median ranking; specifically, we use descriptive language, omit missing values, replace numerical values with text, and include metadata. Lastly, we fine-tune the LLM using this sentence configuration, as it leads to better performance as shown in Figure (c).

4.4 TabText Results

This section presents extensive computational experiments evaluating the performance of our TabText framework. First, we show how our pipeline can quickly generate machine-learning models with competitive performance without any data cleaning by leveraging the flexibility of language. We then demonstrate with pre-processed data that augmenting standard tabular representations with our TabText embeddings can increase out-of-sample AUC by up to 6%, with the largest improvements observed for the most challenging predictions.

Data: For the final experiments we consider a large data set from the same teaching hospital used in the previous section, with inpatient data for the four years following the three-month period used in Section 4.3.1. we summarize the number of data points for each prediction task in Table 4.2.

Prediction Task	**Training**	**Testing**
Discharge 24 hr	572,964	265,917
Discharge 48 hr	572,964	265,917
Enter ICU 24 hr	385,132	180,075
Leave ICU 24 hr	73,013	34,669
Enter ICU 48 hr	292,659	138,947
Leave ICU 48 hr	68,472	33,011
Expire 24 hr	572,964	265,917
Expire 48 hr	572,964	265,917
Mortality	572,964	265,917

Table 4.2: Data sizes (number of patient days) for training and testing sets across the nine healthcare classification tasks.

Text Encoder: We first convert the input training data from tabular to textual format as described previously in Section 4.3.1. We use the sentence configuration that led to the highest average AUCs (i.e., skipping sentences for missing values, replacing numbers with text, using descriptive language, and adding metadata). Then, we use the fine-tuned Clinical-Longformer model to extract language embeddings of size 768.

Training Methodology: For each prediction task, we compare two approaches: our TabText framework (see Figure 4.2) and the standard Tabular approach in which only the tabular data is given as input to the machine learning model. We use gradient-boosted tree models in all experiments performed. For all reported results, the average performance is computed over 10 random 75%-25% train-validation splits (identical 10 splits across all experiments) for a fair comparison. The optimal model is selected using a hyperparameter grid search (see details in the appendix) based on its performance on the validation set. The hyperparameters that we grid-searched to obtain the optimal XGBoost model are:

- Number of estimators: {100, 200, 300},
- Maximum depth: {3, 5, 7},
- Learning rate: $\{0.05, 0.1, 0.3\}$,
- L_2 regularization parameter: $\{1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}, 0\}$.

Implementation: All our code is written in Python 3.8.2. We trained all models using one Intel Xeon Platinum 8260 or Intel Xeon Gold 6248 CPU and GPU. We conducted all of our predictive experiments using the XGBoost (Chen and Guestrin 2016) library from Python. The Clinical-Longformer model is directly accessed from HuggingFace.

4.4.1 High Performance with Minimal Pre-Processing

The TabText framework can be leveraged to replace heavy data cleaning by simply creating a text representation for each data sample using the information as it appears in the raw data tables. In particular, columns that require data cleaning to be converted to appropriate data types can be simply transformed into text. For example, the sentence corresponding to a numerical column for a sedation score with the value "-4 $\rightarrow$ deep sedation" can be written as "sedation score is -4 $\rightarrow$ deep sedation", as opposed to parsing the original string into a numeric value of -4 as part of the traditional pre-processing steps. Therefore, TabText representations enable us to quickly build baseline machine learning models utilizing the tabular data in its raw form.

We predict the same nine classification tasks described in Section 4.2.2 using the raw tables without data cleaning. Only minimal data preprocessing was required, including constructing the meta information of the tables and categorizing columns for different language representations, which is estimated to have taken only a couple of hours of manual work. We then followed our TabText pipeline to train a gradient-boosted tree model for each classification task. As shown in Table 4.3, the baseline TabText models with minimally processed data already achieve high out-of-sample AUC performance, where the average AUCs across 10 random 75%-25% train-validation splits are close or above 0.8 for all prediction tasks except for Enter ICU 48 hr, which is a notoriously difficult classification task (Na et al. 2023).

Prediction Task	**TabText Baseline AUC**
Discharge 24 hr	0.803
Discharge 48 hr	0.790
Enter ICU 24 hr	0.801
Leave ICU 24 hr	0.839
Enter ICU 48 hr	0.757
Leave ICU 48 hr	0.835
Expire 24 hr	0.943
Expire 48 hr	0.933
Mortality	0.895

Table 4.3: Out-of-sample average AUCs achieved by baseline TabText models with minimally processed data and across 10 random train-validation splits. All models are highly accurate, reaching practically implementable benchmarks in hospital systems.

4.4.2 Enhanced Performance with Contextual Representation

We next process the data following the same cleaning steps as in Section 4.2.1 and feed into the TabText Framework from Figure 4.2. We perform experiments on the same data and classification tasks as in Section 4.4.1 but using the cleaned data this time. The results obtained using the standard Tabular approach and our TabText framework are shown in Table 4.4. The average AUCs across 10 random 75%-25% train-validation splits for the Enter ICU 48 hr and Discharge 48 hr prediction task are improved by an additive increment of 1.2%–1.4%. We also see a substantial but smaller benefit for Mortality risk prediction. For the remaining tasks, Tabular and TabText achieve similar performance with differences in average AUC smaller than 0.25%. We also notice in Figure 4.4 that the largest TabText benefits occur for the classification tasks with the lowest Tabular performance (high variability and low AUCs), while practically no effect was observed for the tasks with stable Tabular results (low variability and high AUCs).

4.4.3 Larger Benefits for Harder Predictions

To better understand the regimes in which TabText provides the largest improvements in AUC performance, we repeat this experiment using smaller and larger training data sets. For each prediction task, we consider the original training data size as well as smaller data sizes (ranging from 2000, 3000, 5000, 10000, 25000, and 50000 patient days). We plot in Figure 4.5a (resp. Figure 4.5b) the average (resp. worst-case) TabText AUC improvement percentage across 10 random 75%-25% train-validation splits, where the x-axis corresponds to the average (resp. worst-case) AUC of the standard Tabular approach and the y-axis quantifies the relative percentage improvement on average (resp. worst-case) AUC achieved with TabText. Each scatter point represents the result of a prediction task (denoted by legends) on one of the 7 different data subsets. As in Section 4.4.2, we observe larger improvements on the more difficult prediction tasks with Tabular AUCs below 85%. On easier prediction tasks, where

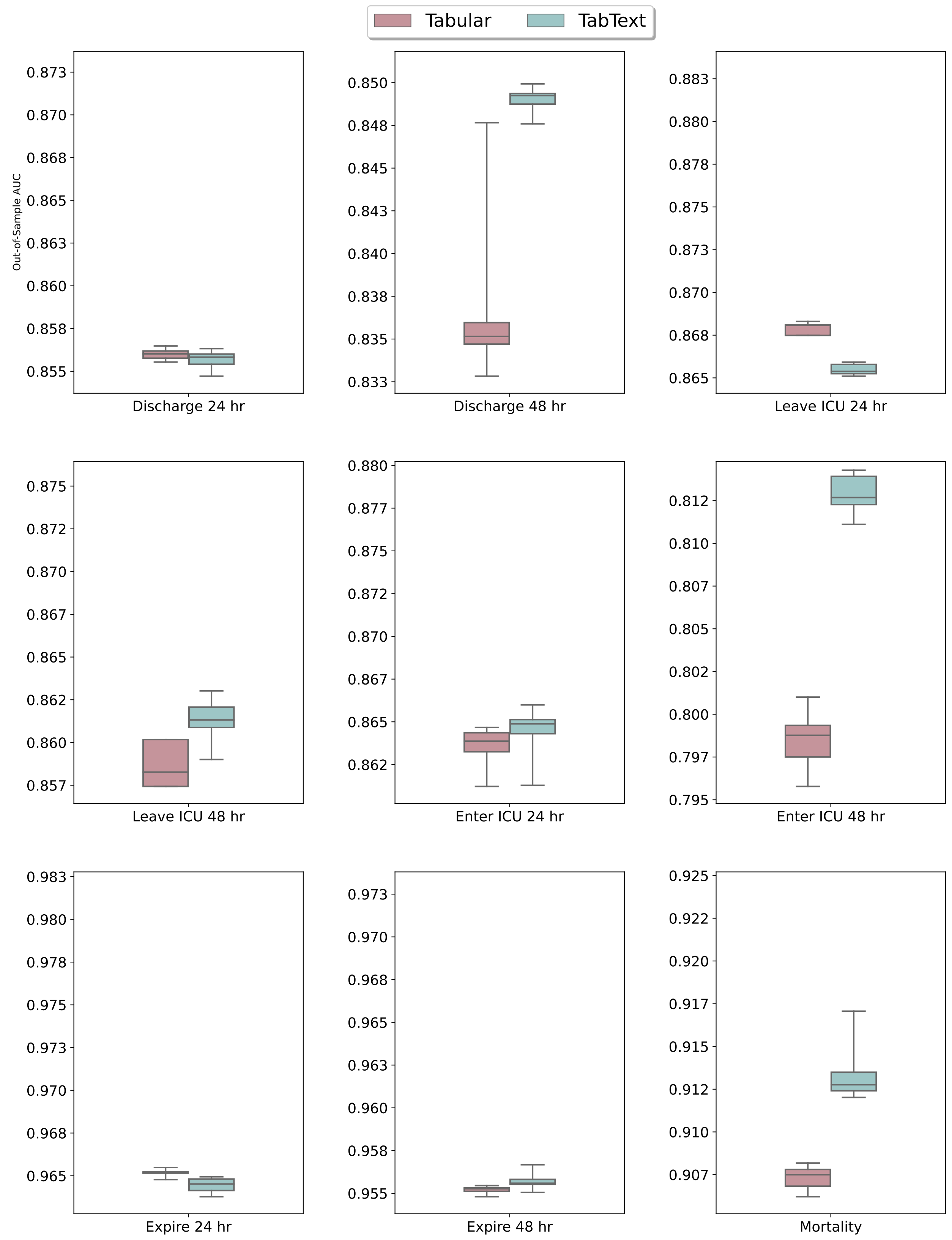
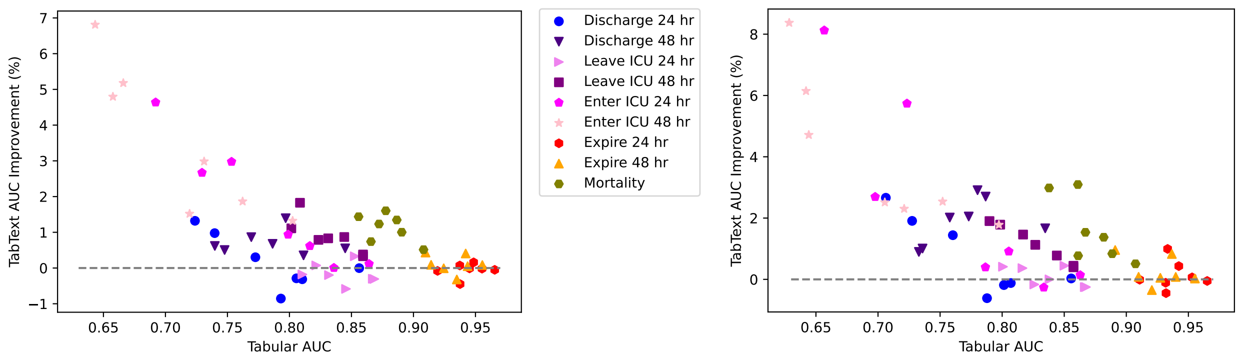


Figure 4.4: Boxplots for the out-of-sample AUCs across 10 random train-validation splits using Tabular vs. TabText models. We see that the largest TabText benefits occur for the classification tasks with high variability and low AUCs, while practically no effect was observed for the tasks with low variability and high AUCs.

Tabular models already achieve AUCs over 90%, the benefit of TabText is near or below zero. When the Tabular AUCs are less than 78%, Tabtext brings a positive improvement on all results, including several instances of improvement over 5–6%. This suggests more potential benefits of augmenting tabular models with TabText representations for tasks with low Tabular performance.



(a) Average out-of-sample AUC improvement at varying data sizes.

(b) Worst-case out-of-sample AUC improvement at varying data sizes.

Figure 4.5: TabText AUC improvement over the standard Tabular approach at varying data sizes. We observe that the improvement of TabText is most prominent when standard Tabular models do not perform well. For high-performing tasks, the advantage is less pronounced. An important implication of this observation is that challenging medical prediction tasks with a lack of difficult-to-observe risk factors or a small sample size can benefit from our framework.

4.5 Conclusion

We first developed a system of machine learning models predicting short-term discharge, ICU risk, expiration, as well as end-of-stay mortality. All models achieve state-of-the-art predictive accuracy. These models are deployed in 7 hospitals and used by over 200 medical staff at HHC, who experienced first-hand benefits to shorten the length of stay, decrease the cost of care, enhance patient safety, and improve the overall patient experience. Empirically, we observe a reduction in average patient length of stay by 0.63 days and project an annual contribution margin increase of $67 million dollars.

Next, we introduce TabText, a novel framework for processing tabular data by converting it into a text representation that captures important contextual information such as column descriptions. Our experiments show that augmenting standard tabular data with our TabText representations can improve the performance of standard machine learning models across all healthcare predictions tasks considered, with larger improvements observed for the more challenging tasks. In addition, we demonstrate the efficiency of TabText in simplifying data pre-processing and cleaning, offering an alternative and flexible pipeline for generating high-performing baseline models for hospitals that have small number of patients or non-standardized, disorganized medical records.

Our experiments reveal the potential of TabText for improving the performance of standard machine learning models, and there are several research directions for further improving our framework. For instance, TabText relies on the use of NLP models that could generate high-quality embeddings for the input text, which motivates the development of more LLMs pre-trained with domain-specific data. Moreover, augmenting tabular data with TabText embeddings adds a layer of complexity to the interpretability of the model output, and developing tools for maintaining the interpretability of the tabular data would be an interesting direction for future research. TabText is a general framework that can be particularly useful for difficult classification tasks (for instance, those with limited data availability), and we hope that this work motivates further research for leveraging language in general machine learning applications.

Chapter 5

Multistage Stochastic Optimization via Kernels

5.1 Introduction

Multistage stochastic optimization arises in numerous applications (e.g., supply chain management, energy planning, inventory management among others) and remains an important research area in the optimization community (Birge and Louveaux 2011, Shapiro et al. 2014, Bertsimas et al. 2011). In these problems, the decision variables are split across multiple periods and decisions are made sequentially as more information becomes available. The goal is to make high quality decisions that minimize the expectation of a given cost function by accurately modeling future uncertainty. In practice, decision makers can use historical data to get a sense of the future uncertainty. For instance, consider a retailer selling products with short life cycles who needs to make frequent orders to restock inventory without knowing the future demands. To minimize costs the retailer must use the remaining inventory quantities as well as historical data to gain insight into future demands. Another example is energy planning, in which operators decide daily production levels without knowing how weather conditions will affect the output of the wind turbines. In this case historical wind patterns are valuable for better planning.

Besides historical data, auxiliary covariates are often available and can help predict uncertainty. For example, in the fashion industry, color and brand are useful factors to predict demand of a new item. Accordingly, recent work has focused on using predictive analytics to leverage available side information and historical data to make better decisions. Ban et al. (2019) for instance, fit covariate and historical data to a regression model and prove theoretical guarantees for the dynamic procurement problem. Another approach is that of Bertsimas et al. (2022c), which considers an uncertainty set around each data sample and applies robust optimization tools to find linear decision rules that are asymptotically optimal under mild conditions. This framework was generalized in Bertsimas and McCord (2019), where machine learning methods are incorporated to find weights that produce more accurate approximations of the objective. However, these dynamic methods are affected by the curse of dimensionality; they require scenario tree enumeration and can require many hours for solving problems with only a few stages.

In this paper, we propose a non-parametric, data-driven and tractable approach to solving multistage stochastic optimization problems. By restricting the decision variables to be in a reproducing kernel Hilbert space (RKHS) generated by a universal kernel, we can approximate a large class of functions using non-parametric functional representations. We incorporate sparsification techniques based on function subspace projections that allow our proposed algorithm to overcome the complexity growth that kernel methods introduce when directly applying the Representer Theorem to large data sets. The input to our algorithm is historical data and we make no assumptions on the correlation structure of the uncertainties across stages. We perform computational experiments on real-world multistage stochastic problems, and show how our method not only produces near optimal solutions but also remains tractable in higher dimensions and with large data sizes.

5.1.1 Related Literature

Kernel methods have been used in recent work to solve stochastic multistage optimization problems with side information. Hanasusanto and Kuhn (2013), for example, approximate the objective using kernel regression, and Pflug and Pichler (2016) apply a kernel density estimator to the historical data to develop a non-parametric predict-then-optimize approach that comes with asymptotic optimality guarantees under strong conditions. However, these are local methods in which the predictions are made based only on those data points that are similar to the current observation. As noted in Bertsimas and Koduri (2022), such approaches require more data and perform worse on high dimensions compared to global methods, which instead optimize over functional variables that make the predictions.

The Machine Learning community has long applied kernel methods to solve online learning problems (Wheeden 2015, Norkin and Keyzer 2009), but they have focused purely on predictive and not on prescriptive tasks. More recently, Bertsimas and Koduri (2022) has aimed to extend kernel methods to data-driven, single-period optimization problems with auxiliary information by using the Representer Theorem to transform the optimization over functions into an optimization over parameters. They show that this approach overcomes the curse of dimensionality; however, its main disadvantage is that the number of parameters per decision grows linearly with the number of observations, resulting in function representations that are as complex as the size of the data and that become potentially intractable especially in multistage settings.

Works on stochastic optimization in a RKHS have developed multiple heuristics to reduce the number of parameters in the function representation. For instance, Zhang et al. (2013) uses random dropping, Kivinen et al. (2004) introduces forgetting factors, and Honeine (2011) as well as Engel et al. (2004) apply compressive sensing techniques. These approaches sucessfully achieve sparser functional representations, but they usually produce suboptimal approximations (Honeine 2011, Engel et al. 2004).

We instead follow the approach from Koppel et al. (2016) of applying Functional Stochastic Gradient Descent (FSGD) and projecting the iterates onto sparse subspaces that are found by removing parameters associated with data points that do not contribute much to the value of the decisions (Pati et al. 1993). This approach maintains optimality while addressing the complexity growth that kernel methods exhibit as the data size increases. Intuitively, since stochastic gradient descent iterates are a noisy signal for the optimal solution, by projecting the iterates to have small model order we can ignore some of the noise while preserving the goal signal. The sparse subspaces of the RKHS onto which projections are made can be effectively found using kernel orthogonal matching pursuit (Vincent and Bengio 2002), an algorithm which given a function f and an error bound ϵ, generates a sparse approximation of f that is in a neighborhood of f of radius ϵ in Hilbert norm. Koppel et al. (2016) show that for a specific choice of ϵ and of step-size for the FSGD algorithm, the projected FSGD iterates produce decisions that converge in mean to the optimal solution.

5.1.2 Contributions

In this paper, we propose a novel data-driven approach for solving multistage stochastic optimization problems with side information using kernels. Specifically, we represent the controls as elements of a reproducing kernel Hilbert space and use loss-minimizing machine learning methods to predict them. In addition, we incorporate sparsification techniques to reduce the total number of parameters per control. We prove that this approach is asymptotically optimal, guaranteeing near optimal approximations for problems with large amounts of data. We also show that our approach remains computationally tractable in high dimensions and with large data sizes. In detail, our contributions are as follows.

1. We propose a novel data-driven approach for multistage stochastic optimization problems with side information based on reproducing kernel Hilbert spaces. The approach takes as input historical data and minimizes the regularized empirical loss by applying functional stochastic gradient descent to optimize the decision rules, i.e., the functions

which specify what decision to make in each stage. To the best of our knowledge, this is the first tractable application of reproducing kernel Hilbert spaces to multistage optimization problems with large data sizes. While a kernel based formulation of the multistage stochastic optimization problem is briefly suggested (without any computational experiments) in Bertsimas and Koduri (2022), their non-stochastic and non-sparse approach is not tractable for large data sizes since both time and memory requirements increase cubically with the amount of data.

2. We extend sparsification techniques used by Koppel et al. (2016) to multistage optimization settings in order to reduce both, space and time complexities of our algorithm. Specifically, we use Functional Stochastic Gradient Descent (FSGD) to minimize the objective and project each iterate onto a sparse subspace that is found by removing parameters corresponding to data points with small contributions. We show that applying FSGD without any sparsification results in methods that do not scale to larger number of periods or data sizes. If sparsity is not added, the computational cost and the storage requirement increase quadratically with the data size. With the proposed method, however, both space and time complexities present linear growth with a constant factor that depends on the step size of the FSGD algorithm.

3. We prove that if the loss function is convex, Lipschitz and differentiable almost everywhere, then the expected loss achieved with our algorithm converges in probability to the expected loss of the optimal decision rules in the space of continuous functions.

4. We demonstrate across several instances of inventory management problems that the proposed method finds near-optimal solutions using only a few parameters and with very low computational times. We show that increasing the number of periods, the

dimension of the data, the dimension of the controls or the data size does not affect the tractability of our approach.

The paper is organized as follows: Section 5.2 introduces the exact framework for the problem being solved, Section 5.3 contains the data-driven formulation of the multistage stochastic optimization problem with side information, Section 5.4 presents the proposed algorithm, Section 5.5 states the convergence theorems, Section 5.6 analyses the complexity of the proposed method, and Section 5.7 shows the results for several computational experiments.

5.2 Problem Setting

We consider a discrete-time, convex, multistage stochastic problem over a finite horizon T. Initially, we observe some auxiliary covariates $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{q_0}$. Then, random disturbances $\boldsymbol{w}_t$ that belong to a known set $\boldsymbol{W}_t \subseteq \mathbb{R}^{q_t}$ are sequentially observed over time. At every stage t, after observing the covariates $\boldsymbol{x}$ and the previous disturbances $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{t-1})$, a decision $\boldsymbol{u}_t \in \mathbb{R}^{r_t}$ is made. The total cost for the observed sequence of covariates, disturbances and decisions is $c(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T, \boldsymbol{x}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_T)$.

A standard decision rule $\bar{\boldsymbol{u}}(\cdot) = (\bar{\boldsymbol{u}}_1(\cdot), \ldots, \bar{\boldsymbol{u}}_T(\cdot))$ consists of functions $\bar{\boldsymbol{u}}_t : \boldsymbol{W}_1 \times \ldots \times \boldsymbol{W}_{t-1} \to \mathbb{R}^{r_t}$ that at each stage t take as input the disturbances up to that point and output a decision for the given stage. Specifically, denoting $\boldsymbol{w} := (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T)$ and $\boldsymbol{w}_{1:t} := (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$, we have that the standard decision rule $\bar{\boldsymbol{u}}(\cdot)$ applied to $\boldsymbol{w}$ outputs $\bar{\boldsymbol{u}}(\boldsymbol{w}) = \big(\bar{\boldsymbol{u}}_1, \bar{\boldsymbol{u}}_2(\boldsymbol{w}_{1:1}), \ldots, \bar{\boldsymbol{u}}_T(\boldsymbol{w}_{1:T-1})\big)$. The multistage optimization problem over the space of continuous decision rules $\hat{\mathcal{F}}$ conditioned on some observed covariates $\boldsymbol{x}_0$ can then be written as

$$\min_{\bar{\boldsymbol{u}} \in \hat{\mathcal{F}}} \quad \mathbb{E}_{\boldsymbol{w}|\boldsymbol{x}} \left[c(\bar{\boldsymbol{u}}(\boldsymbol{w}), \boldsymbol{x}, \boldsymbol{w}) \mid \boldsymbol{x} = \boldsymbol{x}_0 \right], \tag{5.1}$$

where $c(\cdot)$ is a convex loss function. As noted in Bertsimas and Koduri (2022), the conditional problem in Eq. (5.1) can be formulated as an unconditional optimization problem by augmenting the domain of the decision rules to also take the covariates as input, and then evaluating the observed covariates in the decision rules found. In this paper, we adopt the same approach and therefore we consider augmented decision rules $\boldsymbol{u}(\cdot) = \big(\boldsymbol{u}_1(\cdot), \ldots, \boldsymbol{u}_T(\cdot)\big)$ with augmented domains $\boldsymbol{u}_t : \mathcal{X} \times \boldsymbol{W}_1 \times \ldots \times \boldsymbol{W}_{t-1} \to \mathbb{R}^{r_t}$. The augmented decision rule applied to the data point $\boldsymbol{w}$ with covariates $\boldsymbol{x}$ outputs

$$\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{w}) = \big(\boldsymbol{u}_1(\boldsymbol{x}), \boldsymbol{u}_2(\boldsymbol{x}, \boldsymbol{w}_{1:1}), \ldots, \boldsymbol{u}_T(\boldsymbol{x}, \boldsymbol{w}_{1:T-1})\big).$$

From now on we will join the covariates and the disturbances into a single random variable $\boldsymbol{z} \coloneqq (\boldsymbol{x}, \boldsymbol{w})$ to simplify notation, and we index $\boldsymbol{z}$ starting at time 0 instead of time 1, so that $\boldsymbol{z}_{0:t} \coloneqq (\boldsymbol{x}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$. Defining $\mathcal{F}$ as the space of continuous augmented decision rules, and $\mathcal{Z} \coloneqq \mathcal{X} \times \boldsymbol{W}_1 \times \ldots \times \boldsymbol{W}_T$, we obtain that solving Eq. (5.1) is equivalent to solving the problem

$$\min_{\boldsymbol{u} \in \mathcal{F}} \qquad \mathbb{E}_{\boldsymbol{z}}\big[c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big)\big] \tag{5.2}$$

and evaluating the optimal solution $\boldsymbol{u}^*(\cdot)$ at $\boldsymbol{x} = \boldsymbol{x}_0$ to obtain the standard decision rule $\bar{\boldsymbol{u}}^*(\boldsymbol{w}) = \boldsymbol{u}^*(\boldsymbol{x}_0, \boldsymbol{w})$.

5.3 Reproducing Kernel Hilbert space formulation for Multistage Optimization

We now propose a data-driven approach for multistage stochastic optimization problems with side information based on a Reproducing Kernel Hilbert space (RKHS). We include an overview of these spaces in Appendix D.1. We will assume that we have historical observations $\mathcal{S} = \{\boldsymbol{z}^n\}_{n=1}^N = \{(\boldsymbol{x}^n, \boldsymbol{w}_1^n, \ldots, \boldsymbol{w}_T^n)\}_{n=1}^N$ that are independently and identically distributed according to some unknown distribution. Let $K_t : \mathcal{X} \times \boldsymbol{W}_1 \times \ldots \times \boldsymbol{W}_{t-1} \to \mathbb{R}$ be a positive universal kernel and $\mathcal{H}_t$ the reproducing Kernel Hilbert space generated by K_t. We consider the Cartesian product Hilbert space, $\mathcal{H} \coloneqq \mathcal{H}_1^{r_1} \times \ldots \times \mathcal{H}_T^{r_t}$ with inner product defined by

$$\Big\langle\!\Big\langle\big((u_{1,1},\dots,u_{1,r_1}),\dots,(u_{T,1},\dots,u_{T,r_T})\big),\big((v_{1,1},\dots,v_{1,r_1}),\dots,(v_{T,1},\dots,v_{T,r_T})\big)\Big\rangle\!\Big\rangle_{\mathcal{H}}$$
$$:=\sum_{t=1}^{T}\sum_{i=1}^{r_t}\langle u_{t,i},v_{t,i}\rangle_{\mathcal{H}_t},$$

where $\langle u,v\rangle_{\mathcal{H}_t}$ corresponds to the inner-product between u and v with respect to the Hilbert space $\mathcal{H}_t$. We can approximate the solution of problem (5.2) by applying its empirical regularized version and restricting the decision rules to be in $\mathcal{H}$:

$$\min_{\boldsymbol{u}\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N} c(\boldsymbol{u}(\boldsymbol{z}^n),\boldsymbol{z}^n)+\frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2. \tag{5.3}$$

Even though problem (5.3) is not equivalent to problem (5.2), if λ vanishes with the data size then the regularized empirical average becomes a closer estimate of the expectation as N increases. We will then focus on solving problem (5.3), and later in Corollary 5.5.5 we show that as the data size goes to infinity, the expected loss converges in probability to the optimal solution of problem (5.2).

One way to solve the regularized empirical problem (5.3) is to use the multidimensional version of the Representer Theorem (Wahba 1990, Soentpiet et al. 1999, Schölkopf et al. 2002, Shawe-Taylor et al. 2004), which says that for each $t=1,\dots,T$ there exists a scalar matrix $\boldsymbol{A}_t$ such that the optimal solution to (5.3) satisfies

$$\boldsymbol{u}_t(\cdot)=\boldsymbol{A}_t\boldsymbol{K}_t(\boldsymbol{Z}_t,\cdot),$$

where $\boldsymbol{K}_t(\boldsymbol{Z},\cdot):=[K_t(\boldsymbol{z}^1,\cdot),\dots,K_t(\boldsymbol{z}^N,\cdot)]^T$, and the time subscript for a data matrix $\boldsymbol{D}=[\boldsymbol{d}^1,\dots,\boldsymbol{d}^N]$ refers to $\boldsymbol{D}_t=[\boldsymbol{d}^1_{0:t-1},\dots,\boldsymbol{d}^N_{0:t-1}]$. However, with this approach each decision $u_{t,i}$ has as many scalar parameters as data points, which generates both memory and performance problems as the number of data points becomes large. We instead want an algorithm for which more data yields better results overall, without increasing its complexity or worsening performance. General sparisification techniques like those found in Kivinen et al. (2004), Zhang et al. (2013) or Engel et al. (2004), successfully reduce the number of parameters; however they do so at the cost of compromising optimality. We therefore take the pruning approach developed in Koppel et al. (2016) to solve problem (5.3); we apply functional gradient descent to minimize the objective and at each iteration we drop those parameters that add near zero contribution to the value of the decisions, ensuring convergence to an optimal solution.

5.4 Sparse Multistage Optimization with Kernels

In this section, we extend sparsification techniques used by Koppel et al. (2016) to the multistage optimization setting described in the previous section in order to reduce both, space and time complexities of our algorithm. Specifically, we describe an iterative algorithm for solving (5.3) using Functional Stochastic Gradient Descent and sparse projections. In order to ease notation, we first make the following definitions for an augmented decision rule $\boldsymbol{u}$:

$$E(\boldsymbol{u}) \coloneqq \mathbb{E}_{\boldsymbol{z}}\left[c(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z})\right], \tag{5.4}$$

$$E^{\lambda}(\boldsymbol{u}) \coloneqq E(\boldsymbol{u}) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2, \tag{5.5}$$

$$E_{\mathcal{S}}^{\lambda}(\boldsymbol{u}) \coloneqq \frac{1}{N}\sum_{n=1}^{N} c\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2, \tag{5.6}$$

$$E_{n}^{\lambda}(\boldsymbol{u}) \coloneqq c\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2. \tag{5.7}$$

The algorithm relies on the fact that the expectation of $E_n^{\lambda}(\boldsymbol{u})$ over data yields $E^{\lambda}(\boldsymbol{u})$ to make stochastic gradient updates that converge to the optimal solution, while at the same time removing unnecessary parameters along the descent trajectory.

5.4.1 Functional Stochastic Gradient Descent (FSGD)

Thanks to the fact that a RKHS preserves distance and to the continuity properties of real spaces, a derivative with respect to an element f of a RKHS (a function) can be well defined and it satisfies the standard properties of derivatives of real functions. Following Kivinen et al. (2004), we can then derive a generalization of the Stochastic Gradient Descent algorithm for elements of $\mathcal{H}$. This method is referenced as *functional stochastic gradient descent.*

We compute the gradient of $E_n^\lambda(\boldsymbol{u})$ with respect to the functions $\boldsymbol{u}$ using the identity $u_{t,i}(\boldsymbol{z}_{0:t-1}) = \langle K(\boldsymbol{z}_{0:t-1}, \cdot), u_{t,i}\rangle_{\mathcal{H}}$, which is known as the *reproducing property* of kernels. Differentiating on both sides of this equation we obtain

$$\frac{\partial u_{t,i}(\boldsymbol{z}_{0:t-1})}{\partial u_{t,i}} = \frac{\partial \big\langle u_{t,i}, K_t(\boldsymbol{z}_{0:t-1}, \cdot)\big\rangle}{\partial u_{t,i}} = K_t(\boldsymbol{z}_{0:t-1}, \cdot),\ \ \forall\ i \in [r_t],\ \ t \in [T], \tag{5.8}$$

where $[K] = \{1, \dots, K\}$. The stochastic functional gradient can then be computed using the chain rule:

$$\nabla_{\boldsymbol{u}_t} c\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big) = \nabla_{\boldsymbol{u}_t(\boldsymbol{z}_{0:t-1})} c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\ K_t(\boldsymbol{z}^n_{0:t-1}, \cdot), \tag{5.9}$$

$$\implies \nabla_{\boldsymbol{u}_t} E_n^\lambda(\boldsymbol{u}) = \nabla_{\boldsymbol{u}_t(\boldsymbol{z}_{0:t-1})} c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\ K_t(\boldsymbol{z}^n_{0:t-1}, \cdot) + \lambda \boldsymbol{u}_t, \tag{5.10}$$

where $\nabla_{\boldsymbol{u}_t(\boldsymbol{z}_{0:t-1})} c\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big)$ corresponds to the derivative of $c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big)$ with respect to its scalar arguments $u_t^1(\boldsymbol{z}_{0:t-1}), \dots, u_t^{r_t}(\boldsymbol{z}_{0:t-1})$ evaluated at $\boldsymbol{z}^n$:

$$\nabla_{\boldsymbol{u}_t(\boldsymbol{z}_{0:t-1})} c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n) = \left[\frac{\partial c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)}{\partial u_t^1(\boldsymbol{z}_{0:t-1})}, \dots, \frac{\partial c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)}{\partial u_t^{r_t}(\boldsymbol{z}_{0:t-1})}\right].$$

Thus, the update rule for the standard functional stochastic gradient descent (FSGD) algorithm becomes

$$\begin{aligned}\boldsymbol{u}_t^{n+1} &= \boldsymbol{u}_t^n - \eta_n \nabla_{\boldsymbol{u}_t} E_n^{\lambda}(\boldsymbol{u}^n) \\ &= (1-\eta_n\lambda)\boldsymbol{u}_t^n - \eta_n \nabla_{\boldsymbol{u}_t(\boldsymbol{z}_{0:t-1})} c(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\; K_t(\boldsymbol{z}_{0:t-1}^n, \cdot),\end{aligned} \tag{5.11}$$

where η_n is the step-size of the algorithm and the sequence of controllers is initialized at some fixed function $\boldsymbol{u}_0 \in \mathcal{H}$.

Using the update rule in Eq. (5.11), we can easily show by induction on n that if the initial decision is of the form $\boldsymbol{u}_t^0(\cdot) = \boldsymbol{A}_t^0 \boldsymbol{K}_t(\boldsymbol{D}_t^0, \cdot)$ for some initial data matrix $\boldsymbol{D}^0$ and initial parameters $\boldsymbol{A}_t^0$, then the solutions $\boldsymbol{u}^n$ produced at every iteration also have this form. Specifically, for each $n > 0$ and for all $t \in [T]$, there exist a scalar matrix $\boldsymbol{A}_t^n$ and a data matrix $\boldsymbol{D}^n$ such that $\boldsymbol{u}_t^n(\cdot) = \boldsymbol{A}_t^n \cdot \boldsymbol{K}_t(\boldsymbol{D}_t^n, \cdot)$. In fact, this parametrization allows us to rewrite the functional update rule in Eq. (5.11) as a nonfunctional (scalar) update on the data matrix $\boldsymbol{D}^n$ and the parameters $\boldsymbol{A}_1^n, \ldots, \boldsymbol{A}_T^n$ as follows:

$$\boldsymbol{D}^{n+1} = [\boldsymbol{D}^n,\ \boldsymbol{z}^n], \quad \boldsymbol{A}^{n+1} = \left[(1-\eta_n\lambda)\boldsymbol{A}^n,\ \eta_n \nabla_{\boldsymbol{u}(\boldsymbol{z})} c\big(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n\big)\right],$$

where

$$\boldsymbol{A}^n := \begin{bmatrix} \boldsymbol{A}_1^n \\ \vdots \\ \boldsymbol{A}_T^n \end{bmatrix}, \quad \text{and} \quad \nabla_{\boldsymbol{u}(\boldsymbol{z})} c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) := \begin{bmatrix} \nabla_{\boldsymbol{u}_1(\boldsymbol{z}_0)} c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) \\ \vdots \\ \nabla_{\boldsymbol{u}_T(\boldsymbol{z}_{0:T-1})} c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) \end{bmatrix}.$$

Notice that this update forces the data matrix to have one more column after every iteration, which brings us back to the same problem we had when applying the Representer Theorem. However, because this is an iterative algorithm, we will reduce the dimension of the data matrix $\boldsymbol{D}^n$ after every iteration by measuring the contribution of each individual observation $\boldsymbol{z}^n$ and removing those observations that added almost no value to the decision.

5.4.2 Proximal Projection

We now describe how to reduce the number of observations in the data matrix $\boldsymbol{D}^n$ with the goal of reducing the dimension of the parameters $\boldsymbol{A}^n$. We observed that the Representer Theorem as well as the FSGD algorithm generate decisions $u_{t,i}$ that belong to the subspace of $\mathcal{H}_t$ spanned by the functions $K_t(\boldsymbol{z}^1_{0:t-1}, \cdot), \dots, K_t(\boldsymbol{z}^N_{0:t-1}, \cdot)$. What we want is to produce decisions that belong to a smaller subspace, one generated using fewer observations.

Suppose that $\tilde{\boldsymbol{D}}^{n+1}$, and $\tilde{\boldsymbol{A}}^{n+1}$ are the values resulting from the FSGD iterative rule in Eq. (5.11), i.e,

$$\tilde{\boldsymbol{D}}^{n+1} = [\boldsymbol{D}^n, \ \boldsymbol{z}^n] \quad \text{and} \quad \tilde{\boldsymbol{A}}^{n+1} = \left[(1-\eta_n\lambda)\boldsymbol{A}^n, \ \eta_n \nabla_{\boldsymbol{u}(\boldsymbol{z})} c(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\right],$$

which represent the decisions $\tilde{\boldsymbol{u}}^{n+1}_t(\cdot) = \tilde{\boldsymbol{A}}^{n+1}_t \boldsymbol{K}_t(\tilde{\boldsymbol{D}}^{n+1}_t, \cdot)$, and assume that we want to generate a decision that only uses observations from a smaller data matrix $\boldsymbol{D}^{n+1}$. We can approximate $\tilde{\boldsymbol{u}}^{n+1}$ with a decision $\boldsymbol{u}^{n+1}$ that only depends on observations in $\boldsymbol{D}^{n+1}$ by projecting each decision $\tilde{u}^{n+1}_{t,i}$ onto the subspace of $\mathcal{H}_t$ that is spanned by the functions $\boldsymbol{K}_t(\boldsymbol{D}^{n+1}_t, \cdot)$. If we denote this projection by $\Pi_{\boldsymbol{D}^{n+1}}(\cdot)$ then we can define

$$\boldsymbol{u}^{n+1} := \Pi_{\boldsymbol{D}^{n+1}}(\tilde{\boldsymbol{u}}^{n+1}) = \Pi_{\boldsymbol{D}^{n+1}}\big((1-\eta_n\lambda)\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} c(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\big). \tag{5.12}$$

The projection operator can be computed by solving the least squares problem

$$\boldsymbol{A}^{n+1} = \underset{\hat{\boldsymbol{A}}^{n+1}}{\arg\min} \sum_{t=1}^{T} \left\| \tilde{\boldsymbol{A}}^{n+1}_t \boldsymbol{K}_t(\tilde{\boldsymbol{D}}^{n+1}_t, \cdot) - \hat{\boldsymbol{A}}^{n+1}_t \boldsymbol{K}_t(\boldsymbol{D}^{n+1}_t, \cdot) \right\|^2_{\mathcal{H}^{r_t}_t}, \tag{5.13}$$

which has a closed form solution given by

$$\boldsymbol{A}^{n+1}_t = \left(\boldsymbol{K}_t[\boldsymbol{D}^{n+1}_t, \boldsymbol{D}^{n+1}_t]\right)^{-1} \boldsymbol{K}_t[\boldsymbol{D}^{n+1}_t, \tilde{\boldsymbol{D}}^{n+1}_t]\tilde{\boldsymbol{A}}^{n+1}_t, \quad \text{for all } t \in [T]. \tag{5.14}$$

We then have a simple way to project the FSGD solution onto the Hilbert subspace generated by a smaller data matrix $\boldsymbol{D}^{n+1}$, but we are still left the question: how do we find the right data matrix $\boldsymbol{D}^{n+1}$? As in Koppel et al. (2016), we use a method called destructive *kernel orthogonal matching pursuit* (KOMP) with pre-fitting, which was developed in Vincent and Bengio (2002). The KOMP algorithm takes as input a function $\tilde{\boldsymbol{u}} \in \mathcal{H}$ (represented by its data matrix $\tilde{\boldsymbol{D}}$ as well as the corresponding parameters $\tilde{\boldsymbol{A}}$), and a maximum error bound ϵ. For each element $\boldsymbol{d}$ in the data matrix $\tilde{\boldsymbol{D}}$, the algorithm computes the approximation function $\boldsymbol{u} = \Pi_{\tilde{\boldsymbol{D}}\setminus\{\boldsymbol{d}\}}(\tilde{\boldsymbol{u}})$ obtained by removing observation $\boldsymbol{d}$ from $\tilde{\boldsymbol{D}}$. Next, the algorithm removes the observation that produced the lowest error, updates the current function accordingly and then repeats this procedure to remove the next element. The algorithm stops removing elements when the difference between the current function and the best approximation function is larger than ϵ. The exact algorithm can be found in Algorithm 1.

Algorithm 1: Kernel Orthogonal Matching Pursuit (KOMP)

```
Input: Function $\tilde{\boldsymbol{u}}$ represented by data matrix $\tilde{\boldsymbol{D}}$ with $\tilde{M}$ columns, parameters $\tilde{\boldsymbol{A}}$,
       and $\epsilon > 0$.
Initialize $\boldsymbol{D} = \tilde{\boldsymbol{D}}$, $M = \tilde{M}$, $\boldsymbol{A} = \tilde{\boldsymbol{A}}$, and $\boldsymbol{u} = \tilde{\boldsymbol{u}}$ ;
while $\boldsymbol{D}$ is non-empty do
    for $j = 1, \ldots, \tilde{M}$ do
        Find minimal approximation error with data matrix element $\boldsymbol{d}^j$ removed:
            $\gamma_j^2 = \left\|\boldsymbol{u} - \Pi_{\boldsymbol{D}\setminus\{\boldsymbol{d}^j\}}(\boldsymbol{u})\right\|_{\mathcal{H}}^2 = \min_{\hat{\boldsymbol{A}}} \sum_{t=1}^{T} \left\|\boldsymbol{A}_t \cdot \boldsymbol{K}_t(\boldsymbol{D}_t, \cdot) - \hat{\boldsymbol{A}}_t \cdot \boldsymbol{K}_t(\boldsymbol{D}_t\setminus\{\boldsymbol{d}_t^j\}, \cdot)\right\|_{\mathcal{H}_t}^2.$
    end
    Find the index with minimum approximation error: $j^* = \arg\min \gamma_j$
    if $\gamma_{j^*} > \epsilon$ then
        stop;
    else
        Prune data matrix: $\boldsymbol{D} = \boldsymbol{D}\setminus\{\boldsymbol{d}^{j^*}\}$;
        Update $M = M - 1$;
        Update the parameters:
          $\boldsymbol{A} = \arg\min_{\hat{\boldsymbol{A}}} \sum_{t=1}^{T} \left\|\boldsymbol{A}_t \cdot \boldsymbol{K}_t(\boldsymbol{D}_t, \cdot) - \hat{\boldsymbol{A}}_t \cdot \boldsymbol{K}_t(\boldsymbol{D}_t\setminus\{\boldsymbol{d}_t^j\}, \cdot)\right\|_{\mathcal{H}_t}^2$.
    end
end
Output: $\boldsymbol{D}$, $\boldsymbol{A}$, $\boldsymbol{u}$.
```

5.4.3 The Algorithm

By combining Functional Stochastic Gradient Descent with the Kernel Orthogonal Matching Pursuit we are able to develop an algorithm that approximates the minimizer of $\mathbb{E}^{\lambda}(\boldsymbol{u})$ with decision rules that are represented using only a few parameters. The algorithm is initialized with a decision rule $\boldsymbol{u}_t^0 = \boldsymbol{A}^0 \boldsymbol{K}_t(\boldsymbol{D}^0, \cdot)$, which in practice is usually set to 0. Then, in each iteration, it performs one FSGD step and then applies the KOMP algorithm in order to obtain an approximated decision with fewer observations. Notice that if we define the projected gradient $\tilde{\nabla}$ by

$$\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda}(\boldsymbol{u}^n) := \frac{\boldsymbol{u}^n - \Pi_{\boldsymbol{D}^{n+1}}[\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} E_n^{\lambda}(\boldsymbol{u}^n)]}{\eta_n}, \tag{5.15}$$

then we can write the iterative updates of this procedure in the same form as the standard iterative updates of FSGD:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \eta_n \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda}(\boldsymbol{u}^n). \tag{5.16}$$

Since stochasticity does not guarantee a strict objective descent, the algorithm keeps track of the best decision rules observed and at the end it outputs the decision $\boldsymbol{u}_{\mathcal{S}}^*$ with the lowest empirical error $E_{\mathcal{S}}^{\lambda}$ with respect to the data set $\mathcal{S}$. The exact formulation can be found in Algorithm 2.

5.5 Convergence Analysis

In this section, we show that for a specific choice of step-size the objective value of the decision output by the algorithm converges to the objective value of the true minimizer. We first present the three main assumptions that we make on the problem settings in order to

Algorithm 2: Sparse Multistage Optimization via Kernels [SMOK]

```
Input: Data points $\mathcal{S} = \{\boldsymbol{z}^n\}_{n=1,\dots,N}$, error bounds $\epsilon_n$, learning rate $\eta_n$, and initial
       decision $\boldsymbol{u}^0$ represented with data matrix $\boldsymbol{D}^0$ and parameters $\boldsymbol{A}^0$.
for $n = 1, \dots, N$ do
    Take FSGD step using the $n^{th}$ sample $\boldsymbol{z}^n$ to obtain
        $\tilde{\boldsymbol{D}}^{n+1} = [\boldsymbol{D}^n, \boldsymbol{z}^n]$ and $\tilde{\boldsymbol{A}}^{n+1} = \left[(1-\eta_n\lambda)\boldsymbol{A}^n,\ \eta_n \nabla_{\boldsymbol{u}(\boldsymbol{z})} c(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\right]$.
    Reduce the data matrix and number of parameters using
        $\boldsymbol{D}^{n+1}, \boldsymbol{A}^{n+1}, \boldsymbol{u}^{n+1} = \text{KOMP}(\tilde{\boldsymbol{D}}^{n+1}, \tilde{\boldsymbol{A}}^{n+1}, \epsilon_n)$
end
Output: $\boldsymbol{u}^*_{\mathcal{S}} = \arg\min_{\boldsymbol{u} \in \{\boldsymbol{u}^1,\dots,\boldsymbol{u}^N\}} E^\lambda_{\mathcal{S}}(\boldsymbol{u})$.
```

guarantee convergence of the algorithm:

Assumption 5.5.1. *The data space $\mathcal{Z}$ is compact, the kernels K_t are universal, and there exists a constant κ such that*

$$K_t(\boldsymbol{z}_{1:t-1}, \boldsymbol{z}_{1:t-1}) \leq \kappa, \quad \forall\, \boldsymbol{z} \in \mathcal{Z}, \quad \forall\ t \in [T].$$

Assumption 5.5.2. *There exists a constant C such that for all $\boldsymbol{z} \in \mathcal{Z}$ the loss function satisfies*

$$\left| c(\boldsymbol{u}, \boldsymbol{z}) - c(\boldsymbol{u}', \boldsymbol{z}) \right| \leq C \|\boldsymbol{u} - \boldsymbol{u}'\|_2, \quad \forall\ \boldsymbol{u}, \boldsymbol{u}' \in \mathbb{R}^{r_1 + \dots + r_T}.$$

Assumption 5.5.3. *The loss function $c(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z})$ is convex and differentiable with respect to the scalar arguments $\boldsymbol{u}(\boldsymbol{z})$ for all $\boldsymbol{z} \in \mathcal{Z}$.*

Assumption 5.5.1 naturally holds for most data domains, and this is a necessary assumption to ensure that the Hilbert norm of the optimizer of E^λ is bounded. Assumption 5.5.2 holds whenever the cost function c as well as the constraint functions g_q are Lipschitz. This assumption implies that the gradient of c with respect to the scalars $\boldsymbol{u}(\boldsymbol{z})$ is bounded as

$$\|\nabla_{\boldsymbol{u}(\boldsymbol{z})} c(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z})\|_2 \leq C, \tag{5.17}$$

which in turn allows us to upper bound the expected norm of the gradient $\mathbb{E}\left[\|\nabla_{\boldsymbol{u}} E_n^\lambda(\boldsymbol{u})\|_{\mathcal{H}_t}^2\right]$. Assumption 5.5.3 is a standard condition for convergence of descent methods, and it can be relaxed to the case in which the loss function is almost everywhere differentiable by applying subgradients instead of gradients.

Theorem 5.5.4. *Let $\boldsymbol{u}_\mathcal{S}^* := \arg\min_{\boldsymbol{u}\in\{\boldsymbol{u}^1,\dots,\boldsymbol{u}^N\}} E_\mathcal{S}^\lambda(\boldsymbol{u})$ be the decisions generated by Algorithm 2 when given the set $\mathcal{S} = \{\boldsymbol{z}^n\}_{n=1}^N$ as input, and let $\mathbf{u}^\lambda$ be the true minimizer of $E^\lambda(\boldsymbol{u})$ over $\mathcal{H}$. If we use constant step-size η and constant error bounds $\epsilon = P_2\eta^2$ for some constant $P_2 > 0$, then under Assumptions 1-3, we have that*

$$\mathbb{E}\left[E^\lambda(\boldsymbol{u}_\mathcal{S}^*) - E^\lambda(\boldsymbol{u}^\lambda)\right] \le \mathcal{O}\left(\frac{\eta}{\lambda}\right).$$

Proof. See Appendix D.3. □

Corollary 5.5.5. *Let $\boldsymbol{u}^*$ be the true minimizer of $E(\cdot)$ over $\mathcal{F}$. If we use constant step-size with $\eta = \frac{P_1}{\sqrt{N}} < \frac{1}{\lambda}$, and $P_1 > 0$, constant error bounds $\epsilon = P_2\eta^2$ for some constant $P_2 > 0$, and regularization parameter λ such that $\lambda \xrightarrow[N\to\infty]{} 0$ and $\lambda\sqrt{N} \xrightarrow[N\to\infty]{} \infty$, then under Assumptions 1-3 we have that*

$$\lim_{N\to\infty} \mathbb{E}[|E(\boldsymbol{u}_\mathcal{S}^*) - E(\boldsymbol{u}^*)|] = 0. \tag{5.18}$$

Proof. See Appendix D.3. □

Since L_1 convergence implies convergence in probability, the corollary also implies that the expected loss achieved with Algorithm 2 converges in probability to the optimal solution.

In addition, from Theorem 5.5.4 we observe that setting $\eta = \frac{P_1}{\sqrt{N}}$ makes the objective value of the solution found by Algorithm 2 converge to the optimal solution of problem (5.3) with a rate of convergence of $\mathcal{O}\left(\frac{1}{\lambda\sqrt{N}}\right)$. Convergence can also be achieved under diminishing step size, although with a slower rate of $\mathcal{O}\left(\frac{1}{\lambda\log N}\right)$. In practice, a diminishing step size or a very small constant step size might make our data matrix $\boldsymbol{D}^n$ grow arbitrarily large, since little or no pruning would be done at each iteration. A constant step size is then what allows us to control the trade-off between accuracy and memory required; we want to use a step size η that is small enough to make the error in Theorem 5.5.4 small, but large enough for the pruning to be done.

5.6 Complexity Analysis

Let M_n be the size of the data matrix $\boldsymbol{D}^n$ during the n^{th} iteration of Algorithm 2. We analyze both space and time complexities per iteration in terms of M_n.

Space: At each iteration we need to store the kernel matrix $\boldsymbol{K}_t[\boldsymbol{D}_t^n, \boldsymbol{D}_t^n] \in \mathbb{R}^{M_n \times M_n}$ and its inverse as well as the parameters $\boldsymbol{A}_t^n \in \mathbb{R}^{r_t \times M_n}$ for each t. This results in $\mathcal{O}(TM_n^2 + M_n \sum_{t=1}^T r_t)$ memory requirement.

Time: For the FSGD step, computing the gradient takes $\mathcal{O}(M_n \sum_{t=1}^T r_t)$ time. Computing from scratch the kernel matrices $\boldsymbol{K}_t[\boldsymbol{D}_t^n, \boldsymbol{D}_t^n] \in \mathbb{R}^{M_n \times M_n}$ and its inverses (needed for the pruning step) takes $\mathcal{O}(M_n^2 \sum_{t=0}^T q_t)$ and $\mathcal{O}(TM_n^3)$ time respectively. However, by using a recursive rule to compute these matrices in terms of the corresponding values in the previous iteration, the times become $\mathcal{O}(M_n \sum_{t=0}^T q_t)$ and $\mathcal{O}(M_n^2)$ respectively. In addition, the matrix multiplication in Eq. (5.14) takes $O(M_n^2)$ time for each t. Since at most M_n elements can be removed from the dictionary at the n^{th} iteration, we obtain that in the worst case scenario the time per iteration becomes $\mathcal{O}(TM_n^3 + M_n^2 \sum_{t=0}^T q_t)$.

Let us now discuss the size of M_n. In the worst-case, we know that for all iterations the size of the data matrix is upper bounded by the covering number M of the data domain (Zhou 2002). More specifically, for fixed step size η and fixed error bound $\epsilon = P_2\eta^2$, we have that if the data space $\mathcal{Z}$ is compact (Assumption 5.5.1), then M_n is upper bounded by the minimum number of balls of radius $\frac{P_2\eta}{C}$ needed to cover the compact set $K_1(\mathcal{Z}_0, \cdot) \times \ldots \times K_T(\mathcal{Z}_{0:T-1}, \cdot)$

of kernel transformations (see for example the proof of Theorem 3 in Koppel et al. (2016)). While an exact expression for this cover number M is unknown, the number is finite (Anthony and Bartlett 2009) and it decreases as η or P_2 increases. In particular, the maximum number of samples in the data matrix depends on the step size η and the constant P_2, but not on the data size N.

Denoting the cover number described above by M and considering fixed values of T and of the dimensions $r_1, \dots, r_T$ and $q_0, \dots, q_T$, we obtain that the worst case total time across the N iterations of Algorithm 2 can be upper bounded by $\mathcal{O}(NM^3)$ and worst case total space required is $\mathcal{O}(NM^2)$. While the worst case scenario cannot happen for all iterations (for example, if M elements are pruned in one iteration, the next iteration is very fast), this bound is enough to conclude that total time and total space are in the worst case linear in the number of iterations. Notice that if we removed the pruning step, the entire algorithm would require $\Omega(N^2)$ space to store the kernel matrix and $\Omega(N^2)$ time for computations, showing that Algorithm 2 indeed reduces the overall complexity as the number of iterations becomes much larger than M.

5.7 Computational Experiments

We perform computational experiments for the inventory control and the shipment planning problems to analyze the average out-of-sample performance as well as the tractability of the proposed algorithms. For both applications we compare the SMOK algorithm proposed in Algorithm 2, to the MOK algorithm (Multistage Optimization with Kernels), which is the result of applying the FSGD algorithm without the pruning step. Moreover, we compare the SMOK and MOK algorithms against three other benchmarks:

1. **SRO:** Sample robust optimization approach from Bertsimas et al. (2022c), in which all samples are assigned equal weight $\frac{1}{N}$. We use uncertainty sets bounded by ϵ in the ℓ_1 norm as well as multi-policy approximation with linear decision rules.

2. **SRO-knn:** Sample robust optimization with covariates approach developed in Bertsimas and McCord (2019), using uncertainty sets bounded by ϵ in the ℓ_1 norm as well as multi-policy approximation with linear decision rules. The weights were obtained using the k_N-nearest neighbors approach.

3. **SAA-knn:** Sample average approximation method, which is equivalent to the SRO-knn approach with ($\epsilon = 0$).

We analyze the computational results for several instances of the inventory control problem. First, we consider a high dimensional instance of the problem to show the tractability of the SMOK algorithm as well as to compare its performance against other methods. Next, we analyze how the performance of the proposed algorithms varies with the dimensions of the problem (number of periods, data size, dimension of the data as well as dimension of the controllers). For instances in which the number of periods is less than 5 we are also able to compute lower bounds for the loss achieved by the optimal decision rules, which enables us to quantify the optimality gap of the proposed methods.

For the shipment planning application we reproduce the results from Bertsimas and McCord (2019) to compare the SMOK and MOK algorithms against sample robust optimization (with and without covariates) and sample average approximation. For training all these benchmarks we use the same parameter values reported in Bertsimas and McCord (2019).

Handling Constraints: Often the sequence of decisions $\boldsymbol{u}(\boldsymbol{z})$ must satisfy certain convex constraints for all possible disturbances, transforming the problem of interest into

$$\begin{aligned} \min_{\boldsymbol{u}\in\mathcal{F}} \quad & \mathbb{E}_{\boldsymbol{z}}\big[c\big(\boldsymbol{u}(\boldsymbol{z}),\boldsymbol{z}\big)\big] \\ \text{s.t.} \quad & g_q\big(\boldsymbol{u}(\boldsymbol{z})\big) \le 0, \quad \forall\, \boldsymbol{z}\in\mathcal{Z}, \quad \forall\, q\in[Q]. \end{aligned} \tag{5.19}$$

We address this problem by relaxing the constraints into the objective with a penalty function. More specifically, in Algorithm 2 we replace the cost $c(\boldsymbol{u}(\boldsymbol{z}),\boldsymbol{z})$ with a new loss function c^{ψ}

defined as

$$c^{\psi}\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) \coloneqq c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) + \psi \sum_{q=1}^{Q} \max\Big(0, g_q\big(\boldsymbol{u}(\boldsymbol{z})\big)\Big)^2, \tag{5.20}$$

where ψ is the penalty parameter. Although feasibility is not guaranteed, the constraint violation is expected to vanish for large enough ψ (see Lemma D.2.7). Convergence analysis for the SMOK algorithm applied to this constrained problem can be found in Appendix D.3.

Parameter Settings: We train the SMOK and MOK algorithms using Gaussian kernels and constant step size. The values for λ, ψ and θ were found using validation, and the decisions were projected onto the space of feasible decisions before making any evaluations, both at training and testing stages (this means that the decisions evaluated had 0 constraint violation). For each instance of the problem the constant step size η was initially set to 10^{-5} and it was repeatedly increased by factors of 5 so long as the average training loss did not worsen and the iterations were reaching convergence. The parameter P_2 for the error bound ϵ was initially set to 0.1 and was repeatedly increased by factors of 2; we stopped increasing it when the average training loss significantly worsened.

Software Utilized: Experiments were implemented in Python 3 (Van Rossum and Drake 2009b) using the NumPy library (Harris et al. 2020). We clarify that Eq. (5.14) can often be difficult to compute due to numerical instability in the calculations for the inverse matrix. To address this issue we add a small value $\lambda = 1e^{-7}$ to the diagonal of a matrix before computing its inverse. In terms of hardware, all experiments where run on an Intel(R) Core(TM) i7-8557U CPU @ 1.70GHz processor with 4 physical cores (hyper-threading enabled). The machine has a 32KB L1 cache and 256KB L2 cache per core, and an 8MB L3 cache. There is a total of 16GB DRAM.

5.7.1 Inventory Control Problem

We consider a multistage inventory control problem with linear constraints. At each stage t with initial inventory s_t, a retailer places procurement orders $\boldsymbol{u}_t \in \mathbb{R}^r$ at various suppliers, and later observes the demands $\boldsymbol{w}_t \in \mathbb{R}^q$. At the end of each stage, the firm incurs a per-unit holding cost of h_t and a back-order cost of b_t. The inventory is not backlogged, and therefore the initial inventory for the next period is given by the linear equation $s_t = s_{t-1} + \mathbf{1}^\top \boldsymbol{u}_t - \mathbf{1}^\top \boldsymbol{w}_t$, with zero initial inventory for the first period. In addition, the procurement orders are upper bounded by a constant L and the sum of procurement orders for two consecutive stages cannot exceed a constant ℓ. As in Ban et al. (2018), we consider the scenario in which retailers can observe auxiliary covariates $\boldsymbol{x}$ that relate to the future demands (e.g. in the fashion industry color and brand are useful factors for predicting demand of the products). For a problem with T periods, we can formulate this optimization problem as

$$\begin{array}{lll}
\min\limits_{\boldsymbol{u}_{1:T}} & \mathbb{E}_{\boldsymbol{w}|\boldsymbol{x}}\left[\sum\limits_{t=1}^{T} h_t\,[s_t]^+ + b_t\,[-s_t]^+ \,\middle|\, \boldsymbol{x} = \boldsymbol{x}_0\right] & \\
\text{s.t.} & s_t = s_{t-1} + \mathbf{1}^\top \boldsymbol{u}_t - \mathbf{1}^\top \boldsymbol{w}_t, & \forall t \in [T], \\
& \boldsymbol{u}_t \geq \mathbf{0}, & \forall t \in [T], \\
& \boldsymbol{u}_t \leq L\mathbf{1}, & \forall t \in [T], \\
& \boldsymbol{u}_t + \boldsymbol{u}_{t+1} \leq \ell\mathbf{1}, & \forall t \in [T-1].
\end{array}$$

The parameters h_t, b_t were chosen to be 2 and 1, respectively. The data sets used in these experiments were generated by sampling $\boldsymbol{x}$ from a Truncated Gaussian Distribution with mean 2 and standard deviation 0.5, and with truncating bounds 0 and 6. The demands $\boldsymbol{w}_t$ were then obtained as a linear function of the covariates with some added noise; specifically, $\boldsymbol{w}_t = \alpha_t \boldsymbol{x} + \boldsymbol{\epsilon}_t$, where ϵ_t was sampled from a standard distribution and the constants α_t were selected to be close to 50.

We first consider a large instance of the problem with $T = q = r = 10$, and we set the control bounds as $L = 150$ and $\ell = 200$. We use a training set with 2000 sample paths and we approximate the expected loss achieved by each method by averaging the losses across a common testing set with 10^4 sample paths. Since the SRO and SRO-knn methods become intractable for problems of this magnitude, in this experiment we only compare the SMOK and MOK methods to SAA-knn. We use validation to choose the best parameters for all methods and we evaluate the results on the testing set. In table 5.1 we observe that both SMOK and MOK outperform SAA-knn in terms of average out-of-sample loss and computational time. Moreover, the number of parameters needed for the SMOK algorithm is smaller by two orders of magnitude compared to the other methods. Even though we observe an increase in computation time for SMOK with respect to MOK (due to the overhead computation time for the pruning step), we also see that adding sparsity helped SMOK achieve a better average loss.

	Avg OOS Loss	**Total Time (hours)**	**No. of Params**
SMOK	491.30	0.3	1.5×10^3
MOK	493.74	0.1	2×10^5
SAA-knn	496.04	14.36	2.2×10^5

Table 5.1: Average out-of-sample (OOS) loss and total computation time for inventory problem with $T = q = r = 10$.

We next consider other instances of the inventory problem to analyze how the dimensions of the problem affect the overall performance of the SMOK and MOK algorithms. We compared these two methods to a third algorithm ADR (Affine Decision Rules), which refers to the common approximation technique of restricting the space of decision rules to be affine functions. We train all methods using the same training sets and the same validation sets (with size equal to 30% of the training size), and we approximate the expected loss achieved by averaging across a common testing set of 10^5 sample paths. In addition, we compute lower bounds for the optimal expected loss when $T \leq 5$ (see Appendix D.4), which allows us to analyze the optimality gap for the different methods.

Multiple data sets were generated to analyze the performance of the algorithms as we increase the number of periods, the training size, the dimension of the data and the dimension of the controls. In each case we analyze the average out-of-sample loss and the size M of the data matrix, which refers to number of parameters per control. We also analyze the computational time for each iteration of Stochastic Gradient Descent (projected or not projected), and the evaluation time (time it takes to evaluate the empirical loss function $E_{\mathcal{S}}^{\lambda}(\boldsymbol{u})$ given the parameters for the functional representation of $\boldsymbol{u}$). Notice that since the stochastic gradient descent algorithm does not strictly descend, the empirical loss of the validation set needs to be evaluated every certain number of iterations, which makes the evaluation time part of the total training time.

Varying the Number of Periods: $(L = 150, \ell = 200, q = r = 1, N = 2000, T = 2, 3, 4, 5)$

In Figure 5.1a, we observe that the convergence trajectory is not significantly affected by the pruning step, and the number of iterations needed until convergence does not change much for $T \geq 3$. In addition, we see in Figure 5.1b that ADR results in very poor performance, while both the SMOK and MOK algorithms are quite close to the lower bounds found for the optimal expected loss. In Figure 5.1c we observe that the time per iteration of stochastic gradient descent grows linearly for both SMOK and MOK, but MOK takes longer times due to the overhead introduced by the pruning step. The evaluation time (Figure 5.1d) also grows linearly for both algorithms, although unlike the time per iteration, the slope is larger for MOK than for SMOK because the number of parameters is significantly smaller for this last method (SMOK algorithm reduced the size M of the data matrix from 2000 to values below 15).

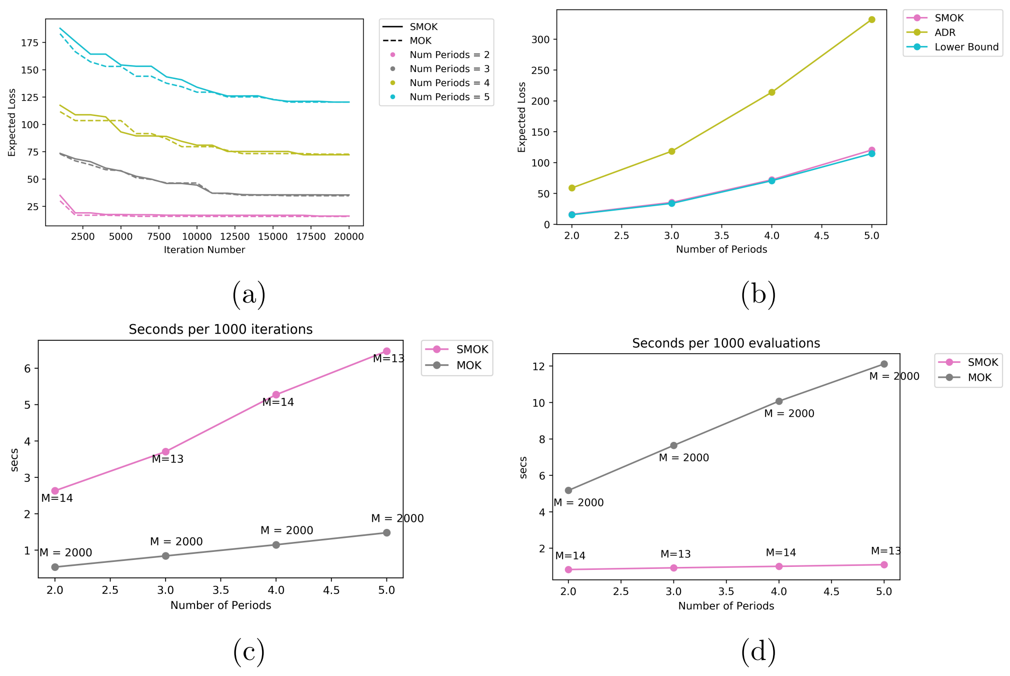
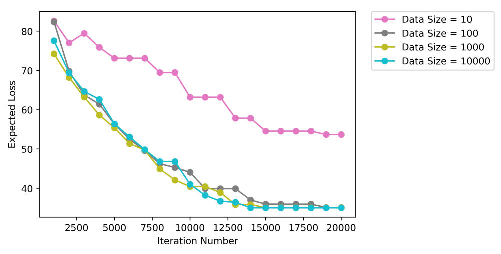


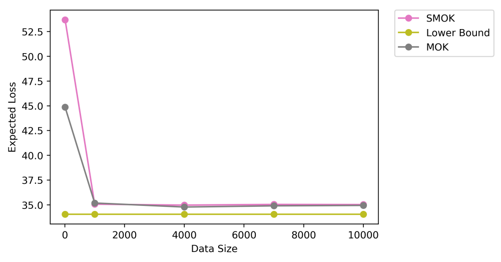
Figure 5.1: Expected loss and computational time for varying number of periods.

Varying the Data Size: $(L = 150, \ell = 200, q = r = 1, T = 3,$ $N = 10, 100, 1000, 4000, 7000, 10000)$

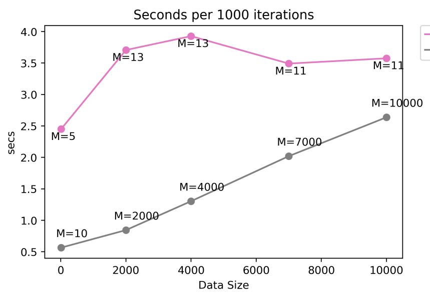
Figure 5.2b shows that, as anticipated, the expected loss achieved by both MOK and SMOK algorithms decreases as the size of the training set becomes larger. The number of iterations required to reach convergence (Figure 5.2a) does not change much with the data size and the expected loss achieved remains relatively constant after a large enough training size, which occurs around $N = 1000$. In Figures 5.2c,5.2d we can observe a significant memory improvement of SMOK over MOK when N becomes very large. For $N = 10^4$, for example, SMOK outputs decision rules with only 11 parameters, while SMOK requires 10^4 parameters per control. The evaluation time in Figure 5.2d grows quadratically with the number of parameters in each control (the quadratic factor comes from computing the kernel matrix $\boldsymbol{K}_t[\boldsymbol{D}_t, \boldsymbol{D}_t]$), which in the case of MOK corresponds to the size of the training set. Since the SMOK algorithm has much fewer parameters, it takes under half a second to evaluate the average loss of 1000 samples regardless of the training data size. Notice that the time per iteration (Figure 5.2c) is higher for SMOK than for MOK when N is small due to the pruning step. However, we observe that the time per iteration increases linearly for MOK while it stabilizes for SMOK, implying that for bigger values of N the SMOK method actually takes less time per iteration and per evaluation.



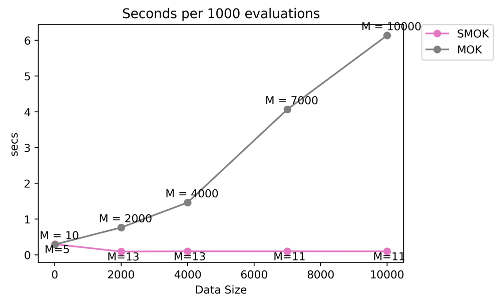
(a)



(b)



(c)

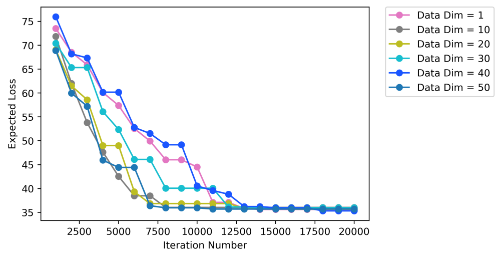


(d)

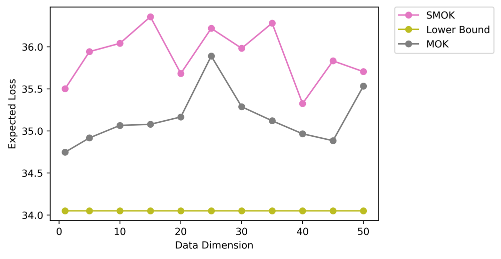
Figure 5.2: Expected loss and computational time for varying data sizes.

Varying Data Dimension: $(L = 150, \ell = 200, r = 1, T = 3, N = 2000, q = 1, 10, 20, 30, 40, 50)$

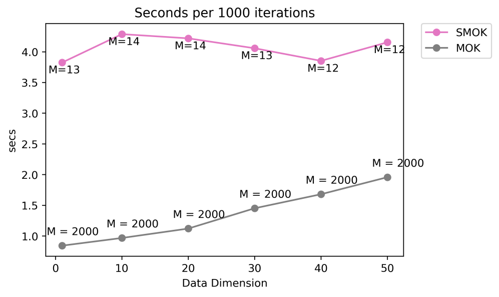
When generating data sets for this part we enforce that the value $\sum_q (\boldsymbol{w}_t)_q$ remains constant for all $t \in [T]$, which guarantees that the optimal expected loss is the same across instances. In Figure 5.3a, we observe that the trajectories of the expected loss across the FSGD iterations are quite similar for all the different dimensions of the data. More importantly, the error gap does not worsen as the dimension of the data increases (Figure 5.3b), showing that the accuracy of our algorithms does not worsen for data sets in large dimensional spaces. Additionally, in Figure 5.3d we observe that there is a slight linear increase in the evaluation time for both SMOK and MOK algorithms, which is expected since the dimension of the demand vector affects the computation of the exponent in the Gaussian kernel. In terms of the iteration time (Figure 5.3c), we can see that SMOK remains quite stable around 4 seconds per 1000 iterations, while MOK shows linear increase. As in the previous examples, the number of parameters of the SMOK algorithm is quite similar across the different experiments and remains under 15.



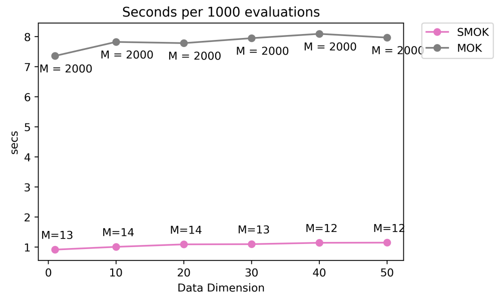
(a)



(b)



(c)



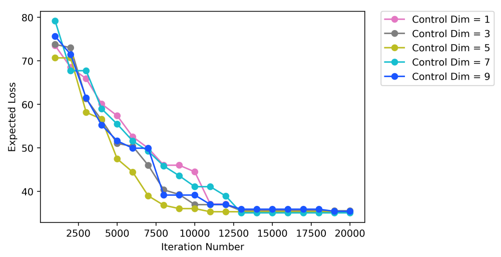
(d)

Figure 5.3: Expected loss and computational time for varying data dimensions.

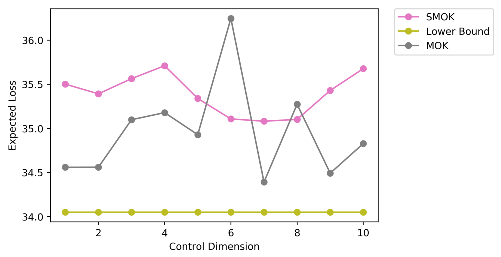
Varying Control Dimension: $(L = 150, \ell = \frac{200}{r}, q = 1, T = 3, N = 2000, r = 1, 3, 5, 10)$

In order to make a fair comparison, we set $L = \frac{150}{r}$ and $\ell = \frac{200}{r}$, which guarantees that the optimal expected loss is the same across instances. We observe in Figure 5.4b that the SMOK and MOK algorithms achieve very similar average out-of-sample loss across the different dimensions, and there are a couple of scenarios in which the pruning step helped to improve the expected loss. In addition, the number of iterations required for convergence (Figure 5.4a) does not seem to depend on the dimension of the control. Lastly, in Figure 5.4c we observe a slight linear increase in iteration time for both SMOK and MOK algorithms, with

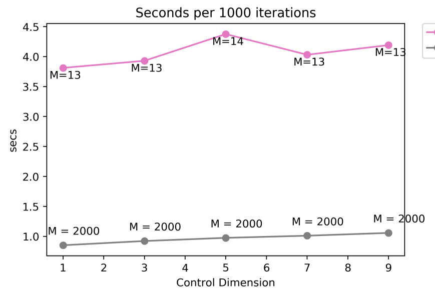
MOK having an advantage of around 4 seconds per 1000 iterations. In terms of evaluation time (Figure 5.4d) both algorithms grow linearly. As in the previous examples, the number of parameters for the SMOK algorithm is very low and varies between 13 and 14 across the different experiments.



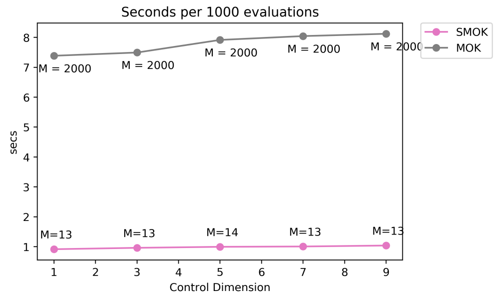
(a)



(b)



(c)



(d)

Figure 5.4: Expected loss and computational time for varying control dimensions.

5.7.2 Shipment Planning

We next analyze a two-stage shipment planning problem, following the same problem setting as in Bertsimas et al. (2022b) and Bertsimas and Kallus (2020). In this example, a decision maker has access to side information $\boldsymbol{x}$ (market trends, advertisements, etc.) and the goal is to ship items from the production facilities to multiple locations as to satisfy demand at minimum cost. First, the decision maker chooses an initial inventory quantity $u_{1f} \geq 0$ to be produced in each of the production facilities $f \in [F]$ at a per unit cost of p_1. Next, the demands $w_\ell \geq 0$ are observed in each location $\ell \in [L]$. If needed, the decision maker can produce additional units in each facility to satisfy demand, but at a higher per unit cost $p_2 > p_1$. Finally, demand is fulfilled by shipping $u_{2f\ell}$ units from facility f to location ℓ at per-unit cost $c_{f\ell}$, and each unit of satisfied demand generates revenue $a > 0$. The multistage optimization problem can then be written as

$$\begin{aligned}\min_{\boldsymbol{u}_1,\boldsymbol{u}_2} \mathbb{E}_{\boldsymbol{w}|\boldsymbol{x}} & \left[p_1 \sum_{f=1}^{F} u_{1f} - a \sum_{\ell\in[L]} w_\ell + p_2 \sum_{f=1}^{F} \left[\sum_{\ell=1}^{L} u_{2f\ell} - u_{1f} \right]^+ \sum_{f=1}^{F}\sum_{\ell=1}^{L} c_{f\ell}\, u_{2f\ell} \middle| \boldsymbol{x} = \boldsymbol{x}_0 \right] \\ \text{s.t.} \quad & \sum_{f=1}^{F} u_{2f\ell} \geq w_\ell, \quad \forall \ell \in [L], \forall \boldsymbol{w} \in \mathcal{W},\end{aligned}$$

where $\mathcal{W}$ is the set of all possible demand realizations. We reproduced the computational experiments performed in Bertsimas et al. (2022b) using the same parameters, the same data generation procedure as well as the same data set sizes. More specifically, we use $F = 4, L = 12, p_1 = 5, p_2 = 100$ and $a = 90$. The costs $\boldsymbol{c}$ and covariates $\boldsymbol{x}$ are also generated in an identical manner as in Bertsimas et al. (2022b).

We compare the SMOK and MOK algorithms against **SRO**, **SRO-knn** and **SAA-knn**. We train all methods over 100 independent training sets and evaluate them on a test set of size 100. The average out-of-sample profits achieved across the different methods are shown in Table 5.2. We observe that both MOK and SMOK outperform the other methods, with MOK achieving the highest revenues. However, as observed in Table 5.3, only the SAA and SMOK methods have tractable growth as the data size increases. In particular, the SMOK algorithm achieves high accuracies using only around 60 parameters per decision even when the data size increases to large numbers.

N	**SRO**	**SRO**	**SRO-knn**	**SRO-knn**	**SAA-knn**	**MOK**	**SMOK**
	$(\epsilon = 100)$	$(\epsilon = 500)$	$(\epsilon = 100)$	$(\epsilon = 500)$			
100	160007.0	159866.7	157522.9	158671.5	156639.6	**161536.9**	160737.0
200	160221.1	160075.0	157863.5	159136.9	156911.9	**164050.1**	163039.2
300	160431.0	160145.6	158697.6	159656.2	157669.6	**164860.6**	163703.8

Table 5.2: Out-of-sample profit for the shipment planning problem.

N	SRO	SRO	SRO-knn	SRO-knn	SAA-knn	MOK	SMOK
	($\epsilon = 100$)	($\epsilon = 500$)	($\epsilon = 100$)	($\epsilon = 500$)			
100	8	6	30	35	4	38	150
200	11	12	78	75	4	42	260
300	19	21	125	132	4	46	240
500	38	39	276	280	5	50	245
1000	74	76	772	790	10	65	255
5000	559	581	54000	54100	54	288	252

Table 5.3: Total computation time (seconds) for solving one instance of the shipment planning problem.

5.8 Conclusion

In this work, we developed a tractable data-driven approach for solving multistage stochastic optimization problems in which the uncertainties are independent of previous decisions. We represented the decision rules as elements of a reproducing kernel Hilbert space and performed functional stochastic gradient descent to minimize the empirical regularized loss. We next incorporated sparsification techniques based on function subspace projections, which decreased the number of parameters per controller. We prove that the proposed approach is asymptotically optimal for multistage stochastic programming with side information.

The practical value of the proposed data-driven approach was shown across various computational experiments on stochastic inventory management problems, demonstrating that it produces high-quality decisions, does not worsen in multidimensional settings and remains tractable even with large data sizes. This approach does not rely on the traditional use of approximation with scenario trees, and provides a novel method for leveraging advances in machine learning to solve multistage stochastic optimization problems.

Appendix A

Chapter 2 Appendix

A.1 Proofs of Lemmas

In this Appendix, we prove Lemmas 2.3.3, 2.5.1, 2.5.2 and 2.5.3.

A.1.1 Proof of Lemma 2.3.3

Proof: By Assumption 2.3.1 with $c = \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})$ we know

$$\min_{\theta} \max_{\boldsymbol{\delta}\in\mathcal{U}} \mathcal{L}(y, \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})) = \min_{\theta} \max_{\boldsymbol{\delta}\in\mathcal{U}} \mathcal{L}\left(y, \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})\boldsymbol{e}\right).$$

Define $\bar{\boldsymbol{z}}(\boldsymbol{\delta}) \coloneqq \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})\boldsymbol{e}$ and $\bar{\boldsymbol{z}}' = (\max_{\boldsymbol{\delta}\in\mathcal{U}} \bar{\boldsymbol{z}}_1(\boldsymbol{\delta}), \ \dots, \max_{\boldsymbol{\delta}\in\mathcal{U}} \bar{\boldsymbol{z}}_K(\boldsymbol{\delta}))$. Notice that the y^{th} coordinates of $\bar{\boldsymbol{z}}(\boldsymbol{\delta})$ and $\bar{\boldsymbol{z}}'$ are both zero, and therefore for all $k \in [K]$ we have

$$\bar{\boldsymbol{z}}_k(\boldsymbol{\delta}) - \bar{\boldsymbol{z}}_y(\boldsymbol{\delta}) = \bar{\boldsymbol{z}}_k(\boldsymbol{\delta}) \leq \max_{\boldsymbol{\delta}\in\mathcal{U}} \bar{\boldsymbol{z}}_k(\boldsymbol{\delta}) = \bar{\boldsymbol{z}}'_k = \bar{\boldsymbol{z}}'_k - \bar{\boldsymbol{z}}'_y.$$

Therefore, we can apply Assumption 2.3.2 with $\boldsymbol{z} = \bar{z}(\boldsymbol{\delta})$ and $\boldsymbol{z}' = \bar{z}'$ to obtain

$$\begin{aligned}
&\min_{\theta} \max_{\boldsymbol{\delta}\in\mathcal{U}} \mathcal{L}\left(y, \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})\boldsymbol{e}\right) \\
\le &\min_{\theta} \mathcal{L}\left(y, \left(\max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_1^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}), \dots, \max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_K^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})\right)\right).
\end{aligned}$$

We then conclude

$$\begin{aligned}
&\min_{\theta} \max_{\boldsymbol{\delta}\in\mathcal{U}} \mathcal{L}(y, \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})) \\
\le &\min_{\theta} \mathcal{L}\left(y, \left(\max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_1^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}), \dots, \max_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}_K^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - \boldsymbol{z}_y^L(\theta, \boldsymbol{x}+\boldsymbol{\delta})\right)\right).
\end{aligned}$$

A.1.2 Proof of Lemma 2.5.1

Proof. Since f is convex and closed, we have $f = (f^*)^*$ (Rockafellar 1970), and applying the definition of the convex conjugate function we obtain

$$f(\boldsymbol{z}(\boldsymbol{\delta})) = (f^\star)^\star(\boldsymbol{z}(\boldsymbol{\delta})) = \sup_{\boldsymbol{u}\in\mathrm{dom}(f^*)} \boldsymbol{z}(\boldsymbol{\delta})^\top \boldsymbol{u} - f^\star(\boldsymbol{u}),$$

which implies

$$\begin{aligned}
\sup_{\boldsymbol{\delta}\in\mathcal{U}} f(\boldsymbol{z}(\boldsymbol{\delta})) + g(\boldsymbol{z}(\boldsymbol{\delta})) &= \sup_{\boldsymbol{\delta}\in\mathcal{U}} \sup_{\boldsymbol{u}\in\mathrm{dom}(f^\star)} \boldsymbol{z}(\boldsymbol{\delta})^T \boldsymbol{u} - f^\star(\boldsymbol{u}) + g(\boldsymbol{z}(\boldsymbol{\delta})) \\
&= \sup_{\boldsymbol{u}\in\mathrm{dom}(f^\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}(\boldsymbol{\delta})^T \boldsymbol{u} - f^\star(\boldsymbol{u}) + g(\boldsymbol{z}(\boldsymbol{\delta})),
\end{aligned}$$

as desired. □

A.1.3 Proof of Lemma 2.5.2

Let $\mathcal{Z} = \{\boldsymbol{z}(\boldsymbol{\delta}) : \boldsymbol{\delta} \in \mathcal{U}\}$. Defining the indicator function

$$\gamma(\boldsymbol{z}|\mathcal{Z}) = \begin{cases} 0 & \text{if } \boldsymbol{z} \in \mathcal{Z}, \\ \infty & \text{otherwise,} \end{cases}$$

and applying the Fenchel duality theorem (Rockafellar 1970), we obtain:

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} g(\boldsymbol{z}(\boldsymbol{\delta})) = \sup_{\boldsymbol{z}\in\mathcal{Z}} g(\boldsymbol{z}) = \sup_{\boldsymbol{z}\in\operatorname{dom}(g)\cap\operatorname{dom}(\gamma)} g(\boldsymbol{z}) - \gamma(\boldsymbol{z}|\mathcal{Z}) = \inf_{\boldsymbol{v}\in\operatorname{dom}(g_\star)} \gamma^\star(\boldsymbol{v}|\mathcal{Z}) - g_\star(\boldsymbol{v}). \quad \text{(A.1)}$$

Finally, since $\gamma^\star(\boldsymbol{v}|\mathcal{Z}) = \sup_{\boldsymbol{z}\in\mathcal{Z}} \boldsymbol{z}^\top\boldsymbol{v}$, we conclude

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} g(\boldsymbol{z}(\boldsymbol{\delta})) = \inf_{\boldsymbol{v}\in\operatorname{dom}(g_\star)} \sup_{\boldsymbol{z}\in\mathcal{Z}} \boldsymbol{z}^\top\boldsymbol{v} - g_\star(\boldsymbol{v}) = \inf_{\boldsymbol{v}\in\operatorname{dom}(g_\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{z}(\boldsymbol{\delta})^\top\boldsymbol{v} - g_\star(\boldsymbol{v}).$$

A.1.4 Proof of Lemma 2.5.3

We first prove part a). By definition, we have

$$f^\star(\boldsymbol{z}) = \sup_{\boldsymbol{x}} \boldsymbol{z}^\top\boldsymbol{x} - \boldsymbol{p}^\top[\boldsymbol{x}]^+.$$

Notice that if the i^{th} component of $\boldsymbol{z}$ is negative for any i, then $f^\star(\boldsymbol{z}) = \infty$ because $\boldsymbol{x}$ can be the vector with an arbitrarily large negative value in the i^{th} coordinate and 0 everywhere else. Similarly, if the i^{th} component of $\boldsymbol{z}$ is larger than the i^{th} coordinate of $\boldsymbol{p}$ for any i, then again $f^\star(\boldsymbol{z}) = \infty$ because $\boldsymbol{x}$ can be the vector with an arbitrarily large positive value in the i^{th} coordinate and 0 everywhere else. Moreover, if $\boldsymbol{0} \le \boldsymbol{z} \le \boldsymbol{p}$, then

$$\sup_{\boldsymbol{x}} \boldsymbol{z}^\top\boldsymbol{x} - \boldsymbol{p}^\top[\boldsymbol{x}]^+ \le \sup_{\boldsymbol{x}} \boldsymbol{z}^\top\boldsymbol{x} - \boldsymbol{z}^\top[\boldsymbol{x}]^+ = \sup_{\boldsymbol{x}} \boldsymbol{z}^\top(\boldsymbol{x} - [\boldsymbol{x}]^+) \le 0.$$

Since $\boldsymbol{x} = \boldsymbol{0}$ achieves an objective value of 0, we conclude that $\boldsymbol{0} \leq \boldsymbol{z} \leq \boldsymbol{p}$ implies $f^{\star}(\boldsymbol{z}) = 0$ as desired.

Next, we proceed to prove part b). By definition of the concave conjugate we have

$$g_{\star}(\boldsymbol{z}) = \inf_{\boldsymbol{x}} \boldsymbol{z}^{\top}\boldsymbol{x} - (\boldsymbol{x}^{\top}\boldsymbol{u} - \boldsymbol{q}^{\top}[\boldsymbol{x}]^{+}) = \inf_{\boldsymbol{x}} (\boldsymbol{z} - \boldsymbol{u})^{\top}\boldsymbol{x} + \boldsymbol{q}^{\top}[\boldsymbol{x}]^{+}.$$

If the i^{th} component of $\boldsymbol{z}$ is larger than the i^{th} component of $\boldsymbol{u}$ for any i, then $g_{\star}(\boldsymbol{z}) = \infty$ because $\boldsymbol{x}$ can be the vector with an arbitrarily large negative value in the i^{th} coordinate and 0 everywhere else. Similarly, if the i^{th} component of $\boldsymbol{z}$ is smaller than the i^{th} coordinate of $(\boldsymbol{u} - \boldsymbol{q})$ for any i, then again $g_{\star}(\boldsymbol{z}) = \infty$ because $\boldsymbol{x}$ can be the vector with an arbitrarily large positive value in the i^{th} coordinate and 0 everywhere else. In addition, if $\boldsymbol{u} - \boldsymbol{q} \leq \boldsymbol{z} \leq \boldsymbol{u}$, then

$$\inf_{\boldsymbol{x}} (\boldsymbol{z} - \boldsymbol{u})^{\top}\boldsymbol{x} + \boldsymbol{q}^{\top}[\boldsymbol{x}]^{+} \geq \inf_{\boldsymbol{x}} (\boldsymbol{z} - \boldsymbol{u})^{\top}\boldsymbol{x} + (\boldsymbol{u} - \boldsymbol{z})^{\top}[\boldsymbol{x}]^{+} = \inf_{\boldsymbol{x}} (\boldsymbol{u} - \boldsymbol{z})^{\top}([\boldsymbol{x}]^{+} - \boldsymbol{x}) \geq 0.$$

Since $\boldsymbol{x} = \boldsymbol{0}$ achieves an objective value of 0, we conclude that $\boldsymbol{u} - \boldsymbol{q} \leq \boldsymbol{z} \leq \boldsymbol{u}$ implies $g_{\star}(\boldsymbol{z}) = 0$ as desired.

A.2 Generalized Results

We now state and proof the generalization of Theorem 2.5.4, Corollary 2.5.5 and Theorem 2.5.6 for the case in which the neural network has more than 2 layers.

Theorem A.2.1 (Generalization of Theorem 2.5.4)**.** *For all $2 \leq l \leq L$, it holds*

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^\ell(\theta, \boldsymbol{x}+\boldsymbol{\delta}) = \tag{A.2}$$

$$\sup_{\boldsymbol{s}_L} \inf_{\boldsymbol{t}_L} \ldots \sup_{\boldsymbol{s}_l} \inf_{\boldsymbol{t}_l} \sup_{\boldsymbol{\delta}\in\mathcal{U}} (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{z}^{l-1}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) + \sum_{\ell=l}^{L-1} (\boldsymbol{p}_{\ell+1} - \boldsymbol{q}_{\ell+1})^\top \boldsymbol{b}^\ell + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^L$$

$$\begin{aligned} s.t.\quad & \boldsymbol{p}_L = [(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s}_L \\ & \boldsymbol{q}_L = [-(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}_L \\ & \boldsymbol{p}_\ell = (([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{s}_\ell \quad \forall\, \ell = l, \ldots, L-1 \\ & \boldsymbol{q}_\ell = (([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{t}_\ell \quad \forall\, \ell = l, \ldots, L-1 \\ & 0 \leq \boldsymbol{s}_\ell, \boldsymbol{t}_\ell \leq 1 \quad \forall\, \ell = l, \ldots, L. \end{aligned} \tag{A.3}$$

Proof. We will proceed by backward induction on the layer number l.

Case $l = L$:

The proof is equivalent to the case $L = 2$ already proved in Section 2.5.

Case $l-1$:

Suppose the theorem holds for some fixed l with $l > 2$. We have

$$\begin{aligned} & (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{z}^{l-1}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) \\ =& (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top (\boldsymbol{W}^{l-1}[\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})]^+ + \boldsymbol{b}^{l-1}) \\ =& f_+(\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})) - f_-(\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})) + (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{b}^{l-1}, \end{aligned}$$

where

$$f_+(\boldsymbol{x}) = (\boldsymbol{p}_l^\top [\boldsymbol{W}^{l-1}]^+ + \boldsymbol{q}_l^\top [-\boldsymbol{W}^{l-1}]^+)[\boldsymbol{x}]^+, \quad \text{and}$$
$$f_-(\boldsymbol{x}) = (\boldsymbol{p}_l^\top [-\boldsymbol{W}^{l-1}]^+ + \boldsymbol{q}_l^\top [\boldsymbol{W}^{l-1}]^+)[\boldsymbol{x}]^+.$$

By Lemma 2.5.1 we then obtain

$$\begin{aligned}
&\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{z}^{l-1}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) \\
=&\sup_{\boldsymbol{\delta}\in\mathcal{U}} \; f_+(\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})) - f_-(\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})) + (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{b}^{l-1} \\
=& \sup_{\boldsymbol{u}_{l-1}\in\mathrm{dom}(f_+^\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{u}_{l-1}^\top \boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) - f_-(\boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta})) + (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{b}^{l-1}.
\end{aligned}$$

Defining the concave function $g(\boldsymbol{x}) = \boldsymbol{u}_{l-1}^\top \boldsymbol{x} - f_-(\boldsymbol{x})$, and applying Lemma 2.5.2 we obtain

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{z}^{l-1}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) \tag{A.4}$$
$$= \sup_{\boldsymbol{u}_{l-1}\in\mathrm{dom}(f_+^\star)} \inf_{\boldsymbol{v}_{l-1}\in\mathrm{dom}(g_\star)} \sup_{\boldsymbol{\delta}\in\mathcal{U}} \boldsymbol{v}_{l-1}^\top \boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) + (\boldsymbol{p}_l - \boldsymbol{q}_l)^\top \boldsymbol{b}^{l-1}. \tag{A.5}$$

Lastly, by Lemma 2.5.3 we can substitute

$$\begin{aligned}
\boldsymbol{u}_{l-1} &= ([(\boldsymbol{W}^{l-1}]^+)^\top \boldsymbol{p}_l + ([-\boldsymbol{W}^{l-1}]^+)^\top \boldsymbol{q}_l) \odot \boldsymbol{s}_{l-1} \\
&= \boldsymbol{p}_{l-1}, \quad \text{and} \\
\boldsymbol{v}_{l-1} &= (([\boldsymbol{W}^{l-1}]^+)^\top \boldsymbol{p}_l + ([-\boldsymbol{W}^{l-1}]^+)^\top \boldsymbol{q}_l) \odot \boldsymbol{s}_{l-1} - (\boldsymbol{p}_l[-\boldsymbol{W}^{l-1}]^+ + \boldsymbol{q}_l[\boldsymbol{W}^{l-1}]^+) \odot \boldsymbol{t}_{l-1} \\
&= \boldsymbol{p}_{l-1} - \boldsymbol{q}_{l-1},
\end{aligned}$$

which together with the induction hypothesis imply that Eq. (A.3) is equivalent to

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^{l-1}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) =$$

$$\begin{aligned}
\sup_{\boldsymbol{s}_L} \inf_{\boldsymbol{t}_L} \dots \sup_{\boldsymbol{s}_{l-1}} \inf_{\boldsymbol{t}_{l-1}} \sup_{\boldsymbol{\delta}\in\mathcal{U}} (\boldsymbol{p}_{l-1} - \boldsymbol{q}_{l-1})^\top \boldsymbol{z}^{l-2}(\theta, \boldsymbol{x}+\boldsymbol{\delta}) + \sum_{\ell=l-1}^{L-1} (\boldsymbol{p}_{\ell+1} - \boldsymbol{q}_{\ell+1})^\top \boldsymbol{b}^\ell + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^L \\
\text{s.t.} \quad & \boldsymbol{p}_L = [(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s}_L \\
& \boldsymbol{q}_L = [-(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}_L \\
& \boldsymbol{p}_\ell = (([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{s}_\ell \quad \forall\, l-1 \le \ell \le L-1 \\
& \boldsymbol{q}_\ell = (([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{t}_\ell \quad \forall\, l-1 \le \ell \le L-1 \\
& 0 \le \boldsymbol{s}_\ell, \boldsymbol{t}_\ell \le 1 \quad \forall\, \ell = l-1, \dots, L,
\end{aligned}$$

and therefore the theorem holds for $l-1$ as desired. □

Corollary A.2.2 (Generalization of Corollary 2.5.5)**.**

If $\mathcal{U} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \le \rho\}$*, then:*

$$\sup_{\boldsymbol{\delta}\in\mathcal{U}} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^L(\theta, \boldsymbol{x}+\boldsymbol{\delta}) \tag{A.6}$$

$$\begin{aligned}
= \sup_{\boldsymbol{s}_L} \inf_{\boldsymbol{t}_L} \dots \sup_{\boldsymbol{s}_2} \inf_{\boldsymbol{t}_2} \rho \|(\boldsymbol{p}_2 - \boldsymbol{q}_2)^\top \boldsymbol{W}^1\|_q + (\boldsymbol{p}_2 - \boldsymbol{q}_2)^\top \boldsymbol{W}^1 \boldsymbol{x} + \sum_{\ell=1}^{L-1} (\boldsymbol{p}_{\ell+1} - \boldsymbol{q}_{\ell+1})^\top \boldsymbol{b}^\ell + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^L \\
s.t. \quad & \boldsymbol{p}_L = [(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s}_L \\
& \boldsymbol{q}_L = [-(\boldsymbol{W}^L)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}_L \\
& \boldsymbol{p}_\ell = (([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{s}_\ell \quad \forall\, \ell = 2, \dots, L-1 \\
& \boldsymbol{q}_\ell = (([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1} + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}) \odot \boldsymbol{t}_\ell \quad \forall\, \ell = 2, \dots, L-1 \\
& 0 \le \boldsymbol{s}_\ell, \boldsymbol{t}_\ell \le 1 \quad \forall\, \ell = 2, \dots, L,
\end{aligned} \tag{A.7}$$

where $\|\cdot\|_q$ *is the conjugate norm of* $\|\cdot\|_p$*.*

Proof. The proof follows directly after applying Theorem A.2.1 with $l = 2$ and using again

Eq. (2.26). □

Definition A.2.2.1. *We introduce the following definitions to simplify notation:*

$$\begin{aligned}
\boldsymbol{s} &\coloneqq (\boldsymbol{s}_2, \dots, \boldsymbol{s}_L) \\
\boldsymbol{t} &\coloneqq (\boldsymbol{t}_2, \dots, \boldsymbol{t}_L) \\
\boldsymbol{p}_L(\boldsymbol{s}, \boldsymbol{t}) &\coloneqq [(\boldsymbol{W}^2)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{s}_L \\
\boldsymbol{q}_L(\boldsymbol{s}, \boldsymbol{t}) &\coloneqq [-(\boldsymbol{W}^2)^\top (\Delta \boldsymbol{e}_k^y)]^+ \odot \boldsymbol{t}_L
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{p}_\ell(\boldsymbol{s}, \boldsymbol{t}) &\coloneqq \big(([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\big) \odot \boldsymbol{s}_\ell \quad \forall\, 1 \le \ell < L \\
\boldsymbol{q}_\ell(\boldsymbol{s}, \boldsymbol{t}) &\coloneqq \big(([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\big) \odot \boldsymbol{t}_\ell \quad \forall\, 1 \le \ell < L
\end{aligned}$$

$$R_\ell(\boldsymbol{s}, \boldsymbol{t}) \coloneqq \sum_{\ell'=\ell}^{L-1} \left(\boldsymbol{p}_{\ell'+1}(\boldsymbol{s}, \boldsymbol{t}) - \boldsymbol{q}_{\ell'+1}(\boldsymbol{s}, \boldsymbol{t})\right)^\top \boldsymbol{b}^{\ell'}.$$

Theorem A.2.3 (Generalization of Theorem 2.5.6)**.**

$$\begin{aligned}
&\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_1 \le \rho} (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{z}^L(\theta, \boldsymbol{x} + \boldsymbol{\delta}) \\
&\qquad \le \inf_{0 \le \boldsymbol{t} \le 1} \max_{m \in [M]} \max \left\{ g_{k,m}^L(\theta, \boldsymbol{x}, \boldsymbol{t}, \rho), g_{k,m}^L(\theta, \boldsymbol{x}, \boldsymbol{t}, -\rho) \right\},
\end{aligned} \tag{A.8}$$

where the new network g is defined by the equations

$$\begin{aligned}
g_m^1(\boldsymbol{W}, \boldsymbol{x}, \boldsymbol{t}, a, r) &= r(a\boldsymbol{W}_m^1 + \boldsymbol{W}^1 \boldsymbol{x} + \boldsymbol{b}^1) \\
g_m^\ell(\theta, \boldsymbol{x}, \boldsymbol{t}, a, r) &= [r\boldsymbol{W}^\ell]^+ [g_m^{\ell-1}(\boldsymbol{W}, \boldsymbol{x}, \boldsymbol{t}, a, 1)]^+ + [-r\boldsymbol{W}^\ell]^+ [g_m^{\ell-1}(\boldsymbol{W}, \boldsymbol{x}, \boldsymbol{t}, a, -1)] \odot \boldsymbol{t}_\ell + r\boldsymbol{b}^\ell \\
g_{k,m}^L(\theta, \boldsymbol{x}, \boldsymbol{t}, a) &= [(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ [g_m^{L-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1)]^+ + \\
&\qquad [-(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ [g_m^{L-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1)] \odot \boldsymbol{t}_L + (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^L,
\end{aligned}$$

for all $1 < \ell < L$, $1 \leq k \leq K$, $a \in \{\rho, -\rho\}$, *and* $r \in \{-1, 1\}$.

The proof of this theorem relies on the following lemma.

Lemma A.2.4. *For all* $2 \leq \ell \leq L-1$ *it holds*

$$\sup_{0 \leq \boldsymbol{s}_\ell \leq 1} \boldsymbol{p}_\ell(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) + \boldsymbol{q}_\ell(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1) \tag{A.9}$$

$$= \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) + \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1) - (\boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) - \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}))^\top \boldsymbol{b}^\ell. \tag{A.10}$$

Proof. Let $2 \leq \ell \leq L-1$, we have

$$\sup_{0 \leq \boldsymbol{s}_\ell \leq 1} \boldsymbol{p}_\ell(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) + \boldsymbol{q}_\ell(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1)$$

$$= \sup_{0 \leq \boldsymbol{s}_\ell \leq 1} \left((([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})) \odot \boldsymbol{s}_\ell\right)^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) +$$
$$\left((([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})) \odot \boldsymbol{t}_\ell\right)^\top g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1)$$

$$= \sup_{0 \leq \boldsymbol{s}_\ell \leq 1} \left(([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\right)^\top \left(g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) \odot \boldsymbol{s}_\ell\right) +$$
$$\left(([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\right)^\top \left(g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1) \odot \boldsymbol{t}_\ell\right)$$

$$= \left(([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\right)^\top [g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1)]^+ +$$
$$\left(([-\boldsymbol{W}^\ell]^+)^\top \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) + ([\boldsymbol{W}^\ell]^+)^\top \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})\right)^\top \left(g_m^{\ell-1}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1) \odot \boldsymbol{t}_\ell\right)$$

$$= \boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, 1) + \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t})^\top g_m^{\ell}(\theta, \boldsymbol{x}, \boldsymbol{t}, a, -1) - (\boldsymbol{p}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}) - \boldsymbol{q}_{\ell+1}(\boldsymbol{s}, \boldsymbol{t}))^\top \boldsymbol{b}^\ell,$$

as desired. □

Proof of theorem A.2.3.

By Corollary A.2.2 with $p = 1$ we know

$$\sup_{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_1 \le \rho} (\Delta \boldsymbol{e}_k^y)^\top (\boldsymbol{z}^L(\theta, \boldsymbol{x} + \boldsymbol{\delta}) - \boldsymbol{b}^L) \tag{A.11}$$

$$\le \sup_{\boldsymbol{s}_L} \inf_{\boldsymbol{t}_L}, \dots \sup_{\boldsymbol{s}_2} \inf_{\boldsymbol{t}_2} \rho \|(\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1 \boldsymbol{x} + R_1(\boldsymbol{s},\boldsymbol{t}) \tag{A.12}$$

$$\le \inf_{\boldsymbol{t}_L \dots \boldsymbol{t}_2} \sup_{\boldsymbol{s}_L \dots \boldsymbol{s}_2} \rho \|(\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1 \boldsymbol{x} + R_1(\boldsymbol{s},\boldsymbol{t}), \tag{A.13}$$

where the last inequality follows from the min-max inequality. Observe that $\boldsymbol{p}_{\ell'}(\boldsymbol{s},\boldsymbol{t})$ and $\boldsymbol{q}_{\ell'}(\boldsymbol{s},\boldsymbol{t})$ are independent on $\boldsymbol{s}_\ell$ for all $\ell' > \ell$, which in turn implies that $R_{\ell'}(\boldsymbol{s},\boldsymbol{t})$ does not depend on $\boldsymbol{s}_\ell$ for all $\ell' > \ell$. We can then solve the optimization problem in Eq. (A.7) for fixed $\boldsymbol{t}$ as follows:

$$\sup_{0 \le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2 \le 1} \rho \|(\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1 \boldsymbol{x} + R_1(\boldsymbol{s},\boldsymbol{t})$$

$$= \max_{m \in [M]} \max \Bigg\{ \sup_{0 \le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2 \le 1} (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top (\boldsymbol{W}^1(\boldsymbol{x} + \rho \boldsymbol{e}_m) + \boldsymbol{b}^1) + R_2(\boldsymbol{s},\boldsymbol{t}),$$
$$\sup_{0 \le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2 \le 1} (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t}) - \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top (\boldsymbol{W}^1(\boldsymbol{x} - \rho \boldsymbol{e}_m) + \boldsymbol{b}^1) + R_2(\boldsymbol{s},\boldsymbol{t}) \Bigg\}$$

$$= \max_{m \in [M]} \max \Bigg\{ \sup_{0 \le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2 \le 1} \boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t})^\top g_m^1(\theta, \boldsymbol{x}, \boldsymbol{t}, \rho, 1) + \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t})^\top g_m^1(\theta, \boldsymbol{x}, \boldsymbol{t}, \rho, -1) + R_2(\boldsymbol{s},\boldsymbol{t}),$$
$$\sup_{0 \le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2 \le 1} \boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t})^\top g_m^1(\theta, \boldsymbol{x}, \boldsymbol{t}, -\rho, 1) + \boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t})^\top g_m^1(\theta, \boldsymbol{x}, \boldsymbol{t}, -\rho, -1) + R_2(\boldsymbol{s},\boldsymbol{t}) \Bigg\}.$$

By repeatedly applying Lemma A.2.4 for each $\ell = 2, \dots, L-1$ we obtain

$$\sup_{0\le \boldsymbol{s}_L,\dots,\boldsymbol{s}_2\le 1} \rho\|(\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t})-\boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1\|_\infty + (\boldsymbol{p}_2(\boldsymbol{s},\boldsymbol{t})-\boldsymbol{q}_2(\boldsymbol{s},\boldsymbol{t}))^\top \boldsymbol{W}^1\boldsymbol{x} + R_1(\boldsymbol{s},\boldsymbol{t})$$

$$= \max_{m\in[M]} \max\Bigg\{ \sup_{0\le \boldsymbol{s}_L\le 1} \boldsymbol{p}_L(\boldsymbol{s},\boldsymbol{t})^\top g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},\rho,1) + \boldsymbol{q}_L(\boldsymbol{s},\boldsymbol{t})^\top g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},\rho,-1),$$
$$\sup_{0\le \boldsymbol{s}_L\le 1} \boldsymbol{p}_L(\boldsymbol{s},\boldsymbol{t})^\top g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},-\rho,1) + \boldsymbol{q}_L(\boldsymbol{s},\boldsymbol{t})^\top g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},-\rho,-1)\Bigg\}$$

$$= \max_{m\in[M]} \max\Bigg\{ \sup_{0\le \boldsymbol{s}_L\le 1} [(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ \big(g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},\rho,1)\odot \boldsymbol{s}_L\big)$$
$$+[-(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ (g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},\rho,-1)\odot \boldsymbol{t}_L),$$
$$\sup_{0\le \boldsymbol{s}_L\le 1} [(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ \big(g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},-\rho,1)\odot \boldsymbol{s}_L\big)$$
$$+[-(\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{W}^L]^+ (g_m^{L-1}(\theta,\boldsymbol{x},\boldsymbol{t},-\rho,-1)\odot \boldsymbol{t}_L)\Bigg\}$$

$$= \max_{m\in[M]} \max\Big\{ g_{k,m}^L(\theta,\boldsymbol{x},\boldsymbol{t},\rho), g_{k,m}^L(\theta,\boldsymbol{x},\boldsymbol{t},-\rho)\Big\} - (\Delta \boldsymbol{e}_k^y)^\top \boldsymbol{b}^L,$$

as desired. □

A.3 Convolutional Neural Networks

While in this paper we only consider feed forward neural networks, it is possible to extend the RUB method to convolutional neural networks that use ReLU and MaxPool activation functions. In fact, Lemma 2.5.3 can be modified as follows:

Lemma A.3.1. *Let $x \in \mathbb{R}^{A\times B}$. Define $MP(x) \in \mathbb{R}^{C\times D}$ as the MaxPool function whose (c,d) coordinate corresponds to $\max_{i\in I_{cd}}\{x_i\}$ for fixed sets of indices I_{cd}, and denote by $\circledast$ the convolution operation. If $\boldsymbol{u} \in \mathbb{R}^{A\times B}$, $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{C\times D}$ have all nonnegative coordinates, then the functions $f(\boldsymbol{x}) = \boldsymbol{p} \circledast MP[\boldsymbol{x}]^+$ and $g(\boldsymbol{x}) = \boldsymbol{x} \circledast \boldsymbol{u} - \boldsymbol{q} \circledast MP[\boldsymbol{x}]^+$ satisfy*

$$a)\ f^\star(\boldsymbol{z}) = \begin{cases} 0 & \text{if } 0 \le \sum_{i\in I_{cd}} z_i \le p_{cd}, \forall\, c \in [C], d \in [D], \\ \infty & \text{otherwise,} \end{cases} \quad \text{and}$$

$$b)\ g_\star(\boldsymbol{z}) = \begin{cases} 0 & \text{if } u_{cd} - q_{cd} \le \sum_{i\in I_{cd}} z_i \le u_{cd}, \forall\, c \in [C], d \in [D], \\ -\infty & \text{otherwise.} \end{cases}$$

The lemma above allows to obtain an upper bound for the adversarial loss of convolutional networks with a very similar proof to that of Theorem A.2.3. However, convolutional networks are notoriously more memory consuming and therefore computation of the robust upper bound requires more resources. We have then left this computation for future work, but we here report results for the other robust training methods using these more complex neural networks.

We evaluate a convolutional neural network (denoted as *CNN*) that has been commonly used in previous works of adversarial robustness Madry et al. (2019). It has two convolutional layers alternated with pooling operations, and two dense layers. We compare adversarial accuracy across four different methods: aRUB-L_∞, aRUB-L_1, PGD-L_∞ and Nominal training. Results for the CIFAR data set are shown in Table A.1. We observe that the proposed methods aRUB-L_1 and aRUB-L_∞ yield the highest adversarial accuracies with respect to

PGD-L_2 attacks. For the MNIST data set and the FASHION MNIST data set (Table A.2 and Table A.3, respectively), we see that aRUB-L_∞ has the highest adversarial accuracies with respect to PGD-L_2 attacks when $\rho \leq 0.1$, whereas PGD-L_∞ does best for larger values of ρ.

Table A.1: Adversarial Accuracy for CIFAR with CNN architecture and PGD-L_2 attacks.

$\boldsymbol{\rho} =$	0.000	0.010	0.020	0.030	0.100	1.000	3.000	5.000	10.000
aRUB-L_1	71.17	70.98	70.70	70.51	**69.88**	**60.31**	**44.69**	**35.35**	**17.89**
aRUB-L_∞	**72.03**	**71.76**	**71.45**	**71.25**	69.84	56.13	43.98	34.26	16.25
PGD-L_∞	70.78	70.55	70.43	70.39	69.65	59.96	43.59	29.10	16.21
Nominal	71.29	71.13	71.05	70.66	69.38	48.16	16.05	10.04	10.04

Table A.2: Adversarial Accuracy for Fashion MNIST with CNN architecture and PGD-L_2 attacks.

$\boldsymbol{\rho} =$	0.000	0.010	0.020	0.030	0.100	1.000	3.000	5.000	10.000
aRUB-L_1	91.25	91.09	90.78	90.55	89.02	82.46	68.95	54.80	33.20
aRUB-L_∞	**91.37**	**91.13**	**91.05**	**90.86**	**90.59**	82.81	71.45	60.23	30.86
PGD-L_∞	90.59	90.55	90.43	90.23	89.30	**84.45**	**73.20**	**67.54**	**57.77**
Nominal	91.02	90.90	90.62	90.43	89.02	77.85	56.72	40.51	10.16

Table A.3: Adversarial Accuracy for MNIST with CNN architecture and PGD-L_2 attacks.

$\boldsymbol{\rho} =$	0.000	0.010	0.020	0.030	0.100	1.000	3.000	5.000	10.000
aRUB-L_1	**99.38**	**99.38**	**99.38**	99.34	**99.30**	98.32	91.68	70.51	33.83
aRUB-L_∞	**99.38**	**99.38**	**99.38**	**99.38**	**99.30**	**98.79**	95.70	87.46	47.93
PGD-L_∞	99.22	99.22	99.14	99.14	99.02	98.55	**96.37**	**91.29**	**55.39**
Nominal	99.30	99.26	99.26	99.26	99.18	97.93	89.69	60.00	9.34

Appendix B

Chapter 3 Appendix

B.1 Results Tables

We present the evaluation results for natural accuracy, adversarial accuracy, stability, and sparsity on the test sets across all data sets and methods discussed in the paper. The natural accuracy results can be found in Table B.1, adversarial accuracy results in Table B.2, stability results in Table B.3, and sparsity results in Table B.4.

Name	n	p	k	DL	Rob.	St.	Sp.	Rob. +Sp.	St. +Sp.	St. +Rob.	HDL
echocardiogram	131	7	3	40.00	42.22	42.96	40.00	42.22	42.22	37.78	36.30
hill-valley	605	100	2	47.21	47.54	53.61	47.70	47.87	52.30	51.48	51.64
planning-relax	181	12	2	53.51	52.97	54.05	54.05	54.05	54.05	54.59	54.05
poker-hand	25009	10	10	54.87	55.24	55.09	54.89	54.61	54.41	55.19	54.17
hill-valley-noise	605	100	2	56.23	53.44	59.67	57.87	51.31	53.11	53.61	50.82
yeast	1483	8	10	58.45	59.06	59.12	59.80	60.88	59.80	59.46	60.54
haberman-survival	305	3	2	62.58	61.94	61.61	63.23	61.61	61.94	60.97	62.90
glass-identification	213	9	6	64.65	59.53	65.58	61.40	61.40	63.72	61.40	61.40
brst-cancer-ws-prog	197	32	2	71.00	70.50	72.50	71.50	73.00	69.50	71.50	66.50
hayes-roth	131	4	3	74.81	79.26	77.78	73.33	82.22	77.78	74.81	79.26
spectf-heart	79	44	2	76.25	86.25	73.75	73.75	80.00	75.00	83.75	87.50
hepatitis	154	19	2	78.71	80.00	77.42	78.71	79.35	79.35	77.42	79.35
connectionist-bench	989	10	11	79.19	80.30	83.54	78.08	72.32	74.44	83.23	76.67
libras-movement	359	90	15	79.44	82.50	80.00	75.83	80.56	77.22	79.44	80.00
bld-transf-serv-ctr	747	4	2	79.47	79.07	79.07	79.33	77.20	78.93	79.20	79.60
connect-bench-sonar	207	60	2	83.33	86.67	89.52	86.19	85.71	88.10	83.81	86.67
image-segmentation	209	19	7	83.81	87.14	84.29	83.33	89.52	84.76	87.14	83.81
ecoli	335	7	8	84.71	84.41	85.29	85.59	85.59	85.29	84.71	85.00
qsar-biodegradation	1054	41	2	85.12	85.69	85.21	84.55	84.74	85.21	84.93	84.36
parkinsons	194	21	2	86.67	85.64	87.18	88.72	86.15	86.15	86.67	85.13
magic-gamma-tel	19019	10	2	87.11	87.20	86.98	87.17	86.66	86.50	87.28	86.80
letter-recognition	19999	16	26	90.26	89.57	91.15	82.63	87.60	86.31	90.84	88.79
statlog-proj-landsat	4434	36	6	90.39	90.67	90.64	90.44	89.97	90.53	90.46	90.03
wall-robot-nav-24	5455	24	4	92.36	92.23	92.11	92.49	92.89	92.88	92.47	93.08
spambase	4600	57	2	92.73	93.36	93.42	92.94	92.81	93.03	93.12	92.83
seeds	209	7	3	92.86	92.38	91.43	93.33	93.33	92.86	92.38	93.33
ozone-level-eight	2533	72	2	93.69	93.06	93.89	93.37	93.73	93.49	93.69	93.96
cnae-9	1079	856	9	93.70	93.98	94.07	92.87	93.24	92.59	93.89	92.87
balance-scale	624	4	3	94.24	92.96	93.60	91.68	88.64	93.12	94.24	94.56
ionosphere	350	34	2	94.93	94.93	94.37	95.21	91.83	93.24	95.77	94.08
brst-cancer-ws-orig	698	9	2	96.00	96.29	95.86	96.14	96.86	96.00	96.43	96.29
brst-cancer-ws-diag	568	30	2	97.37	96.49	97.37	96.49	96.49	96.14	97.19	96.67
ozone-level-one	2535	72	2	97.40	97.13	97.20	97.01	97.44	97.44	97.28	97.28
wall-robot-nav-4	5455	4	4	97.77	97.82	97.69	98.04	98.44	98.13	97.80	98.33
climate-simu-crash	539	18	2	97.78	97.59	97.78	96.67	96.85	96.67	97.59	97.04
optical-recog-digits	3822	64	10	97.83	98.01	97.73	97.59	98.09	98.07	98.14	97.93
wall-robot-nav-2	5455	2	4	97.88	98.13	98.02	98.30	98.13	98.30	98.33	98.39
dermatology	365	34	6	98.38	98.38	98.38	98.65	98.38	98.38	98.38	98.92
thyroid-disease-new	214	5	3	98.60	99.07	98.14	98.60	99.07	96.74	99.07	97.67
thyroid-disease-ann	3771	21	3	98.86	98.91	98.78	98.70	98.89	98.86	99.05	98.91
wine	177	13	3	98.89	98.33	98.89	100.00	97.78	97.22	97.78	98.33
pen-recog-digits	7493	16	10	99.23	99.05	99.16	99.11	99.41	99.31	99.19	99.41
skin-segmentation	245056	3	2	99.91	99.90	99.90	99.90	99.88	99.89	99.91	99.89
banknote-authent	1371	4	2	99.93	100.00	100.00	99.85	97.16	89.75	100.00	95.71
iris	149	4	3	100.00	99.33	98.00	100.00	92.67	98.67	99.33	99.33
MNIST	70000	784	10	99.19	99.29	99.21	99.13	99.22	99.16	99.31	99.30
Fashion-MNIST	70000	784	10	91.77	91.88	91.93	91.54	92.12	91.43	92.00	92.15
CIFAR10	60000	1024	10	68.42	69.15	68.25	66.60	57.74	69.82	68.78	69.03

Table B.1: Natural accuracy results for all UCI and vision data sets, where n denotes the data size, p denotes the data dimension, and k denotes the number of classes. Darker blue corresponds to higher nominal DL natural accuracy for the UCI data sets.

Name	n	p	k	DL	Rob.	St.	Sp.	Rob. +Sp.	St. +Sp.	St. +Rob.	HDL
echocardiogram	131	7	3	4.44	44.44	7.41	17.78	44.44	17.78	41.48	44.44
hill-valley	605	100	2	7.54	40.16	28.52	20.00	39.02	20.16	38.52	36.39
planning-relax	181	12	2	21.62	54.05	44.86	49.19	54.05	54.05	54.05	54.05
poker-hand	25009	10	10	50.70	50.70	50.70	50.70	50.70	50.70	50.70	51.95
hill-valley-noise	605	100	2	11.31	21.48	22.30	12.79	30.00	20.98	24.10	34.43
yeast	1483	8	10	0.47	32.26	0.27	0.20	32.26	0.00	32.26	32.26
haberman-survival	305	3	2	17.74	59.68	20.97	47.74	59.68	34.19	59.68	59.68
glass-identification	213	9	6	4.65	20.93	5.58	5.12	20.93	9.30	25.58	25.58
brst-cancer-ws-prog	197	32	2	18.00	72.50	21.00	43.50	72.50	39.50	72.50	72.50
hayes-roth	131	4	3	9.63	32.59	8.89	5.19	36.30	13.33	40.74	40.74
spectf-heart	79	44	2	27.50	25.00	32.50	40.00	25.00	13.75	21.25	25.00
hepatitis	154	19	2	5.81	70.97	9.03	21.94	70.97	29.68	70.97	70.97
connectionist-bench	989	10	11	4.04	1.52	4.34	1.41	6.46	3.84	8.79	8.99
libras-movement	359	90	15	0.00	0.83	0.28	0.00	2.50	0.00	4.72	2.50
bld-transf-serv-ctr	747	4	2	0.00	72.67	0.13	17.60	72.67	43.60	72.67	72.67
connect-bench-sonar	207	60	2	5.71	20.00	18.10	32.38	43.81	33.81	22.86	40.00
image-segmentation	209	19	7	4.29	0.95	2.86	2.38	9.52	3.33	9.52	8.10
ecoli	335	7	8	0.59	41.18	1.76	2.06	51.47	0.88	41.18	51.47
qsar-biodegradation	1054	41	2	36.21	72.04	24.08	31.75	72.04	29.67	72.04	72.04
parkinsons	194	21	2	58.97	74.36	63.08	63.08	74.36	68.72	74.36	74.36
magic-gamma-tel	19019	10	2	15.07	64.59	15.28	18.62	64.59	10.36	64.59	64.59
letter-recognition	19999	16	26	0.76	3.73	1.11	0.52	3.68	0.34	3.57	3.74
statlog-proj-landsat	4434	36	6	8.66	25.48	10.35	7.51	25.48	5.77	20.79	25.39
wall-robot-nav-24	5455	24	4	3.04	39.69	3.28	1.63	39.69	3.06	39.69	39.65
spambase	4600	57	2	40.67	58.41	48.08	45.06	58.41	49.25	58.41	58.41
seeds	209	7	3	23.81	24.29	31.90	27.14	32.38	15.24	31.43	30.95
ozone-level-eight	2533	72	2	49.59	94.48	48.88	94.48	94.48	56.69	94.48	94.48
cnae-9	1079	856	9	0.00	1.02	0.09	0.83	1.94	2.50	3.80	3.80
balance-scale	624	4	3	17.44	40.80	13.28	8.16	43.84	14.72	49.92	49.92
ionosphere	350	34	2	19.44	57.75	25.35	34.65	57.75	10.99	57.75	57.75
brst-cancer-ws-orig	698	9	2	17.71	60.71	23.71	31.29	60.71	15.14	60.71	60.71
brst-cancer-ws-diag	568	30	2	25.44	47.02	25.61	47.02	58.77	13.33	47.02	58.77
ozone-level-one	2535	72	2	73.78	97.44	81.22	97.44	97.44	89.17	97.44	97.44
wall-robot-nav-4	5455	4	4	3.42	39.69	4.85	7.69	39.69	10.49	39.69	39.69
climate-simu-crash	539	18	2	0.00	94.44	2.41	75.56	94.44	83.33	94.44	94.44
optical-recog-digits	3822	64	10	1.36	7.48	1.39	0.76	7.16	0.60	8.94	10.27
wall-robot-nav-2	5455	2	4	5.26	39.69	5.48	17.75	39.69	21.03	39.69	39.69
dermatology	365	34	6	4.59	20.27	2.97	5.68	20.27	10.81	20.27	20.27
thyroid-disease-new	214	5	3	9.77	65.12	10.23	13.02	65.12	13.02	65.12	65.12
thyroid-disease-ann	3771	21	3	48.42	91.79	54.97	54.12	91.79	55.23	91.79	91.79
wine	177	13	3	1.67	44.44	1.11	3.89	37.78	5.56	43.33	31.67
pen-recog-digits	7493	16	10	2.76	10.13	2.63	2.05	10.13	1.80	8.71	10.27
skin-segmentation	245056	3	2	79.30	79.30	79.30	79.30	79.30	79.28	79.30	79.30
banknote-authent	1371	4	2	39.20	54.25	42.25	54.25	54.25	43.13	54.25	54.25
iris	149	4	3	12.00	16.00	12.00	10.00	24.67	10.00	32.67	32.67
MNIST	70000	784	10	49.60	78.37	51.52	43.80	74.67	39.94	79.50	75.97
Fashion-MNIST	70000	784	10	78.70	87.74	78.30	80.76	87.07	80.24	87.80	86.91
CIFAR10	60000	1024	10	28.81	48.61	28.75	34.87	43.98	33.73	47.32	43.96

Table B.2: Adversarial accuracy results for all UCI and vision data sets, where n denotes the data size, p denotes the data dimension, and k denotes the number of classes. We use $\rho = 0.1$ for all data sets except CIFAR10 and Fashion-MNIST, for which we set $\rho = 0.01$. Darker blue corresponds to higher nominal (DL) natural accuracy.

Name	n	p	k	DL	Rob.	St.	Sp.	Rob. +Sp.	St. +Sp.	St. +Rob.	HDL
echocardiogram	131	7	3	40.74	37.04	37.04	40.74	33.33	33.33	40.74	33.33
hill-valley	605	100	2	43.44	43.44	49.18	43.44	46.72	50.82	45.08	49.18
planning-relax	181	12	2	51.35	48.65	54.05	54.05	54.05	51.35	54.05	54.05
poker-hand	25009	10	10	54.60	54.32	54.56	54.48	53.20	53.52	54.70	52.32
hill-valley-noise	605	100	2	54.92	47.54	52.46	54.92	47.54	47.54	48.36	50.00
yeast	1483	8	10	57.58	56.90	56.57	57.24	59.60	58.92	56.90	58.92
haberman-survival	305	3	2	59.68	59.68	59.68	59.68	59.68	59.68	59.68	59.68
glass-identification	213	9	6	55.81	58.14	53.49	58.14	55.81	58.14	58.14	58.14
brst-cancer-ws-prog	197	32	2	67.50	67.50	67.50	67.50	62.50	60.00	72.50	67.50
hayes-roth	131	4	3	70.37	74.07	70.37	70.37	70.37	74.07	70.37	74.07
spectf-heart	79	44	2	68.75	68.75	75.00	68.75	68.75	62.50	75.00	75.00
hepatitis	154	19	2	70.97	70.97	70.97	70.97	74.19	70.97	70.97	77.42
connectionist-bench	989	10	11	75.25	77.78	80.81	71.72	63.13	60.61	76.26	70.71
libras-movement	359	90	15	75.00	75.00	79.17	68.06	75.00	70.83	81.94	75.00
bld-transf-serv-ctr	747	4	2	79.33	77.33	78.00	78.67	74.67	74.00	78.00	79.33
connect-bench-sonar	207	60	2	78.57	80.95	83.33	80.95	83.33	83.33	80.95	83.33
image-segmentation	209	19	7	78.57	78.57	80.95	76.19	73.81	83.33	78.57	78.57
ecoli	335	7	8	80.88	77.94	83.82	82.35	83.82	83.82	83.82	82.35
qsar-biodegradation	1054	41	2	84.83	84.83	83.89	81.99	83.41	83.89	84.36	83.89
parkinsons	194	21	2	84.62	82.05	82.05	84.62	79.49	76.92	84.62	79.49
magic-gamma-tel	19019	10	2	86.65	86.93	86.44	86.75	85.73	86.01	87.01	86.25
letter-recognition	19999	16	26	86.98	86.75	89.60	82.27	84.47	83.85	90.20	86.55
statlog-proj-landsat	4434	36	6	88.84	88.05	90.08	90.19	88.61	89.40	89.29	88.39
wall-robot-nav-24	5455	24	4	92.03	91.12	91.58	92.31	92.03	92.22	90.75	92.40
spambase	4600	57	2	92.18	92.73	92.62	92.29	92.40	92.51	92.40	92.62
seeds	209	7	3	90.48	90.48	92.86	92.86	92.86	90.48	90.48	92.86
ozone-level-eight	2533	72	2	93.10	94.08	93.10	93.29	94.48	92.31	93.10	93.49
cnae-9	1079	856	9	91.20	93.06	92.59	92.13	92.59	92.13	93.06	91.20
balance-scale	624	4	3	91.20	91.20	92.00	90.40	72.80	91.20	91.20	92.80
ionosphere	350	34	2	92.96	90.14	92.96	91.55	94.37	91.55	90.14	91.55
brst-cancer-ws-orig	698	9	2	93.57	93.57	95.71	95.71	95.71	95.00	95.71	95.71
brst-cancer-ws-diag	568	30	2	95.61	94.74	95.61	94.74	95.61	93.86	94.74	95.61
ozone-level-one	2535	72	2	97.24	96.46	96.85	96.85	97.44	97.44	96.85	97.05
wall-robot-nav-4	5455	4	4	97.44	97.16	97.34	97.25	97.71	97.62	97.62	97.80
climate-simu-crash	539	18	2	96.30	96.30	95.37	94.44	96.30	93.52	96.30	96.30
optical-recog-digits	3822	64	10	97.39	97.39	96.99	97.12	97.91	97.91	97.78	97.65
wall-robot-nav-2	5455	2	4	96.98	97.53	97.34	97.25	97.53	98.17	97.89	97.44
dermatology	365	34	6	97.30	97.30	97.30	98.65	95.95	97.30	97.30	94.59
thyroid-disease-new	214	5	3	97.67	97.67	95.35	93.02	95.35	93.02	97.67	93.02
thyroid-disease-ann	3771	21	3	98.28	98.81	98.01	98.54	98.68	98.41	98.54	98.28
wine	177	13	3	94.44	97.22	94.44	94.44	97.22	88.89	97.22	97.22
pen-recog-digits	7493	16	10	98.93	98.67	99.00	99.07	99.13	99.33	99.07	99.07
skin-segmentation	245056	3	2	99.90	99.89	99.89	99.89	99.86	99.87	99.90	99.87
banknote-authent	1371	4	2	99.64	100.00	100.00	99.64	87.27	70.18	100.00	87.27
iris	149	4	3	100.00	100.00	93.33	100.00	70.00	96.67	96.67	96.67
MNIST	70000	784	10	99.14	99.24	99.15	99.06	99.07	98.86	99.25	99.19
Fashion-MNIST	70000	784	10	91.53	91.36	91.75	91.29	91.65	91.15	91.38	90.19
CIFAR10	60000	1024	10	68.28	65.44	68.13	63.14	56.04	61.76	63.76	56.48

Table B.3: Stability (worst case accuracy) results for all UCI and vision data sets, where n denotes the data size, p denotes the data dimension, and k denotes the number of classes. Darker blue corresponds to higher nominal (DL) natural accuracy.

	Name	***n***	***p***	***k***	**DL**	**Rob.**	**St.**	**Sp.**	**Rob. +Sp.**	**St. +Sp.**	**St. +Rob.**	**HDL**
🟦	echocardiogram	131	7	3	0.0	0.0	0.0	52.17	62.57	65.69	0.0	65.49
🟦	hill-valley	605	100	2	0.0	0.0	0.0	49.00	41.94	34.89	0.0	28.80
🟦	planning-relax	181	12	2	0.0	0.0	0.0	54.47	63.00	69.13	0.0	70.56
🟦	poker-hand	25009	10	10	0.0	0.0	0.0	29.56	49.23	51.31	0.0	49.83
🟦	hill-valley-noise	605	100	2	0.0	0.0	0.0	50.67	39.73	45.04	0.0	29.28
🟦	yeast	1483	8	10	0.0	0.0	0.0	53.08	53.41	54.71	0.0	54.36
🟦	haberman-survival	305	3	2	0.0	0.0	0.0	47.21	49.40	53.34	0.0	53.10
🟦	glass-identification	213	9	6	0.0	0.0	0.0	56.14	63.65	64.86	0.0	64.85
🟦	brst-cancer-ws-prog	197	32	2	0.0	0.0	0.0	61.87	67.34	69.49	0.0	67.78
🟦	hayes-roth	131	4	3	0.0	0.0	0.0	61.56	67.49	66.39	0.0	67.38
🟦	spectf-heart	79	44	2	0.0	0.0	0.0	81.02	85.84	86.10	0.0	85.58
🟦	hepatitis	154	19	2	0.0	0.0	0.0	64.56	68.71	74.01	0.0	72.74
🟦	connectionist-bench	989	10	11	0.0	0.0	0.0	57.95	63.20	63.34	0.0	63.45
🟦	libras-movement	359	90	15	0.0	0.0	0.0	66.76	70.90	71.00	0.0	70.73
🟦	bld-transf-serv-ctr	747	4	2	0.0	0.0	0.0	46.21	48.90	52.65	0.0	50.52
🟦	connect-bench-sonar	207	60	2	0.0	0.0	0.0	76.09	80.04	81.10	0.0	80.07
🟦	image-segmentation	209	19	7	0.0	0.0	0.0	60.95	66.47	69.23	0.0	67.24
🟦	ecoli	335	7	8	0.0	0.0	0.0	54.91	62.25	63.26	0.0	62.31
🟦	qsar-biodegradation	1054	41	2	0.0	0.0	0.0	44.89	44.32	51.82	0.0	47.65
🟦	parkinsons	194	21	2	0.0	0.0	0.0	59.42	60.95	67.40	0.0	63.23
🟦	magic-gamma-tel	19019	10	2	0.0	0.0	0.0	50.80	51.43	57.52	0.0	52.84
🟦	letter-recognition	19999	16	26	0.0	0.0	0.0	58.77	63.84	65.32	0.0	64.86
🟦	statlog-proj-landsat	4434	36	6	0.0	0.0	0.0	45.65	50.60	52.06	0.0	51.53
🟦	wall-robot-nav-24	5455	24	4	0.0	0.0	0.0	51.85	55.96	56.59	0.0	55.53
🟦	spambase	4600	57	2	0.0	0.0	0.0	56.81	56.56	59.90	0.0	55.84
🟦	seeds	209	7	3	0.0	0.0	0.0	65.56	71.45	72.73	0.0	72.33
🟦	ozone-level-eight	2533	72	2	0.0	0.0	0.0	66.09	68.13	67.86	0.0	67.38
🟦	cnae-9	1079	856	9	0.0	0.0	0.0	61.94	62.06	73.43	0.0	61.91
🟦	balance-scale	624	4	3	0.0	0.0	0.0	57.25	63.41	63.35	0.0	63.14
🟦	ionosphere	350	34	2	0.0	0.0	0.0	63.16	66.76	68.17	0.0	66.74
🟦	brst-cancer-ws-orig	698	9	2	0.0	0.0	0.0	48.29	53.44	55.78	0.0	55.97
🟦	brst-cancer-ws-diag	568	30	2	0.0	0.0	0.0	68.91	71.38	71.46	0.0	70.69
🟦	ozone-level-one	2535	72	2	0.0	0.0	0.0	66.22	68.25	67.97	0.0	68.58
🟦	wall-robot-nav-4	5455	4	4	0.0	0.0	0.0	52.89	61.43	60.75	0.0	60.50
🟦	climate-simu-crash	539	18	2	0.0	0.0	0.0	59.75	64.56	65.88	0.0	66.29
🟦	optical-recog-digits	3822	64	10	0.0	0.0	0.0	65.42	68.85	71.25	0.0	69.22
🟦	wall-robot-nav-2	5455	2	4	0.0	0.0	0.0	63.85	72.12	72.68	0.0	71.74
🟦	dermatology	365	34	6	0.0	0.0	0.0	67.03	73.49	74.61	0.0	74.21
🟦	thyroid-disease-new	214	5	3	0.0	0.0	0.0	67.25	74.48	74.62	0.0	74.12
🟦	thyroid-disease-ann	3771	21	3	0.0	0.0	0.0	48.81	50.15	53.22	0.0	50.36
🟦	wine	177	13	3	0.0	0.0	0.0	74.62	81.05	80.60	0.0	80.59
🟦	pen-recog-digits	7493	16	10	0.0	0.0	0.0	59.13	62.58	63.33	0.0	63.23
🟦	skin-segmentation	245056	3	2	0.0	0.0	0.0	63.77	71.27	72.15	0.0	72.56
🟦	banknote-authent	1371	4	2	0.0	0.0	0.0	76.88	84.17	84.97	0.0	84.98
🟦	iris	149	4	3	0.0	0.0	0.0	69.89	78.03	77.56	0.0	77.79
🟥	MNIST	70000	784	10	0.0	0.0	0.0	39.20	75.06	44.77	0.0	76.05
🟥	Fashion-MNIST	70000	784	10	0.0	0.0	0.0	45.41	68.94	48.40	0.0	69.18
🟥	CIFAR10	60000	1024	10	0.0	0.0	0.0	50.51	82.34	55.58	0.0	81.41

Table B.4: Sparsity results for all UCI and vision data sets, where n denotes the data size, p denotes the data dimension, and k denotes the number of classes. Darker blue corresponds to higher nominal (DL) natural accuracy.

Appendix C

Chapter 4 Appendix

C.1 HHC Data Summary Statistics

We report the summary statistics of the data after the process of inclusion, exclusion, and splits from the Machine Learning Modeling section. For each hospital HA-HG, the number of patients, admissions, and patient days in the union of training, validation, and testing sets are summarized in Table C.1.

Table C.1: Summary of Data Size.

Hospital	HA	HB	HC	HD	HE	HF	HG
# Patients	105,184	15,493	7,956	20,011	15,576	11,624	4,838
# Admissions	171,072	23,354	12,822	29,490	21,612	15,319	6,924
# patient days	879,357	106,662	52,931	139,542	90,924	79,615	26,184

C.2 Accuracy for each Hospital at HHC

We report the accuracy, precision, and recall for green and red alerts at all seven hospitals in Table C.2. Among the hospitals, green alerts have 0.687-0.768 accuracy, 0.588-0.629 precision, and 0.701-0.8 recall; red alerts have 0.885-0.925 accuracy, 0.477-0.55 precision, and 0.471-0.715 recall.

Table C.2: Precision and Recall under Selected Thresholds for Alerts.

Alert	Hospital	HA	HB	HC	HD	HE	HF	HG
New green	Accuracy	0.767	0.734	0.714	0.751	0.739	0.768	0.687
	Precision	0.621	0.604	0.588	0.611	0.623	0.617	0.629
	Recall	0.746	0.786	0.8	0.764	0.796	0.701	0.78
Red	Accuracy	0.899	0.901	0.895	0.881	0.896	0.886	0.925
	Precision	0.477	0.55	0.574	0.492	0.528	0.505	0.53
	Recall	0.705	0.668	0.553	0.691	0.715	0.663	0.471

C.3 Empirical Treatment Effect for HHC

Table C.3 presents information and deployment progress of all units with general level of care and offering cardiology, medicine or surgical services. By January 16, 2023, 15 treatment units had fully integrated the predictions in their review process, where unit leads review the predictions with the provider team daily and adjust decisions accordingly. As of April 15, 2023, 12 control units (with an NA Start Date) had not officially integrated the daily process.

We consider a linear regression approach to estimate the impact of the adoption of the tool on four different outcomes (time is measured in days): average LOS in the unit (Avg. LOS), its logarithm (log(Avg. LOS)), the average of the log-LOS in the unit (Avg. log(LOS)), and the average time difference between the discharge order and the first occurrence of the green alert (we refer to this outcome as $\Delta_{\text{order - alert}}$).

We use discharge data from all half-month periods in January 16, 2020 - April 15, 2023, and for the same treatment and control units as in Section 4.2.3. We consider the unit to be not treated (resp. treated) on all time periods before (resp. after) the utilization start date from Table C.3. Periods containing the utilization start date are excluded. We include two categories of control variables: calendar variables (year-month-half indicators) and unit controls (hospital name, unit name, and the number of beds). We then construct a linear regression model to predict each outcome as a function of a binary variable indicating whether the tool is deployed in that unit at that time period, and the control variables. Formally, for

Table C.3: Unit Deployment Progress Information.

Hospital	Unit	Start Date	Specialty	Capacity
HA	HA CONKLIN 2	NA	Medicine/Oncology	27
HA	HA CONKLIN 4	9/13/22	Medicine	25
HA	HA CONKLIN 5	7/11/22	Medicine	47
HA	HA BLISS 7 EAST	8/23/22	Medicine	17
HA	HA BLISS 10 EAST	NA	Cardiology	14
HA	HA CENTER 10	NA	Cardiology	26
HA	HA CENTER 12	7/11/22	Medicine	26
HA	HA NORTH 10	NA	Cardiology	27
HA	HA NORTH 12	7/11/22	Medicine	20
HB	HB A3 MEDSURG	8/23/22	Medicine/Surgical	30
HB	HB E4 Cardiology	8/23/22	Cardiology	28
HC	HC FOURTH FLOOR	8/23/22	Medicine/Surgical	28
HC	HC FIFTH FLOOR	8/23/22	Medicine/Surgical	29
HD	HD EAST 2	NA	Medicine/Observation	12
HD	HD WEST 2	NA	Medicine	15
HD	HD NORTH 3	1/15/23	Medicine	24
HD	HD NORTH 4	10/22/22	Medicine/Cardiology	28
HD	HD NORTH 5	8/23/22	Medicine/Stroke	30
HE	HE PAVILION D	NA	Medicine	28
HE	HE PAVILION E	1/15/23	Medicine	28
HF	HF 6 NORTH	NA	Cardiology	20
HF	HF 6 SOUTH	NA	Cardiology	20
HF	HF 9 NORTH	NA	Medicine	22
HF	HF 10 NORTH	NA	Medicine	29
HG	HG 4 SHEA EAST	1/15/23	Medicine/Surgical	30
HG	HG 4 SHEA NORTH	1/15/23	Medicine/Surgical	12
HG	HG GREER	NA	Medicine/Surgical	23

every unit i and time period t, we consider a regression model of the form

$$Y_{i,t} = \alpha_i + \lambda_t + \mu W_{i,t} + \beta X_i + \varepsilon_{i,t},$$

where $Y_{i,t}$ is the outcome of interest, α_i (resp. λ_t) is the unit (resp. time) fixed effect, $W_{i,t}$ is the binary treatment indicator variable for unit i at time t, and X_i is the set of additional controls for unit i (in our case, hospital and number of beds). The coefficient μ corresponds to the treatment effect, assumed homogeneous across units and time.

We select this approach given our staggered roll-out setup (different units started using the tool at different times). This estimation strategy is referred to in the literature as Difference-in-Difference with variations in treatment times (Callaway and Sant'Anna 2021,

Table C.4: Regression Results of our Difference-in-Difference Model for Estimating the Impact of our Tool after Deployment.

Outcome	Length of Stay Avg. LOS	log(Avg. LOS)	Avg. log(LOS)	Avg. $\Delta_{\text{order - alert}}$
Coefficient	-0.457	-0.082	-0.055	-0.194
Standard Error	0.167	0.028	0.023	0.09
p-value	6.4×10^{-3}	3.9×10^{-3}	1.7×10^{-2}	3.3×10^{-2}
Obs.	1846	1846	1846	1827
R-square	0.394	0.454	0.508	0.424
Adjusted R-square	0.357	0.421	0.479	0.389

Controls: hospital, unit, number of beds, month-year-half.
Standard Errors: cluster-robust at the unit level.

Goodman-Bacon 2021), and it accounts for variations in LOS due to time non-stationarity (calendar variables) and differences in patient populations and hospital types (hospital and unit fixed effects). Given the correlation in our setting between the units and the treatment assignment, the default standard errors could overestimate the precision of the estimator (Cameron and Miller 2015), so we compute cluster-robust standard errors (Liang and Zeger 1986) at a unit-level instead.

The results of the linear regression model are shown in Table C.4. The treatment variable has a negative and statistically significant coefficient across all outcomes. In particular, for the Avg. LOS, the coefficient value of -0.457 (p-value of 0.006) indicates that, everything else (i.e., all the control variables) being equal, using the tool reduces the average LOS by 0.457 days. Similarly, regressing the average of the log(LOS) or the logarithm of the average LOS indicates a statistically significant negative treatment effect, yet in multiplicative rather than additive terms (around 8–5% reduction in LOS). Lastly, the coefficient of the treatment variable for the Avg. ($\Delta_{\text{order - alert}}$) outcome has a value of -0.194 (p-value of 0.033), suggesting that the reduction in LOS can be partially attributed to discharge orders being placed earlier (0.194 days sooner), potentially thanks to observing the green alert.

There are limitations to our approach, including some discussed in Section 4.2.3. Our analysis assumes that the treatment effect on the outcomes of interest can be decomposed into two-way fixed effects and follows a linear relationship with the control and treatment variables. However, there may be other confounding factors at the unit and patient levels that can influence LOS, such as patient demographics and other initiatives aimed at reducing LOS in these hospitals. Most importantly, the design of the staggered roll-out was not done with empirical validation in mind. The units treated and the time where deployment started in these units at random and may be correlated with LOS and the potential impact of the tool (e.g., prioritizing units that would benefit the most or would be ‘easy adopters’). This deficiency can lead to bias in our DiD estimates, in addition to issues of treatment effect heterogeneity across units and over time (Bertrand et al. 2004). Lastly, discussions with the hospital network reveal challenges in accurately measuring and quantifying the exact extent of tool usage. For instance, treatment units exhibit varying degrees of tool utilization across different medical teams.

Appendix D

Chapter 5 Appendix

D.1 Reproducing Kernel Hilbert Spaces Overview

A reproducing kernel Hilbert space (RKHS) is a Hilbert space in which the elements are functions that preserve pointwise distance. Specifically, if two functions are close with respect to the Hilbert space norm, then their pointwise evaluations are close with respect to the norm of the functions' output space. Each RKHS is generated by a positive definite kernel $K(\cdot,\cdot)$; a function $K:\ \mathcal{Z}\times\mathcal{Z}\to\mathbb{R}$ satisfying

$$\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j K(\boldsymbol{z}^i,\boldsymbol{z}^j)\geq 0,\quad \forall m\in\mathbb{N},\quad \boldsymbol{z}^1,\ldots,\boldsymbol{z}^m\in\mathcal{Z},\quad a_1,\ldots,a_m\in\mathbb{R}\,.$$

Definition D.1.0.1. *A reproducing kernel Hilbert space $\mathcal{H}$ generated by a positive definite kernel $K:\mathcal{Z}\times\mathcal{Z}\to\mathbb{R}$ is the closure of the set of functions*

$$\left\{ f:\mathcal{Z}\to\mathbb{R}\ \middle|\ f(\boldsymbol{z})=\sum_{c=1}^{L} a^c K(\boldsymbol{z}^c,\boldsymbol{z}),\ \textit{for}\ \boldsymbol{z}^1,\ldots,\boldsymbol{z}^L\in\mathcal{Z}\ \textit{and}\ L\in\mathbb{N}\right\},$$

with inner product of $f_1(\boldsymbol{z}) = \sum_{c=1}^{L_1} a_1^c K(\boldsymbol{z}_1^c, \boldsymbol{z})$ *and* $f_2(\boldsymbol{z}) = \sum_{c=1}^{L_2} a_2^c K(\boldsymbol{z}_2^c, \boldsymbol{z})$ *defined as*

$$\langle f_1, f_2 \rangle_{\mathcal{H}} = \sum_{c_1=1}^{L_1} \sum_{c_2=1}^{L_2} a_1^{c_1} a_2^{c_2} K(\boldsymbol{z}_1^{c_1}, \boldsymbol{z}_2^{c_2}).$$

The complexity of a reproducing kernel Hilbert space depends on the kernel generating it. A linear kernel, for example, generates the Hilbert space of linear functions. A Gaussian kernel generates much more complex spaces; it has the property that for compact spaces $\mathcal{Z}$ it generates spaces that are dense in $C(\mathcal{Z})$ (the space of continuous bounded funcions on $\mathcal{Z}$) in the maximum norm. Kernels with this property are called *universal kernels* (Micchelli et al. 2006) and they are very useful for solving functional optimization problems over continuous functions, since the problem can be solved over the RKHS instead.

One of the main reasons why RKHSs are so useful when working with data is the fact that they transform pointwise evaluation into an inner product of elements in the Hilbert space, and vice-versa. Specifically, if f belongs to the reproducing kernel Hilbert space $\mathcal{H}$ generated by kernel K, we have

$$f(\boldsymbol{x}) = \langle K(\boldsymbol{x}, \cdot), f \rangle_{\mathcal{H}}, \tag{D.1}$$

for all $\boldsymbol{x}$ in the domain of f. This equivalence is known as the *reproducing property.* The next result, known as the Representer Theorem, illustrates how in many cases solving functional optimization problems over a RKHS is equivalent to solving an optimization problem over a real space, and the proof relies mostly on the reproducing property.

Theorem D.1.1 (Representer Theorem)**.** *Suppose we have a data matrix* $\boldsymbol{Z} = [\boldsymbol{z}^1, \ldots, \boldsymbol{z}^N]$ *for some fixed data points* $\boldsymbol{z}^1, \ldots, \boldsymbol{z}^N \in \mathcal{Z}$. *Let* $\mathcal{H}$ *be the reproducing kernel Hilbert space generated by a kernel* $K : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. *Then, for any arbitrary loss function* $c : \mathbb{R} \times \mathcal{Z} \to \mathbb{R}$

and any regularization parameter $\lambda \geq 0$, there exists a solution to

$$\inf_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} c(h(\boldsymbol{z}^n), \boldsymbol{z}^n) + \frac{\lambda}{2} \|h\|_{\mathcal{H}}^2 \tag{D.2}$$

that takes the form

$$h^*(\cdot) = \sum_{n=1}^{N} a_n K(\boldsymbol{z}^n, \cdot)\,, \tag{D.3}$$

for some scalars $a_1, \ldots, a_N \in \mathbb{R}^N$.

The Representer Theorem implies that the solution to the functional optimization problem (D.2) can be found by solving instead the following finite dimensional optimization problem

$$\min_{\boldsymbol{a} \in \mathbb{R}^N} \frac{1}{N} \sum_{n=1}^{N} c\big((\boldsymbol{K}[\boldsymbol{Z}, \boldsymbol{Z}]\boldsymbol{a})_n, \boldsymbol{z}^n\big) + \frac{\lambda}{2} \boldsymbol{a}^T \boldsymbol{K}[\boldsymbol{Z}, \boldsymbol{Z}]\boldsymbol{a}\,, \tag{D.4}$$

where $\boldsymbol{K}[\boldsymbol{Z}, \tilde{\boldsymbol{Z}}]$ is the kernel matrix (between equal size matrices $\boldsymbol{Z}, \tilde{\boldsymbol{Z}}$) whose (n, m) component is $K(\boldsymbol{z}^n, \tilde{\boldsymbol{z}}^m)$.

The proof of this theorem follows from the fact that any function in $\mathcal{H}$ can be decomposed as the sum of a function of the form (D.3) and a function orthogonal to every function of this form. The theorem then follows after showing that, thanks to the reproducing property, the sum in the objective of (D.2) is independent of the orthogonal part, and the second term in the objective is increasing in the orthogonal part. This theorem can be extended to a multidimensional version in which the optimization problem is over multiple functions $h_1, \ldots, h_r \in \mathcal{H}$. In this case, the Representer Theorem tells us that there exists a solution $\boldsymbol{h}^* \in \mathcal{H}^r$ that takes the form

$$\boldsymbol{h}^*(\cdot) = \boldsymbol{A}\boldsymbol{K}(\boldsymbol{Z}, \cdot)\,, \tag{D.5}$$

where $\boldsymbol{K}(\boldsymbol{Z}, \cdot) := [K(\boldsymbol{z}^1, \cdot), \ldots, K(\boldsymbol{z}^N, \cdot)]^T$ and $\boldsymbol{A} \in \mathbb{R}^{r \times N}$. For more details about the representer theorem's proof and applications we refer the reader to Wahba (1990), Soentpiet et al. (1999), Schölkopf et al. (2002), Shawe-Taylor et al. (2004).

D.2 Lemmas

In this appendix we will state and proof several lemmas needed for the proof of Theorems 5.5.4 and Corollary 5.5.5. For generality, we consider the constrained problem in Eq. 5.19, and define:

$$c^{\psi}\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) := c\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big) + \psi \sum_{q=1}^{Q} \max\Big(0, g_q\big(\boldsymbol{u}(\boldsymbol{z})\big)\Big)^2, \tag{D.6}$$

$$E(\boldsymbol{u}) := \mathbb{E}_{\boldsymbol{z}}\left[c(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z})\right], \tag{D.7}$$

$$E^{\psi}(\boldsymbol{u}) := \mathbb{E}_{\boldsymbol{z}}\left[c^{\psi}\big(\boldsymbol{u}(\boldsymbol{z}), \boldsymbol{z}\big)\right], \tag{D.8}$$

$$E^{\lambda,\psi}(\boldsymbol{u}) := E^{\psi}(\boldsymbol{u}) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2, \tag{D.9}$$

$$E_{\mathcal{S}}^{\lambda,\psi}(\boldsymbol{u}) := \frac{1}{N}\sum_{n=1}^{N} c^{\psi}\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2, \tag{D.10}$$

$$E_{n}^{\lambda,\psi}(\boldsymbol{u}) := c^{\psi}\big(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n\big) + \frac{\lambda}{2}\|\boldsymbol{u}\|_{\mathcal{H}}^2. \tag{D.11}$$

Lemma D.2.1. *Under assumption 1, we have*

$$\|\boldsymbol{u}(\boldsymbol{z})\|_2 \leq \kappa \|\boldsymbol{u}\|_{\mathcal{H}} \quad \forall\ \boldsymbol{z} \in \mathcal{Z},\ \boldsymbol{u} \in \mathcal{H}.$$

Proof. Let $\boldsymbol{u} \in \mathcal{H}$ and $\boldsymbol{z} \in \mathcal{Z}$. We have

$$\begin{aligned}
\|\boldsymbol{u}(\boldsymbol{z})\|_2^2 &= \sum_{t=1}^{T}\sum_{i=1}^{r_t} u_{t,i}(\boldsymbol{z}_{1:t-1})^2 = \sum_{t=1}^{T}\sum_{i=1}^{r_t} \langle u_{t,i}, K_t(\boldsymbol{z}_{1:t-1}, \cdot)\rangle_{\mathcal{H}_t}^2 && \text{(by Eq. (D.1))},\\
&\leq \sum_{t=1}^{T}\sum_{i=1}^{r_t} \|K_t(\boldsymbol{z}_{1:t-1}, \cdot)\|_{\mathcal{H}_{t,i}}^2 \|u_{t,i}\|_{\mathcal{H}_t}^2 && \text{(Cauchy-Schwarts Ineq.)},\\
&\leq \kappa^2 \|\boldsymbol{u}\|_{\mathcal{H}}^2 && \text{(Assumption 5.5.1)},
\end{aligned}$$

and the lemma follows. ∎

Lemma D.2.2. *Under Assumptions 1 and 2, we have* *a) The true minimizer* $\boldsymbol{u}^{\lambda,\psi}$ *of* $E^{\lambda,\psi}$ *defined in D.9, satisfies*

$$\|\boldsymbol{u}^{\lambda,\psi}\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}.$$

b) If Algorithm 2 is initialized such that $\|\boldsymbol{u}^0\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}$*, then*

$$\|\boldsymbol{u}^n\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda} \quad \forall\ n \in [N].$$

Proof. a) We proceed as in Kivinen et al. (2004). We define $\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}$ as the minimizer of $E_{\mathcal{S}}^{\lambda,\psi}$, and $\hat{\boldsymbol{u}} = (1-\epsilon)\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}$ for $\epsilon > 0$. We have

$$\begin{aligned}
0 &\leq E_{\mathcal{S}}^{\lambda,\psi}(\hat{\boldsymbol{u}}) - E_{\mathcal{S}}^{\lambda,\psi}(\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}) && \text{(by Optimality of } \boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\text{)},\\
&= \frac{1}{N}\sum_{n=1}^{N}\Big(c^{\psi}(\hat{\boldsymbol{u}}(\boldsymbol{z}^n),\boldsymbol{z}^n) - c^{\psi}(\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}(\boldsymbol{z}^n),\boldsymbol{z}^n)\Big) + \frac{\lambda}{2}\big(\|\hat{\boldsymbol{u}}\|_{\mathcal{H}}^2 - && \|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2\big)\\
&\leq \frac{C}{N}\sum_{n=1}^{N}\|\hat{\boldsymbol{u}}(\boldsymbol{z}^n) - \boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}(\boldsymbol{z}^n)\|_2 + \frac{\lambda}{2}\big((1-\epsilon)^2 - 1\big)\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2 && \text{(by Assumption 5.5.2)},\\
&\leq \frac{\kappa C}{N}\sum_{n=1}^{N}\|\hat{\boldsymbol{u}} - \boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}} - \lambda\epsilon\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2 + \frac{\lambda}{2}\epsilon^2\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2 && \text{(by Lemma D.2.1)},\\
&= \kappa C\epsilon\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}} - \lambda\epsilon\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2 + \frac{\lambda}{2}\epsilon^2\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}^2.
\end{aligned}$$

Dividing by $\epsilon\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}}$ on both sides and taking the limit as $\epsilon \to 0$ we obtain $\|\boldsymbol{u}_{\mathcal{S}}^{\lambda,\psi}\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}$ and the desired result then follows by taking the limit as $N \to \infty$.

b) To prove the upper bound for the decisions output by the algorithm in each iteration we proceed by induction on the iteration number n. We have $\|\boldsymbol{u}^0\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}$ by assumption.

Suppose the bound holds for n. Then, we have

$$\begin{aligned}
\|\boldsymbol{u}^{n+1}\|_{\mathcal{H}} &= \left\|\Pi_{\boldsymbol{D}^{n+1}}[(1-\eta_n\lambda)\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} c^{\psi}(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)]\right\|_{\mathcal{H}} && \text{(by definition 5.12)},\\
&\leq \|(1-\eta_n\lambda)\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} c^{\psi}(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_{\mathcal{H}} && \text{(by definition of } \Pi_{\boldsymbol{D}^{n+1}})\\
&\leq (1-\eta_n\lambda)\|\boldsymbol{u}^n\|_{\mathcal{H}} + \eta_n \|\nabla_{\boldsymbol{u}} c^{\psi}(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_{\mathcal{H}} && \text{(by triangle inequality)},\\
&\leq (1-\eta_n\lambda)\|\boldsymbol{u}^n\|_{\mathcal{H}} + \eta_n \kappa \|\nabla_{\boldsymbol{u}(\boldsymbol{z})} c^{\psi}(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_2 && \text{(by Eq. (5.9))},\\
&\leq (1-\eta_n\lambda)\frac{\kappa C}{\lambda} + \eta_n \kappa \|\nabla_{\boldsymbol{u}(\boldsymbol{z})} c^{\psi}(\boldsymbol{u}^n(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_2 && \text{(by assumption for } n),\\
&\leq (1-\eta_n\lambda)\frac{\kappa C}{\lambda} + \eta_n \kappa C = \frac{\kappa C}{\lambda} && \text{(by Eq. 5.17)},
\end{aligned}$$

and therefore the result holds for all $n \in \mathbb{N}$ as desired. □

Lemma D.2.3. *Under Assumptions 1-3, for any* $\boldsymbol{u} \in \mathcal{H}$ *satisfying* $\|\boldsymbol{u}\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}$*, we have*

$$\mathbb{E}\left[\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u})\|_{\mathcal{H}_t}^2\right] \leq 4\kappa^2 C^2.$$

Proof. Fix some $t \in \{1, \ldots, T\}$. Using the fact that $\|\boldsymbol{a}+\boldsymbol{b}\|_{\mathcal{H}}^2 \leq 2(\|\boldsymbol{a}\|_{\mathcal{H}}^2 + \|\boldsymbol{b}\|_{\mathcal{H}}^2)$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{H}$, as well as Assumption 5.5.3, we obtain that for any $\boldsymbol{u} \in \mathcal{H}$, it holds

$$\begin{aligned}
\mathbb{E}\left[\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u})\|_{\mathcal{H}}^2\right] &\leq 2\mathbb{E}\left[\|\nabla_{\boldsymbol{u}} c^{\psi}(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_{\mathcal{H}}^2\right] + 2\lambda^2\|\boldsymbol{u}\|_{\mathcal{H}}^2,\\
&\leq 2\kappa^2\mathbb{E}\left[\|\nabla_{\boldsymbol{u}(\boldsymbol{z})} c^{\psi}(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_{\mathcal{H}}^2\right] + 2\lambda^2\|\boldsymbol{u}\|_{\mathcal{H}}^2 && \text{(By Eq. (5.9))},\\
&\leq 2\kappa^2\mathbb{E}\left[\|\nabla_{\boldsymbol{u}(\boldsymbol{z})} c^{\psi}(\boldsymbol{u}(\boldsymbol{z}^n), \boldsymbol{z}^n)\|_{\mathcal{H}}^2\right] + 2\lambda^2\left(\frac{\kappa C}{\lambda}\right)^2 && \text{(By Lemma D.2.2)},\\
&\leq 2\kappa^2 C^2 + 2\lambda^2\left(\frac{\kappa C}{\lambda}\right)^2 && \text{(by Eq. (5.17))},\\
&= 4\kappa^2 C^2.
\end{aligned}$$

□

Lemma D.2.4. *Under Assumption 3, given independently and identically distributed realiza-*

tions $\{\boldsymbol{z}^n\}$ *of* $\boldsymbol{z}$*, we have*

$$\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}} \leq \frac{\epsilon_n}{\eta_n},$$

where $\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)$ *was defined in Eq.* (5.15)*.*

Proof. By definition of $\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)$ we have that

$$\begin{aligned}
&\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 \\
=&\left\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \frac{\boldsymbol{u}^n - \Pi_{\boldsymbol{D}^{n+1}}[\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)]}{\eta_n}\right\|_{\mathcal{H}}^2, \\
=&\left\|\frac{1}{\eta_n}\Pi_{\boldsymbol{D}^{n+1}}[\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)] - \frac{1}{\eta_n}(\boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n))\right\|_{\mathcal{H}}^2, \\
=&\frac{1}{\eta_n^2}\|\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}\|_{\mathcal{H}}^2,
\end{aligned}$$

where $\tilde{\boldsymbol{u}}^{n+1} := \boldsymbol{u}^n - \eta_n \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)$ is the result of applying one FSGD iteration to $\boldsymbol{u}^n$. By the stopping criterion of the KOMP algorithm we know that $\left\|\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}\right\|_{\mathcal{H}} \leq \epsilon_n$, and therefore the lemma follows after taking the square root on both sides. □

Lemma D.2.5. *Under Assumption 3, for any* $\boldsymbol{u} \in \mathcal{H}$ *with* $\|\boldsymbol{u}\|_{\mathcal{H}} \leq \frac{\kappa C}{\lambda}$*, we have*

$$\begin{aligned}
&E_n^{\lambda,\psi}(\boldsymbol{u}^n) - E_n^{\lambda,\psi}(\boldsymbol{u}) && \text{(D.12)} \\
\leq&\frac{1}{2\eta_n}\left(\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - \|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2\right) + \eta_n\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 + \frac{\epsilon_n}{\eta_n}\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}} + \frac{\epsilon_n^2}{\eta_n}. && \text{(D.13)}
\end{aligned}$$

Proof. Firstly, notice that

$$\begin{aligned}
\|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2 &= \|\boldsymbol{u}^n - \eta_n \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \boldsymbol{u}\|_{\mathcal{H}}^2 \\
&= \langle \boldsymbol{u}^n - \eta_n \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \boldsymbol{u}, \boldsymbol{u}^n - \eta_n \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \boldsymbol{u}\rangle \\
&= \|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - 2\eta_n\langle \boldsymbol{u}^n - \boldsymbol{u}, \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\rangle - 2\eta_n\langle \boldsymbol{u}^n - \boldsymbol{u}, \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) \\
&\quad - \nabla_{\boldsymbol{u}} E_n^{\lambda}(\boldsymbol{u}^n)\rangle + \eta_n^2\|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2
\end{aligned} \qquad \text{(D.14)}$$

By the Cauchy Schwartz inequality and Lemma D.2.4, we have

$$\begin{aligned} |\langle \boldsymbol{u}^n - \boldsymbol{u}, \tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \nabla_{\boldsymbol{u}} E_n^{\lambda}(\boldsymbol{u}^n)\rangle| &\leq \|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}} \|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}} \quad \text{(D.15)} \\ &\leq \frac{\epsilon_n}{\eta_n}\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}. \quad \text{(D.16)} \end{aligned}$$

Substituting Eq. (D.16) in Equation (D.14) and rearranging terms we obtain

$$\langle \boldsymbol{u}^n - \boldsymbol{u}, \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\rangle \leq \frac{\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - \|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2}{2\eta_n} + \frac{\epsilon_n}{\eta_n}\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}} + \frac{\eta_n}{2}\|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2. \quad \text{(D.17)}$$

Then, we have

$$\begin{aligned} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - E_n^{\lambda,\psi}(\boldsymbol{u}) &\leq \langle \boldsymbol{u}^n - \boldsymbol{u}, \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\rangle \qquad \text{(By convexity of } E_n^{\lambda,\psi}(\boldsymbol{u}^n)\text{)}, \\ &\leq \frac{\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - \|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2}{2\eta_n} + \frac{\epsilon_n}{\eta_n}\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}} + \frac{\eta_n}{2}\|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 \end{aligned} \quad \text{(D.18)}$$

Furthermore,

$$\begin{aligned} \|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 &\leq 2\|\tilde{\nabla}_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n) - \nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 + 2\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 \\ &\leq \frac{2\epsilon_n^2}{\eta_n^2} + 2\|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2 \quad \text{(By Lemma D.2.4)}. \end{aligned} \quad \text{(D.19)}$$

The Lemma follows from applying Eq (D.19) to Eq (D.18). □

Lemma D.2.6. *Under Assumptions 1-3, for* $\epsilon_n = P_2\eta_n^2$ *we have*

$$\mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^{N^*}) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right] \leq \frac{\|\boldsymbol{u}^{\lambda,\psi} - \boldsymbol{u}^0\|_{\mathcal{H}}}{2\sum_{n=1}^N \eta_n} + \frac{\sum_{n=1}^N \eta_n^2}{\sum_{n=1}^N \eta_n}\left(\frac{2P_2\kappa C}{\lambda} + 4\kappa^2 C^2\right) + \frac{\sum_{n=1}^N \epsilon_n^2}{\sum_{n=1}^N \eta_n}.$$

Proof. Taking expectation over data and sampling on both sides of Equation (D.12) we have

$$\begin{aligned}
&\eta_n \mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^n) - E^{\lambda,\psi}(\boldsymbol{u})\right] \\
\leq & \frac{\mathbb{E}\left[\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - \|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2 + 2\epsilon_n \|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}\right]}{2} + \mathbb{E}[\eta_n^2 \|\nabla_{\boldsymbol{u}} E_n^{\lambda,\psi}(\boldsymbol{u}^n)\|_{\mathcal{H}}^2] + \epsilon_n^2, \\
\leq & \frac{\mathbb{E}\left[\|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}^2 - \|\boldsymbol{u}^{n+1} - \boldsymbol{u}\|_{\mathcal{H}}^2 + 2\epsilon_n \|\boldsymbol{u}^n - \boldsymbol{u}\|_{\mathcal{H}}\right]}{2} + 4\eta_n^2 \kappa^2 C^2 + \epsilon_n^2,
\end{aligned}$$

where the inequality follows form Lemma D.2.3. Summing over n and evaluating at $\boldsymbol{u} = \boldsymbol{u}^{\lambda,\psi}$ we obtain

$$\begin{aligned}
&\sum_{n=0}^{N} \eta_n \mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^n) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right] \\
\leq & \frac{1}{2}\|\boldsymbol{u}^{\lambda,\psi} - \boldsymbol{u}^0\|_{\mathcal{H}} + \sum_{n=0}^{N} \epsilon_n \|\boldsymbol{u}^n - \boldsymbol{u}^{\lambda,\psi}\|_{\mathcal{H}} + 4\kappa^2 C^2 \sum_{n=0}^{N} \eta_n^2 + \sum_{n=0}^{N} \epsilon_n^2, \\
\leq & \frac{1}{2}\|\boldsymbol{u}^{\lambda,\psi} - \boldsymbol{u}^0\|_{\mathcal{H}} + \sum_{n=0}^{N} 2\epsilon_n \frac{\kappa C}{\lambda} + 4\kappa^2 C^2 \sum_{n=0}^{N} \eta_n^2 + \sum_{n=0}^{N} \epsilon_n^2 \text{ (Lemma D.2.2)}, \\
= & \frac{1}{2}\|\boldsymbol{u}^{\lambda,\psi} - \boldsymbol{u}^0\|_{\mathcal{H}} + \left(\frac{2P_1 \kappa C}{\lambda} + 4\kappa^2 C^2\right) \sum_{n=0}^{N} \eta_n^2 + \sum_{n=0}^{N} \epsilon_n^2.
\end{aligned} \tag{D.20}$$

By definition of $\boldsymbol{u}_{\mathcal{S}}^*$ we then have

$$\left(\sum_{n=1}^{N} \eta_n\right) \mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}_{\mathcal{S}}^*) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right] \leq \sum_{n=1}^{N} \eta_n \mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^n) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right]. \tag{D.21}$$

Dividing by $\sum_{n=1}^{N} \eta_n$ on both sides of Eq. (D.21) and applying the inequality (D.20) we obtain

$$\mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^{N^*}) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right] \leq \frac{\|\boldsymbol{u}^{\lambda,\psi} - \boldsymbol{u}^0\|_{\mathcal{H}}}{2\sum_{n=1}^{N} \eta_n} + \frac{\sum_{n=1}^{N} \eta_n^2}{\sum_{n=1}^{N} \eta_n} \left(\frac{2P_2 \kappa C}{\lambda} + 4\kappa^2 C^2\right) + \frac{\sum_{n=1}^{N} \epsilon_n^2}{\sum_{n=1}^{N} \eta_n}.$$

as desired. □

Lemma D.2.7. *Suppose that there exists a feasible decision $\hat{\boldsymbol{u}}$ and a finite constant C_0 such* *that* $c(\hat{\boldsymbol{u}}(\boldsymbol{z}), \boldsymbol{z}) \leq C_0$ *for all* $\boldsymbol{z} \in \mathcal{Z}$*. Then,*

$$\lim_{\psi \to \infty} \mathbb{E}\left[\sum_{q=1}^{Q} \max\left(0, g_q\big(\boldsymbol{u}^{\lambda,\psi}(\boldsymbol{z})\big)\right)^2\right] = 0.$$

Proof. By definition of $\boldsymbol{u}^{\lambda,\psi}$, we know

$$\psi \mathbb{E}\left[\sum_{q=1}^{Q} \max\left(0, g_q\big(\boldsymbol{u}^{\lambda,\psi}(\boldsymbol{z})\big)\right)^2\right] \leq \mathbb{E}_{\boldsymbol{z}}\left[c(\hat{\boldsymbol{u}}(\boldsymbol{z}), \boldsymbol{z})\right] + \frac{\lambda}{2}\|\hat{\boldsymbol{u}}\|_{\mathcal{H}}^2 \leq C_0 + \frac{\lambda}{2}\|\hat{\boldsymbol{u}}\|_{\mathcal{H}}^2.$$

Therefore, for any violation tolerance $\delta > 0$ we can choose $\psi \geq \frac{2C_0 + \lambda\|\hat{\boldsymbol{u}}\|_{\mathcal{H}}^2}{2\delta}$ to ensure $\mathbb{E}\left[\sum_{q=1}^{Q} \max\left(0, g_q\big(\boldsymbol{u}^{\lambda,\psi}(\boldsymbol{z})\big)\right)^2\right] \leq \delta$ and the lemma follows. □

D.3 Main Theorems

In this section we state and proof a more general version of Theorem 5.5.4 and Corollary 5.5.5, which correspond to the case $\psi = 0$.

Theorem D.3.1 (Generalization of Theorem 1)**.** *Let* $\boldsymbol{u}_{\mathcal{S}}^* := \arg\min_{\boldsymbol{u} \in \{\boldsymbol{u}^1, \ldots, \boldsymbol{u}^N\}} E_{\mathcal{S}}^{\lambda,\psi}(\boldsymbol{u})$ *be the decisions generated by Algorithm 2 when given the set* $\mathcal{S} = \{\boldsymbol{z}^n\}_{n=1}^N$ *as input, and let* $\boldsymbol{u}^{\lambda,\psi}$ *be the true minimizer of* $E^{\lambda,\psi}(\boldsymbol{u})$ *over* $\mathcal{H}$*. If we use constant step-size* η *and constant error bounds* $\epsilon = P_2\eta^2$ *for some constant* $P_2 > 0$*, then under Assumptions 1-3, we have that*

$$\mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}_{\mathcal{S}}^*) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\right] \leq \mathcal{O}\left(\frac{\eta}{\lambda}\right).$$

Proof. Applying Lemma D.2.6 with $\eta_n = \eta$ yields

$$\mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}_{\mathcal{S}}^*) - E^{\lambda}(\boldsymbol{u}^{\lambda,\psi})\right] \leq \mathcal{O}\left(\frac{\sum_{n=1}^N \eta_n^2}{\lambda \sum_{n=1}^N \eta_n}\right) = \mathcal{O}\left(\frac{\eta}{\lambda}\right),$$

as desired. $\square$

Corollary D.3.2 (Generalization of Corollary 1)**.** *Suppose that there exists a feasible decision* $\hat{\boldsymbol{u}}$ *and finite constants* c_0, C_0 *such that* $c(\hat{\boldsymbol{u}}(\boldsymbol{z}), \boldsymbol{z}) \leq C_0$ *and* $c_0 \leq c(\boldsymbol{u}, \boldsymbol{z})$ *for all* $\boldsymbol{z} \in \mathcal{Z}$ *and for all scalar arguments* $\boldsymbol{u}$*. Let* $\boldsymbol{u}^*$ *be the true minimizer of* $E(\cdot)$ *over* $\mathcal{F}$ *and let* $\boldsymbol{u}^\psi$ *be the true minimizer of* $E^\psi(\cdot)$ *over* $\mathcal{F}$*. If we use constant step-size with* $\eta = \frac{P_1}{\sqrt{N}} < \frac{1}{\lambda}$*, and* $P_1 > 0$*, constant error bounds* $\epsilon = P_2\eta^2$ *for some constant* $P_2 > 0$*, and regularization parameter* λ *such that* $\lambda \xrightarrow[N\to\infty]{} 0$ *and* $\lambda\sqrt{N} \xrightarrow[N\to\infty]{} \infty$*, then under Assumptions 1-4 we have that*

a) $\lim_{\psi\to\infty} \lim_{N\to\infty} \mathbb{E}\left[\sum_{q=1}^{Q} \max\Big(0, g_q\big(\boldsymbol{u}^*_{\mathcal{S}}(\boldsymbol{z})\big)\Big)^2\right] = 0.$

b) $\lim_{N\to\infty} \mathbb{E}\left[\left|E^\psi(\boldsymbol{u}^*_{\mathcal{S}}) - E^\psi(\boldsymbol{u}^\psi)\right|\right] = 0$ *for all* $\psi > 0$*.*

c) $\lim_{\psi\to\infty} \lim_{N\to\infty} \mathbb{E}\left[E(\boldsymbol{u}^*_{\mathcal{S}})\right] \leq E(\boldsymbol{u}^*)$*.*

Proof. Part a) We have

$$\begin{aligned}
\lim_{N\to\infty} \psi \mathbb{E}\left[\sum_{q=1}^{Q} \max\Big(0, g_q\big(\boldsymbol{u}^*_{\mathcal{S}}(\boldsymbol{z})\big)\Big)^2\right] &\leq \lim_{N\to\infty} \mathbb{E}\left[E^{\lambda,\psi}(\boldsymbol{u}^*_{\mathcal{S}})\right] \; - c_0, \\
&\leq E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi}) - c_0 \qquad \text{(by Theorem 5.5.4)}, \\
&\leq E^{\lambda,\psi}(\hat{\boldsymbol{u}}) - c_0 \qquad \text{(by optimality of } \boldsymbol{u}^{\lambda,\psi}), \\
&= E^{\lambda}(\hat{\boldsymbol{u}}) - c_0 \qquad \text{(by feasibility of } \hat{\boldsymbol{u}}),
\end{aligned}$$

and therefore

$$0 \leq \lim_{\psi\to\infty} \lim_{N\to\infty} \mathbb{E}\left[\sum_{q=1}^{Q} \max\Big(0, g_q\big(\boldsymbol{u}^*_{\mathcal{S}}(\boldsymbol{z})\big)\Big)^2\right] \leq \lim_{\psi\to\infty} \frac{E^\lambda(\hat{\boldsymbol{u}}) - c_0}{\psi} = 0,$$

Part b) Let $\boldsymbol{u}^{\psi}_{\mathcal{H}}$ be the true minimizer of E^{ψ} over $\mathcal{H}$. Adding and subtracting terms we obtain

$$\begin{aligned} E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi}) = \big(E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\lambda,\psi}(\boldsymbol{u}^*_{\mathcal{S}})\big) + \big(E^{\lambda,\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi})\big) \\ + \Big(E^{\lambda,\psi}(\boldsymbol{u}^{\lambda,\psi}) - E^{\psi}(\boldsymbol{u}^{\psi}_{\mathcal{H}})\Big) + \Big(E^{\psi}(\boldsymbol{u}^{\psi}_{\mathcal{H}}) - E^{\psi}(\boldsymbol{u}^{\psi})\Big). \end{aligned}$$

The first term on the right hand side is negative, the second term vanishes because of Theorem 5.5.4, the third term vanishes with λ, and the fourth term is zero because we use universal kernels (Assumption 5.5.1). Since $E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi})$ is non-negative, we obtain

$$\lim_{N\to\infty} \mathbb{E}\left[\left|E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi})\right|\right] = 0 \quad \forall \psi \geq 0.$$

Part c) We have

$$\begin{aligned} &\lim_{N\to\infty} \mathbb{E}\left[E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}})\right] = \lim_{N\to\infty} \mathbb{E}\left[E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi})\right] + E^{\psi}(\boldsymbol{u}^{\psi}) \\ \implies &\lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}})\right] = \lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi})\right] + \lim_{\psi\to\infty} E^{\psi}(\boldsymbol{u}^{\psi}) \\ \implies &\lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E(\boldsymbol{u}^*_{\mathcal{S}})\right] = \lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E^{\psi}(\boldsymbol{u}^*_{\mathcal{S}}) - E^{\psi}(\boldsymbol{u}^{\psi})\right] + \lim_{\psi\to\infty} E^{\psi}(\boldsymbol{u}^{\psi}) \quad \text{(By part a))} \\ \implies &\lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E(\boldsymbol{u}^*_{\mathcal{S}})\right] \leq \lim_{\psi\to\infty} E^{\psi}(\boldsymbol{u}^{\psi}) \quad \text{(By part b))} \\ \implies &\lim_{\psi\to\infty}\lim_{N\to\infty} \mathbb{E}\left[E(\boldsymbol{u}^*_{\mathcal{S}})\right] \leq E(\boldsymbol{u}^*), \quad \text{(By optimality of } \boldsymbol{u}^{\psi}) \end{aligned}$$

as desired. □

D.4 Finding Lower Bounds

We emphasize that we only need to find lower bounds for the case in which the dimension of the data and the dimension of the controls is equal to 1; since the experiments run for multidimensional cases were designed to have the same objective value as the one dimensional case. The exact problem we want to lower bound is then

$$\begin{aligned}
\min_{\boldsymbol{u}_{1:T}} \quad & \mathbb{E}_{\boldsymbol{w}|\boldsymbol{x}}\left[\sum_{t=1}^{T} 2\left[s_t\right]^+ + \left[-s_t\right]^+ \,\middle|\, \boldsymbol{x}=\boldsymbol{x}_0\right] && && \text{(D.22)}\\
\text{s.t.} \quad & s_t = s_{t-1} + u_t - w_t && && \text{(D.23)}\\
& u_t \geq 0 && \forall t \in [T], && \text{(D.24)}\\
& u_t \leq 150 && \forall t \in [T], && \text{(D.25)}\\
& u_t + u_{t+1} \leq 200 && \forall t \in [T-1]. && \text{(D.26)}
\end{aligned}$$

The demands w_t were generated as a linear function of the covariates with some added noise; specifically, $w_t = \alpha_t x + \epsilon_t$, where ϵ_t was sampled from a standard distribution and the constants α_t were selected to be close to 50 in order for the triangular constraints to be relevant. In fact, for the specified parameters, we found that the constraints (D.25) and (D.24) are quite lose and we can find a good lower bound for the the optimal objective value by removing these constraints. The problem to solve can then be simplified as

$$\begin{aligned}
&\min_{\boldsymbol{u}_{1:T}} \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{t=1}^{T} 2\left[\sum_{i=1}^{t} u_i(x_0, \boldsymbol{\epsilon}_{1:i-1}) - \alpha_i x_0 - \epsilon_i\right]^+ + \left[-\sum_{i=1}^{t} u_i(x_0, \boldsymbol{\epsilon}_{1:i-1}) - \alpha_i x_0 - \epsilon_i\right]^+\right]\\
&\text{s.t.} \quad u_t(x_0, \boldsymbol{\epsilon}_{1:t-1}) + u_{t+1}(x_0, \boldsymbol{\epsilon}_{1:t}) \leq 200 \quad \forall t \in [T-1]. \qquad \text{(D.27)}
\end{aligned}$$

To solve the problem above, we use the fact that if ϵ has a normal distribution then the function $f(a) = \mathbb{E}_\epsilon\left[2[a-\epsilon]^+ + [\epsilon - a]^+\right]$ is strictly convex and

$$\begin{aligned}
a_0 &:= \arg\min_a \; \mathbb{E}_\epsilon\left[2[a-\epsilon]^+ + [\epsilon - a]^+\right] && \text{(D.28)}\\
&= \min_a \int_{-\infty}^{a} 2(a-x)\frac{e^{-x^2/2}}{\sqrt{2\pi}}\,dx + \int_{a}^{\infty}(x-a)\frac{e^{-x^2/2}}{\sqrt{2\pi}}\,dx\\
&= \min_a \frac{3e^{-a^2/2}}{\sqrt{2\pi}} + \frac{3}{2}a\left(1 + \operatorname{erf}\left(\frac{a}{\sqrt{2}}\right)\right) - a\\
&= -\sqrt{2}\operatorname{erf}^{-1}\left(\frac{1}{3}\right). && \text{(D.29)}
\end{aligned}$$

We will show how to exactly solve problem (D.27) in the case $T = 2$ (the analysis is similar for cases $T = 3, 4, 5$). Suppose then that $T = 2$. For a fix value of $u_1(x_0)$, the optimization problem (D.27) over $u_2(x_0, \epsilon_1)$ becomes

$$\begin{aligned} &\min_{u_2} \mathbb{E}_{\epsilon_2} \left[2 \left[\sum_{i=1}^{2} u_i(x_0, \boldsymbol{\epsilon}_{1:i-1}) - \alpha_i x_0 - \epsilon_i \right]^+ + \left[- \sum_{i=1}^{2} u_i(x_0, \boldsymbol{\epsilon}_{1:i-1}) - \alpha_i x_0 - \epsilon_i \right]^+ \right] \\ &\text{s.t.} \quad u_2(x_0, \boldsymbol{\epsilon}_1) \le 200 - u_1(x_0). \end{aligned}$$

Applying the result from Eq. (D.29) we obtain

$$u_2^*(x_0, \epsilon_1) = \min\{(\alpha_1 + \alpha_2)x_0 + \epsilon_1 - u_1(x_0) + a_0, 200 - u_1(x_0)\},$$

which implies that the term $\sum_{i=1}^{2} u_i(x_0, \boldsymbol{\epsilon}_{1:i-1}) - \alpha_i x_0 - \epsilon_i$ evaluated at $u_2^*(x_0, \epsilon_1)$ is equal to $\min\{a_0 - \epsilon_2, 200 - (\alpha_1 + \alpha_2)x_0 - \epsilon_1 - \epsilon_2)$, which is independent of $u_1(x_0)$. We can then find the optimal u_1 by solving

$$\begin{aligned} &\min_{u_1} \quad \mathbb{E}_{\epsilon_1} \left[2 \left[u_1(x_0) - \alpha_1 x_0 - \epsilon_1 \right]^+ + \left[\alpha_1 x_0 + \epsilon_1 - u_1(x_0) \right]^+ \right] \\ &\text{s.t.} \quad u_1(x_0,) \le 200, \end{aligned}$$

which again, using Eq. (D.29) yields $u_1^*(x_0) = \min\{\alpha_1 x_0 + a_0, 200\}$.

References

Abadi M, Agarwal A, et al (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/, software available from tensorflow.org.

Abadie A (2005) Semiparametric difference-in-differences estimators. *The Review of Economic Studies* 72(1):1–19.

Aghasi A, Abdi A, Romberg J (2020) Fast convex pruning of deep neural networks. *SIAM Journal on Mathematics of Data Science* 2(1):158–188.

Alsentzer E, Murphy JR, Boag W, Weng WH, Jin D, Naumann T, McDermott MBA (2019) Publicly available clinical bert embeddings. URL http://dx.doi.org/10.48550/ARXIV.1904.03323.

Amram M, Dunn J, Zhuo YD (2022) Optimal policy trees. *Machine Learning* 111:2741–2768.

Anderson R, Huchette J, Ma W, Tjandraatmadja C, Vielma JP (2020) Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming* 1–37.

Anthony M, Bartlett PL (2009) *Neural network learning: Theoretical foundations* (cambridge university press).

Arik SÖ, Pfister T (2021) TabNet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(8):6679–6687.

Athalye A, Carlini N, Wagner D (2018) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples.

Baker AC, Larcker DF, Wang CC (2022) How much should we trust staggered difference-in-differences estimates? *Journal of Financial Economics* 144(2):370–395.

Balunovic M, Vechev M (2019) Adversarial training and provable defenses: Bridging the gap. *International Conference on Learning Representations.*

Ban GY, Gallien J, Mersereau AJ (2019) Dynamic procurement of new products with covariate information: The residual tree method. *Manufacturing & Service Operations Management* 21(4):798–815.

Bellec G, Kappel D, Maass W, Legenstein RA (2017) Deep rewiring: Training very sparse deep networks. *CoRR* abs/1711.05136, URL http://arxiv.org/abs/1711.05136.

Beltagy I, Peters ME, Cohan A (2020) Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* .

Ben-Tal A, Den Hertog D, Vial JP (2015) Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical programming* 149(1):265–299.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust optimization* (Princeton university press).

Bertrand M, Duflo E, Mullainathan S (2004) How much should we trust differences-in-differences estimates? *The Quarterly Journal of Economics* 119(1):249–275.

Bertsimas D, Boix X, Carballo KV, Hertog Dd (2021a) Robust upper bounds for adversarial training. *arXiv preprint arXiv:2112.09279* .

Bertsimas D, Brown DB, Caramanis C (2011) Theory and applications of robust optimization. *SIAM Review* 53(3):464–501.

Bertsimas D, Carballo KV (2023) Multistage stochastic optimization via kernels. *arXiv preprint arXiv:2303.06515* .

Bertsimas D, den Hertog D (2022) *Robust and Adaptive Optimization* (Dynamic Ideas LLC).

Bertsimas D, Dunn J (2017) Optimal classification trees. *Machine Learning* 106:1039–1082.

Bertsimas D, Dunn J, Paskov I (2022a) Stable classification. *Journal of Machine Learning Research* 23(296):1–53.

Bertsimas D, Dunn J, Pawlowski C, Zhuo YD (2019) Robust classification. *INFORMS Journal on Optimization* 1(1):2–34.

Bertsimas D, Hertog Dd, Pauphilet J, Zhen J (2023) Robust convex optimization: A new perspective that unifies and extends. *Mathematical Programming* 200(2):877–918.

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Science* 66(3):1025–1044.

Bertsimas D, Koduri N (2022) Data-driven optimization: A reproducing kernel hilbert space approach. *Operations Research* 70(1):454–471.

Bertsimas D, McCord C (2019) From predictions to prescriptions in multistage optimization problems. *Mathematical Programming, under review* ArXiv preprint arXiv:1904.11637.

Bertsimas D, McCord C, Sturt B (2022b) Dynamic optimization with side information. *European Journal of Operational Research, to appear* ArXiv preprint arXiv:1907.07307v2.

Bertsimas D, Paskov I (2020) Stable regression: On the power of optimization over randomization in training regression problems. *Journal of Machine Learning Research* 21(230):1–25.

Bertsimas D, Pauphilet J, Parys BV (2020) Sparse Regression: Scalable Algorithms and Empirical Performance. *Statistical Science* 35(4):555 – 578, URL http://dx.doi.org/10.1214/19-STS701.

Bertsimas D, Pauphilet J, Stevens J, Tandon M (2021b) Predicting inpatient flow at a major hospital using interpretable analytics. *Manufacturing & Service Operations Management* 24(6):2809–2824.

Bertsimas D, Pauphilet J, Van Parys B (2021c) Sparse classification: a scalable discrete optimization perspective. *Machine Learning* 110(11):3177–3209.

Bertsimas D, Shtern S, Sturt B (2022c) A data-driven approach to multistage stochastic linear optimization. *Management Science, to appear* Preprint at https://dbertsim.mit.edu/pdfs/papers/2018-sturt-data-driven-two-stage-approach.pdf.

Bertsimas D, Villalobos Carballo K, Boussioux L, Li ML, Paskov A, Paskov I (2024) Holistic deep learning. *Machine Learning* 113(1):159–183.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer).

Bunel R, Turkaslan I, Torr PH, Kohli P, Kumar MP (2017) A unified view of piecewise linear neural network verification. *arXiv preprint arXiv:1711.00455* .

Callaway B, Sant'Anna PH (2021) Difference-in-differences with multiple time periods. *Journal of Econometrics* 225(2):200–230.

Cameron AC, Miller DL (2015) A practitioner's guide to cluster-robust inference. *Journal of Human Resources* 50(2):317–372.

Carballo KV, Na L, Ma Y, Boussioux L, Zeng C, Soenksen LR, Bertsimas D (2022) Tabtext: A flexible and contextual approach to tabular data representation. *arXiv preprint arXiv:2206.10381* .

Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. *2017 IEEE symposium on security and privacy (sp)*, 39–57.

Changpinyo S, Sandler M, Zhmoginov A (2017) The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257* .

Chassein A, Goerigk M (2019) On the complexity of robust geometric programming with polyhedral uncertainty. *Operations Research Letters* 47(1):21–24.

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 (Association for Computing Machinery).

Cohen J, Rosenfeld E, Kolter Z (2019) Certified adversarial robustness via randomized smoothing. *International Conference on Machine Learning*, 1310–1320 (PMLR).

Dathathri S, Dvijotham K, Kurakin A, Raghunathan A, Uesato J, Bunel R, Shankar S, Steinhardt J, Goodfellow I, Liang P, Kohli P (2020) Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *arXiv preprint arXiv:2010.11645* .

Deng L (2012a) The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29(6):141–142.

Deng L (2012b) The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29(6):141–142.

Dua D, Graff C (2017) UCI machine learning repository. URL http://archive.ics.uci.edu/ml.

Dvijotham K, Gowal S, Stanforth R, Arandjelovic R, O'Donoghue B, Uesato J, Kohli P (2018a) Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265* .

Dvijotham K, Stanforth R, Gowal S, Mann TA, Kohli P (2018b) A dual approach to scalable verification of deep networks. *UAI*, volume 1, 3.

Ek A, Bernardy JP, Chatzikyriakidis S (2020) How does punctuation affect neural models in natural language inference. URL https://aclanthology.org/2020.pam-1.15.

Engel Y, Mannor S, Meir R (2004) The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing* 52(8):2275–2285.

Gale T, Elsen E, Hooker S (2019) The state of sparsity in deep neural networks. *CoRR* abs/1902.09574, URL http://arxiv.org/abs/1902.09574.

Gehr T, Mirman M, Drachsler-Cohen D, Tsankov P, Chaudhuri S, Vechev M (2018) Ai2: Safety and robustness certification of neural networks with abstract interpretation. *2018 IEEE Symposium on Security and Privacy (SP)*, 3–18 (IEEE).

Geneviève LD, Martani A, Mallet MC, Wangmo T, Elger BS (2019) Factors influencing harmonized health data collection, sharing and linkage in denmark and switzerland: A systematic review. *PLOS ONE* 14(12):e0226015, URL http://dx.doi.org/10.1371/journal.pone.0226015.

Glorot X, Bengio Y (2010a) Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256 (JMLR Workshop and Conference Proceedings).

Glorot X, Bengio Y (2010b) Understanding the difficulty of training deep feedforward neural networks. Teh YW, Titterington M, eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, 249–256 (Chia Laguna Resort, Sardinia, Italy: PMLR), URL https://proceedings.mlr.press/v9/glorot10a.html.

Goldwasser S, Kalai AT, Kalai Y, Montasser O (2020) Beyond perturbations: Learning guarantees with arbitrary adversarial test examples. *Advances in Neural Information Processing Systems* 33:15859–15870.

Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples.

Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples.

Goodman-Bacon A (2021) Difference-in-differences with variation in treatment timing. *Journal of Econometrics* 225(2):254–277, ISSN 0304-4076, Themed Issue: Treatment Effect 1.

Gorishniy Y, Rubachev I, Babenko A (2022) On embeddings for numerical features in tabular deep learning. URL http://dx.doi.org/10.48550/ARXIV.2203.05556.

Gowal S, Dvijotham KD, Stanforth R, Bunel R, Qin C, Uesato J, Arandjelovic R, Mann T, Kohli

P (2019) Scalable verified training for provably robust image classification. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4842–4851.

Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28.

Hanasusanto GA, Kuhn D (2013) Robust data-driven dynamic programming. *Advances in Neural Information Processing Systems* 827–835, ISSN 10495258, URL http://papers.nips.cc/paper/5123-robust-data-driven-dynamic-programming.

Harari A, Katz G (2022) Few-shot tabular data enrichment using fine-tuned transformer architectures.

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825):357–362, URL http://dx.doi.org/10.1038/s41586-020-2649-2.

Hastie T, Tibshirani R, Friedman J (2001) *The Elements of Statistical Learning*. Springer Series in Statistics (New York, NY, USA: Springer New York Inc.).

Hegselmann S, Buendia A, Lang H, Agrawal M, Jiang X, Sontag D (2023) Tabllm: Few-shot classification of tabular data with large language models. Ruiz F, Dy J, van de Meent JW, eds., *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, 5549–5581 (PMLR), URL https://proceedings.mlr.press/v206/hegselmann23a.html.

Hein M, Andriushchenko M (2017) Formal guarantees on the robustness of a classifier against adversarial manipulation. *CoRR* abs/1705.08475, URL http://arxiv.org/abs/1705.08475.

Herzig J, Nowak PK, Müller T, Piccinno F, Eisenschlos J (2020) TaPas: Weakly supervised table parsing via pre-training. URL http://dx.doi.org/10.18653/v1/2020.acl-main.398.

Hoefler T, Alistarh D, Ben-Nun T, Dryden N, Peste A (2021) Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research* 22(1):10882–11005.

Honeine P (2011) Online kernel principal component analysis: A reduced-order model. *IEEE transactions on pattern analysis and machine intelligence* 34(9):1814–1826.

Huang R, Xu B, Schuurmans D, Szepesvari C (2016) Learning with a strong adversary.

Ilyas A, Jalal A, Asteri E, Daskalakis C, Dimakis AG (2017) The robust manifold defense: Adversarial training using generative models. *CoRR* abs/1712.09196, URL http://arxiv.org/abs/1712.09196.

Ilyas A, Santurkar S, Tsipras D, Engstrom L, Tran B, Madry A (2019) Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175* .

Janowsky SA (1989) Pruning versus clipping in neural networks. *Phys. Rev. A* 39:6600–6603, URL http://dx.doi.org/10.1103/PhysRevA.39.6600.

Johnson AE, Pollard TJ, Shen L, Lehman LwH, Feng M, Ghassemi M, Moody B, Szolovits P, Anthony Celi L, Mark RG (2016) Mimic-iii, a freely accessible critical care database. *Scientific data* 3(1):1–9.

Kabilan VM, Morris B, Nguyen A (2018) Vectordefense: Vectorization as a defense to adversarial examples. *CoRR* abs/1804.08529, URL http://arxiv.org/abs/1804.08529.

Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ (2017) Reluplex: An efficient smt solver for verifying deep neural networks. *International Conference on Computer Aided Verification*, 97–117 (Springer).

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems* 30.

Kivinen J, Smola AJ, Williamson RC (2004) Online learning with kernels. *IEEE transactions on signal processing* 52(8):2165–2176.

Koppel A, Warnell G, Stump E, Ribeiro A (2016) Parsimonious online learning with kernels via sparse projections in function space.

Krizhevsky A, Hinton G, et al. (2009a) Learning multiple layers of features from tiny images .

Krizhevsky A, Hinton G, et al. (2009b) Learning multiple layers of features from tiny images. Technical report.

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 1097–1105, NIPS'12 (Red Hook, NY, USA: Curran Associates Inc.).

Krogh A, Vedelsby J (1994) Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems* 7.

Kurakin A, Goodfellow I, Bengio S, et al. (2016) Adversarial examples in the physical world.

Lamb A, Binas J, Goyal A, Serdyuk D, Subramanian S, Mitliagkas I, Bengio Y (2018) Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations. *arXiv preprint arXiv:1804.02485* .

LeCun Y, Denker J, Solla S (1989) Optimal brain damage. *Advances in neural information processing systems* 2.

Lecuyer M, Atlidakis V, Geambasu R, Hsu D, Jana S (2019) Certified robustness to adversarial examples with differential privacy. *2019 IEEE Symposium on Security and Privacy (SP)*, 656–672 (IEEE).

Lee J, Yoon W, Kim S, Kim D, Kim S, So CH, Kang J (2019) BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36(4):1234–1240, ISSN 1367-4803, URL http://dx.doi.org/10.1093/bioinformatics/btz682.

Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. *CoRR* abs/1608.08710, URL http://arxiv.org/abs/1608.08710.

Li Y, Wehbe RM, Ahmad FS, Wang H, Luo Y (2022) Clinical-longformer and clinical-bigbird: Transformers for long clinical sequences. *arXiv preprint arXiv:2201.11838* .

Liang KY, Zeger SL (1986) Longitudinal data analysis using generalized linear models. *Biometrika* 73(1):13–22.

Liu EZ, Haghgoo B, Chen AS, Raghunathan A, Koh PW, Sagawa S, Liang P, Finn C (2021) Just train twice: Improving group robustness without training group information. *International Conference on Machine Learning*, 6781–6792 (PMLR).

Lorena AC, Garcia LP, Lehmann J, Souto MC, Ho TK (2019) How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)* 52(5):1–34.

Louizos C, Welling M, Kingma DP (2017) Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312* .

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30.

Luo R, Sun L, Xia Y, Qin T, Zhang S, Poon H, Liu TY (2022) BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics* 23(6), URL http://dx.doi.org/10.1093/bib/bbac409.

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks.

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2019) Towards deep learning models resistant to adversarial attacks.

May RJ, Maier HR, Dandy GC (2010) Data splitting for artificial neural networks using som-based stratified sampling. *Neural Networks* 23(2):283–294.

Micchelli CA, Xu Y, Zhang H (2006) Universal kernels. *Journal of Machine Learning Research* 7(12).

Mirman M, Gehr T, Vechev M (2018) Differentiable abstract interpretation for provably robust neural networks. *International Conference on Machine Learning*, 3578–3586 (PMLR).

Miyajiwala A, Ladkat A, Jagadale S, Joshi R (2022) On sensitivity of deep learning based text classification algorithms to practical input perturbations.

Mocanu DC, Mocanu E, Stone P, Nguyen PH, Gibescu M, Liotta A (2017) Evolutionary training of sparse artificial neural networks: A network science perspective. *CoRR* abs/1707.04780, URL http://arxiv.org/abs/1707.04780.

Mostafa H, Wang X (2019) Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. *International Conference on Machine Learning*, 4646–4655 (PMLR).

Na L, Villalobos Carballo K, Pauphilet J, Haddad-Sisakht A, Kombert D, Boisjoli-Langlois M, Castiglione A, Khalifa M, Hebbal P, Stein B, Bertsimas D (2023) Patient outcome predictions improve operations at a large hospital network. *arXiv preprint arXiv:2305.15629* .

Nan Y, Ser JD, Walsh S, Schönlieb C, Roberts M, Selby I, Howard K, Owen J, Neville J, Guiot J, Ernst B, Pastor A, Alberich-Bayarri A, Menzel MI, Walsh S, Vos W, Flerin N, Charbonnier JP, van Rikxoort E, Chatterjee A, Woodruff H, Lambin P, Cerdá-Alberich L, Martí-Bonmatí L, Herrera F, Yang G (2022) Data harmonisation for information fusion in digital healthcare: A state-of-the-art systematic review, meta-analysis and future research directions. *Information Fusion* 82:99–122, URL http://dx.doi.org/10.1016/j.inffus.2022.01.001.

Narang S, Elsen E, Diamos G, Sengupta S (2017) Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119* .

Norkin V, Keyzer M (2009) On stochastic optimization and statistical learning in reproducing kernel hilbert spaces by support vector machines (svm). *Informatica* 20(2):273–292.

Padhi I, Schiff Y, Melnyk I, Rigotti M, Mroueh Y, Dognin P, Ross J, Nair R, Altman E (2021) Tabular transformers for modeling multivariate time series. URL http://dx.doi.org/10.1109/ICASSP39728.2021.9414142.

Pati YC, Rezaiifar R, Krishnaprasad PS (1993) Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar conference on signals, systems and computers*, 40–44 (IEEE).

Pflug GC, Pichler A (2016) From empirical observations to tree models for stochastic optimization: convergence properties. *SIAM Journal on Optimization* 26(3):1715–1740.

Prakash A, Moran N, Garber S, DiLillo A, Storer JA (2018) Deflecting adversarial attacks with pixel deflection. *CoRR* abs/1801.08926, URL http://arxiv.org/abs/1801.08926.

Raghunathan A, Steinhardt J, Liang P (2018a) Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344* .

Raghunathan A, Steinhardt J, Liang P (2018b) Semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:1811.01057* .

Rauber J, Brendel W, Bethge M (2017) Foolbox: A python toolbox to benchmark the robustness of machine learning models. *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, URL http://arxiv.org/abs/1707.04131.

Rauber J, Zimmermann R, Bethge M, Brendel W (2020) Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software* 5(53):2607, URL http://dx.doi.org/10.21105/joss.02607.

Rockafellar RT (1970) *Convex analysis.* Princeton Mathematical Series (Princeton, N. J.: Princeton University Press).

Roos E, den Hertog D, Ben-Tal A, De Ruiter F, Zhen J (2020) Tractable approximation of hard uncertain optimization problems. *Available on Optimization-Online* .

Ross AS, Doshi-Velez F (2017) Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients.

Sagawa S, Koh PW, Hashimoto TB, Liang P (2019) Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR* abs/1911.08731, URL http://arxiv.org/abs/1911.08731.

Savarese P, Silva H, Maire M (2020) Winning the lottery with continuous sparsification. *Advances in Neural Information Processing Systems* 33:11380–11390.

Schölkopf B, Smola AJ, Bach F, et al. (2002) *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press).

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

Shapiro A, Dentcheva D, Ruszczyński A (2014) *Lectures on stochastic programming: modeling and theory* (SIAM).

Shawe-Taylor J, Cristianini N, et al. (2004) *Kernel methods for pattern analysis* (Cambridge university press).

Singh G, Gehr T, Mirman M, Püschel M, Vechev MT (2018) Fast and effective robustness certification. *NeurIPS* 1(4):6.

Sinha A, Namkoong H, Volpi R, Duchi J (2017) Certifying some distributional robustness with principled adversarial training. *International Conference on Learning Representations* .

Soentpiet R, et al. (1999) *Advances in kernel methods: support vector learning* (MIT press).

Somepalli G, Goldblum M, Schwarzschild A, Bruss CB, Goldstein T (2021) Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. URL http://dx.doi.org/10.48550/ARXIV.2106.01342.

Staib M, Jegelka S (2019) Distributionally robust optimization and generalization in kernel methods. *Advances in Neural Information Processing Systems* 32.

Sweeney E (2017) Experts say ibm watson's flaws are rooted in data collection and interoperability. URL https://www.fiercehealthcare.com/analytics/ibm-watson-s-flaws-trace-back-to-data-collection-interoperability.

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. *International Conference on Learning Representations*, URL http://arxiv.org/abs/1312.6199.

Thompson NC, Greenewald KH, Lee K, Manso GF (2020) The computational limits of deep learning. *CoRR* abs/2007.05558, URL https://arxiv.org/abs/2007.05558.

Tjeng V, Xiao K, Tedrake R (2019) Evaluating robustness of neural networks with mixed integer programming.

Van Rossum G, Drake FL (2009a) *Python 3 Reference Manual* (Scotts Valley, CA: CreateSpace), ISBN 1441412697.

Van Rossum G, Drake FL (2009b) *Python 3 Reference Manual* (Scotts Valley, CA: CreateSpace), ISBN 1441412697.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Advances in neural information processing systems* 30.

Vincent P, Bengio Y (2002) Kernel matching pursuit. *Machine learning* 48(1):165–187.

Wahba G (1990) *Spline models for observational data* (SIAM).

Weng L, Zhang H, Chen H, Song Z, Hsieh CJ, Daniel L, Boning D, Dhillon I (2018) Towards fast computation of certified robustness for relu networks. *International Conference on Machine Learning*, 5276–5285 (PMLR).

Wheeden RL (2015) *Measure and integral: an introduction to real analysis*, volume 308 (CRC press).

Wong E, Kolter Z (2018) Provable defenses against adversarial examples via the convex outer adversarial polytope. *International Conference on Machine Learning*, 5286–5295 (PMLR).

Wong E, Rice L, Kolter JZ (2020) Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994* .

Wong E, Schmidt FR, Metzen JH, Kolter JZ (2018) Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514* .

Xiao H, Rasul K, Vollgraf R (2017a) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xiao H, Rasul K, Vollgraf R (2017b) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xie C, Wang J, Zhang Z, Ren Z, Yuille A (2017) Mitigating adversarial effects through randomization.

Xu Y, Goodacre R (2018) On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of analysis and testing* 2(3):249–262.

Yan Z, Guo Y, Zhang C (2018) Deep defense: Training dnns with improved adversarial robustness. *arXiv preprint arXiv:1803.00404* .

Yin P, Neubig G, Yih Wt, Riedel S (2020) TaBERT: Pretraining for joint understanding of textual and tabular data. URL http://dx.doi.org/10.18653/v1/2020.acl-main.745.

Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 694–699.

Zhang H, Chen H, Xiao C, Gowal S, Stanforth R, Li B, Boning D, Hsieh CJ (2019) Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316* .

Zhang H, Weng TW, Chen PY, Hsieh CJ, Daniel L (2018) Efficient neural network robustness certification with general activation functions. *arXiv preprint arXiv:1811.00866* .

Zhang L, Yi J, Jin R, Lin M, He X (2013) Online kernel learning with a near optimal sparsity bound. *International Conference on Machine Learning*, 621–629.

Zhen J, de Ruiter F, Den Hertog D (2017) Robust optimization for models with uncertain soc and sdp constraints. *Optimization Online* .

Zhou DX (2002) The covering number in learning theory. *Journal of Complexity* 18(3):739–767, ISSN 0885-064X, URL http://dx.doi.org/https://doi.org/10.1006/jcom.2002.0635.

Zhuang Z, Tan M, Zhuang B, Liu J, Guo Y, Wu Q, Huang J, Zhu J (2018) Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems* 31.