DEPENDABLE NEURAL NETWORKS FOR SAFETY CRITICAL TASKS

by

Molly O'Brien

A dissertation submitted to Johns Hopkins University
in conformity with the requirements for the degree of
Doctor of Philosophy

Baltimore, Maryland
February 2022

© 2022 Molly O'Brien
All rights reserved

Abstract

Neural Networks (NNs) have demonstrated impressive performance improvement over the last decade in safety critical tasks, e.g., perception for autonomous vehicles, medical image analysis, etc., but, NNs performing safety critical tasks poses a risk for harm, as NN performance often degrades when the operating domain changes. Previous work has proposed new training paradigms to improve NN generalization to new operating domains but fails to predict what the NN performance in the new operating domain will be. In addition, performance metrics in Machine Learning (ML) focus on the average probability of success but do not differentiate failures that cause harm from those that do not.

In this thesis, we leverage structure in NN behavior based on the environment context and the NN embedding to predict NN performance for safety critical tasks in unconstrained environments. We denote factors relating to the environment context as context features. First, we define performance metrics that capture both the probability of task success and the probability of causing harm. We then address the task of predicting NN performance in a novel operating domain as Network Generalization Prediction (NGP), and we derive a NGP algorithm from a finite test set using known context features.

Second, we extend our NGP algorithm to identify which context features impact NN performance from a set of observed context features, where it is not known a priori what features are important. Third, we map structure in the NN embedding space that is informative about NN performance and derive a NGP algorithm based on how unlabeled novel operating domain images map into the embedding space. Fourth, we investigate safety functions for NNs. Safety functions are standard practice in functional safety where an external function is added to a process, e.g., a chemical reaction, to improve the overall safety. We introduce the concept of safety functions for NNs and show that external logic around NNs can improve the safety for a robot control task and image classification tasks. We demonstrate these methods on pertinent real-world tasks using state-of-the-art NNs, e.g., DenseNet for melanoma classification and FasterRCNN for pedestrian detection.

Thesis Committee

Primary Readers

Gregory Hager
Mandell Bellmore Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Julia Bukowski
Associate Professor, Retired
Department of Electrical and Computer Engineering
Villanova University

Secondary Reader

Austin Reiter
Assistant Professor
Department of Computer Science
Johns Hopkins Whiting School of Engineering

Acknowledgments

Completing a PhD program has been a challenging, exciting, and humbling process. I could never have made it through this program without support from advisors, lab mates, colleagues, family, and friends. I am so grateful to everyone who helped me through this adventure over the last five and a half years.

First, I would like to thank my advisor Dr. Gregory D. Hager. Over the years, Greg has pushed me to ask critical questions and see broader impacts of my work beyond the fixed problem in front of me. Greg is a flexible thinker and has given me the space to explore different problems, different methods, and different domains. I am so grateful for Greg's insightful feedback on technical writing; Greg has shown me how to write in a way that is more mathematically precise, more clear, and more eloquent. This thesis would not have been possible without Greg's invaluable input.

I would also like to thank the other members of my thesis committee, Dr. Julia Bukowski and Dr. Austin Reiter. Julia supervised my first technical research project, on pressure relief values when I was in high school, and her input has been essential to this thesis. Julia's expertise in functional safety and systems engineering gave me a different perspective through which to

see my research. Over the last two years, she has helped me to focus my research and, as the work progressed, she has guided me to extensions that were both significant and practical. Julia's assistance in technical writing was crucial to helping me write more clearly and her attention to detail has vastly improved the grammar in my writing, e.g., I now know how to punctuate "e.g.". In addition to this, I am so grateful for Julia's mentorship that has helped me navigate through this PhD program. I would also like to thank Austin who guided me in computer vision applications in the beginning of my PhD. Austin's fresh perspective throughout the program has helped me think about cross-disciplinary applications of my work.

In addition to the three advisors on my thesis committee, I would also like to thank other mentors who have advised me along the way. I would like to give a special thanks to Dr. Russ Taylor, with whom I had the pleasure of working closely in my first year in the PhD program. Russ introduced me to many different surgical applications and different surgeons. Working with Russ, I got to see the translation of surgical robots from research to industry. Russ's strong mentorship also guided me in technical presentations, communication, and project management. I would like to thank Dr. Masaru Ishii for his clinical guidance in the Septoplasty project; he is an innovative thinker and it was a pleasure to work with him. Thank you to Dr. S. Swaroop Vedula for his clinical guidance, his insightful questions, and his passion and curiosity for research. I'd like to thank Dr. Mathias Unberath for his cogent guidance and creative ideas; I am grateful I have had the opportunity to work with him in the last months of my PhD. From exida, I am so grateful for the

guidance of Dr. William Goble, Rainer Faller, and Mike Medoff. Thank you, Bill, Rainer, and Mike, for your technical explanations on functional safety and your encouragement. The work in this thesis began in an internship with exida and this thesis would not have been possible without your support. From the U.S. Food and Drug Administration, thank you to Dr. Aria Pezeshk for your unique perspective on regulating ML in healthcare. It was so interesting to learn from Aria about the safety implications of ML in clinical applications.

I am so grateful for the financial assistance I have received over the last five years. This funding includes the Johns Hopkins Computer Science Department First Year Fellowship, NIH grant R01-DE025265, NIH grant R21-DE022656, an Intuitive Surgical grant, a fellowship from exida, and a Critical Path grant from the U.S. Food and Drug Administration, and by an appointment to the Research Participation Program at the Center for Devices and Radiological Health administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and the U.S. Food and Drug Administration.

Thank you to all of the lab mates, colleagues, and friends I have met at Johns Hopkins. Your insightful feedback and friendship has been such a wonderful part of this experience. Thank you to Dr. Ayushi Sinha, Dr. Katie Henry, and Dr. Narges Ahmidi, for being such wonderful friends and mentors. Thank you to Dr. Jie Ying Wu for your friendship, your technical input, and your help with CIS homework in our first semester. From the Malone Center for Engineering in Healthcare, I'd also like to thank Dr. Anand Malpani, Dr. Jonathan Jones, Dr. Princy Parsana, Benj Shapiro, Dr. Andrew Hundt, Dr. Tae

Soo Kim, Dr. Rob DiPietro, Yotam Barnoy, Jin Bai, Mitchell Pavlak, and so many other people that I can't list them all here.

Thank you to all of the faculty for the interesting and challenging courses I had the opportunity to learn from. Thank you so much to Tracy Marshall for your help throughout this process and for making every day in Malone Hall brighter. Thank you to Zack Burwell and Kim Franklin for helping me through the different stages of the PhD program. Outside of Hopkins, I am so grateful for the feedback from Azade Farshad on Generative Adversarial Networks and André Roßbach on the intersection of ML and safety. I'd also like to thank Cheryl Knobloch, the Director of Women in Engineering at Penn State University. It was an alumni panel at the Women in Engineering Program Orientation (WEPO), before freshman year had even begun, that first piqued my interest in a PhD. Throughout my undergraduate career, Cheryl mentored and supported me, and her guidance was crucial during my application to PhD programs.

Finally, I'd like to thank my family for being so supportive in these last five years and throughout my entire life. I'd like to thank my dad, Chris O'Brien for originally encouraging me to become an engineer, for supporting me all through college, for understanding me when I cannot form sentences, and for his technical support working with exida. I would not have had the confidence to embark on this PhD journey without your love and support. Thank you to my mom, Sheri O'Brien, for your love, support, encouragement, and for your patience talking with me on the phone for hours all the years I was away at school. Thank you to my siblings, Melissa, Patrick, and Sarah

O'Brien, for your love and support. Thank you to my best friends, Sam Jenkins and Julia Baldassarre, for your encouragement and for listening patiently as I try to explain my research. Thank you to all my extending family for being with me through these years. I love you all, and I could not have done this without you.

Table of Contents

Abstract ii

Thesis Committee iv

Acknowledgements v

Table of Contents x

List of Tables xv

List of Figures xvi

1 Introduction 1

1.1 Thesis Statement 4

1.2 Outline 4

1.3 Contributions 6

2 Background 8

2.1 Problem Formulation 10

2.2 Domain Generalization 12

2.2.1 Optimization Techniques 13
2.2.1.1 Invariant Risk Minimization 13
2.2.1.2 Group Distributionally Robust Optimization 14
2.2.1.3 Meta-Learning for Domain Generalization . . 15
2.2.1.4 Robust Feature Extraction 16
2.2.2 Identifying Contexts 18
2.3 Relating to Generalization 19
2.3.1 Domain Adaptation 19
2.3.2 Style Transfer 20
2.3.3 Few-Shot and Zero-Shot Learning 21
2.3.4 Out-of-Distribution Detection 21
2.3.5 Adversarial Attacks 22
2.3.6 Robustness 22
2.3.7 Fairness of Generalization 22
2.3.8 Explainability 23
2.3.9 Performance Prediction 23
2.4 The Gap 24

3 Network Generalization Prediction with a Known Context Space **26**

3.1 Methods 27
3.1.1 Machine Learning Dependability 27
3.1.2 Derivation 30

3.1.2.1 Discrete-Bounded Context Space 30
3.1.2.2 Discrete-Unbounded or Continuous Context Space 31
3.1.2.3 Estimating Undependability 32
3.2 Experiments 33
3.2.1 Performance during Testing 35
3.2.2 Predicting Model Performance in Novel Operating Conditions 37
3.3 Discussion 38
3.3.1 Robot Manipulation Task 38
3.3.2 Dependable NNs in Practical Applications 40
3.4 Conclusions 41

4 Identifying the Context Subspace **42**
4.1 Feature Selection 43
4.2 Methods 44
4.2.1 Problem Formulation 44
4.2.2 Defining a Context Subspace 46
4.2.2.1 Ranking Context Features 47
4.2.2.2 Selecting the Context Subspace Dimensionality 48
4.2.3 Using the Context Subspace 50
4.2.4 Network Generalization Prediction 51
4.2.5 Comparing ΔI and I 51

4.3 Experimental Results . . . 53
4.3.1 Pedestrian Detection Generalization . . . 53
4.3.2 Defining the Context Subspace . . . 55
4.3.2.1 Ranking Context Features . . . 56
4.3.2.2 Selecting the Context Subspace Dimensionality 56
4.3.3 Using the Context Subspace . . . 57
4.3.4 Pedestrian Detection Generalization Prediction . . . 58
4.3.5 Generalization Prediction for Unseen Datasets . . . 60
4.4 Discussion . . . 62
4.5 Conclusions . . . 63

5 Mapping the Embedding Subspace 65

5.1 Methods . . . 66
5.1.1 Problem Formulation . . . 66
5.1.2 Decision Tree in Embedding Space . . . 68
5.1.3 Approximating Internal Test Set Manifold . . . 68
5.1.4 Inference on External Operating Data . . . 71
5.1.5 Network Generalization Prediction . . . 71
5.2 Experiments . . . 72
5.2.1 Pedestrian Classification . . . 72
5.2.2 Melanoma Classification . . . 74
5.2.3 Animal Classification . . . 76

5.2.4 Experimental Setup 77
5.2.5 Network Generalization Prediction 77
5.2.6 Numerical Network Generalization Prediction Results 78
5.2.7 Graphical Network Generalization Prediction Results . 79
5.3 Discussion 82
5.4 Conclusions 84

6 Extensions of NGP Subspaces **85**
6.1 Safety Functions for Neural Networks 86
6.2 Safety Function in the Context Space 87
6.2.1 Safety Function in the Context Space Results 88
6.3 Safety Function in the Embedding Space 89
6.3.1 Safety Function in the Embedding Subspace Results . . 90
6.4 Safety Functions Discussion 93
6.5 Predicting Robustness from the
Context Subspace 93
6.5.1 Predicting Robustness Results 95
6.5.2 Predicting Robustness Discussion 98
6.6 Conclusions 100

7 Conclusions **101**

Bibliography **104**

List of Tables

3.1 Notation 28

4.1 Notation. 45

5.1 NGP numerical F1 results for pedestrian, melanoma, and animal classification tasks with different architectures. 79

6.1 Error prediction F1 scores for pedestrian, melanoma, and animal classification tasks. 91

6.2 Error prediction F1 scores for classification tasks where the probability of failure in the operating domain (OD) and architecture (Arch.) is greater than or equal to 15%. 92

List of Figures

2.1 Context diagram. 9

3.1 The simulated robot manipulation task. To succeed, the robot must avoid the obstacle, which moves at a constant velocity v from right to left, starting at time τ, and reach or exceed a goal location, z, between 0 and 50 inches. τ_1: the obstacle has started moving. τ_2: the robot is avoiding collision with the obstacle. τ_3: the robot has successfully reached and/or exceeded its goal position without colliding with the obstacle. 35

3.2 The observed failures during testing, best viewed in color. Blue indicates a task failure. Pink indicates a harmful failure. The task failures (along the left 'wall' of the figure) occurred when the obstacle speed was less than or equal to 0.80 inches/second. The harmful failures (along the 'ceiling' of the figure) occurred when the robot goal was greater than or equal to 38.47 inches. 36

3.3 Predicted and observed performance of the trained NN in Novel Operating Conditions (OC). **Left:** OC Specification. $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian with a mean of μ and a standard deviation of σ. The sampled examples $x \sim \mathcal{N}(\mu, \sigma^2)$ are clipped to lie within the specified context $\mathbf{C}$. τ is not listed because $\tau \sim U(0, 10)$ for all conditions. **Right:** Predicted and observed performance of the trained NN in OCs. OC predicted performance shown left in light colors. Observed performance shown right in bold colors. ML Dependability $D_{\hat{\mathbf{X}}}(f)$ is shown as solid green, Task Undependability $T_{\hat{\mathbf{X}}}(f)$ is shown as blue hatched, and Harmful Undependability $H_{\hat{\mathbf{X}}}(f)$ is shown as pink dotted bars. 37

4.1 Overview of Network Generalization Prediction. 44

4.2 Defining the context subspace. 1) Rank Context Features: The $\Delta I(L, C)$ between different context features and the loss in the BDD Test Set for the first three rounds of Algorithm 1. Note that in iteration one, $\Delta I(L, C) = I(L, C)$ so the features' scores are non-negative. 2) Select K: We estimate the expected prediction error for different context subspace dimensionalities, K, and choose the dimensionality with the lowest expected prediction error: in this case, $K = 3$. We form the context subspace with the three most informative context features: brightness, safety critical flag, and the scene type. 57

4.3 BDD Novel Operating Domains. We define operating domains based on the time of day and the number of pedestrians in an image. Images with fewer than 5 (N)SC pedestrians fall under small groups. Images with 5 or more (N)SC pedestrians fall under large groups. Sample images from the operating domains are shown. NSC pedestrians are outlined in blue. SC pedestrians are outlined in red. Drivable area is shown in random transparent colors. 58

4.4 Network Generalization Prediction (NGP) Results. NSC pedestrian recall and SC pedestrian recall are shown separately. The observed pedestrian recall for the images of each operating domain are shown in bright blue or red. The NGP predicted recall is shown in light blue or red. The naïve baseline indicates the average recall over all pedestrians in the Test Set. Note that the naïve baseline is the same for every operating domain. . . 59

4.5 Unseen Dataset Novel Operating Domains. Sample images from the unseen datasets. NSC pedestrians outlined in blue. SC pedestrians outlined in red. 60

4.6 Network Generalization Prediction for Unseen Datasets. NSC pedestrian recall and SC pedestrian recall are shown separately. The observed pedestrian recall for the images of each novel dataset is shown in bright blue or red. The NGP predicted recall is shown in light blue or red. The naïve baseline indicates the average recall over all pedestrians in the Test Set. Note that the naïve baseline is the same for both unseen datasets. 61

5.1 Components of a typical feed-forward Deep Neural Network (NN): convolutional layers, fully connected layers, and the prediction layer. The prediction layer is also a fully-connected layer that projects the final embedding, $\phi(\mathbf{x})$, into the prediction dimension. 67

5.2 An illustration of the decision tree for mapping NN embeddings. Test data lie on a manifold in the embedding space. We identify structure in the embedding space as it relates to the NN outcome. For binary classification the possible outcomes are true positive (TP), false negative (FN), false positive (FP) and true negative (TN). The structure identified using labeled test data can be leveraged to predict the NN's performance on unlabeled operating data, where the outcome is unknown. Best viewed in color. 70

5.3 Classification tasks. X indicates the internal dataset that is used to train the NN classifier and fit the embedding decision tree. $\hat{X}$ indicates the unlabeled, external operating dataset. For each dataset, the top row shows a random sampling of negative examples, and the bottom row shows a random sampling of positive examples. 73

5.4 DenseNet JAAD visualized results for Network Generalization Prediction. 80

5.5 Network Generalization Prediction results for pedestrian classification, melanoma classification, and animal classification. We show results for three NN architectures: VGG, AlexNet, and DenseNet. 82

6.1 A comparison of the NN performance without the safety function and with the safety function. Task failures are indicated in blue. Harmful failures are indicated in pink. (a) a reprint of Figure 3.2 to facilitate comparison. (b) the observed failures in Testing Conditions with the safety function. (c) a comparison of the NN ML Dependability, Task Undependability, and Harmful Undependability with and without the safety function. Note, the Harmful Undependability is reduced from 5.47% to 0.007% with the safety function. 88

6.2 Sampling of images from the HAM and SIIM-ISIC Skin Lesion datasets separated by malignant/benign labels. For each category the images are shown according to their hue and saturation with consistent ranges across datasets. Hue/saturation ranges without available images are shown as solid colors. Note that the malignant images in the HAM dataset exhibit a smaller spread over saturation compared to the HAM benign images or the SIIM-ISIC images. 96

6.3 Left Top: HAM Dataset test image count indexed by image saturation, note the histogram vertical axes are not at the same scale. Left Bottom: classifier Sensitivity and Specificity indexed by image saturation. Right Top: SIIM-ISIC Dataset test image count indexed by image saturation, note the histogram vertical axes are not at the same scale. Right Bottom: classifier Sensitivity and Specificity indexed by image saturation. Sensitivity and Specificity bars are colored green for high performance and red for poor performance. 98

6.4 Classifier AUC on sub-populations of the HAM test images. For each plot, the True Positive Rate is shown on the y-axis and the False Positive Rate is shown on the x-axis. The overall AUC curve is shown in black in each plot, the sub-population AUC curve is shown in green. We define low saturation as $< 25\%$ and high saturation $\geq 25\%$. We define younger as < 50 years and older as ≥ 50 years. 99

Chapter 1

Introduction

Neural Networks (NNs) perform safety critical tasks in unconstrained environments, e.g., autonomous robot control, perception for self-driving vehicles [1], and medical image analysis, but the structure of NNs is different from other safety critical software. Safety critical software is typically regulated by international functional safety standards, e.g., ISO 26262, IEC 61508. Functional safety standards leverage various techniques to verify the safety of software, including requirement specification, i.e., linking required system behavior to specific code modules, white box testing, i.e., testing specific inputs that cover all branches or behavior in the code, and code review to identify human error. These techniques are challenging or impossible to apply directly to NNs, e.g., labeled data is used to implicitly specify the correct behavior in supervised learning, NNs are black box systems, and NN weights cannot be manually inspected to identify failure cases.

In addition, there is a growing body of evidence that NNs are susceptible to biases that can discriminate on individuals or subpopulations based on race, sex, age, disability, etc. When NNs are used in cyber-physical systems,

this risk for discrimination is compounded by a risk of physical harm such that different subgroups face greater risk of harm than others. Pedestrians with a dark skin-tone face a greater risk of not being detected by autonomous vehicles and thus being struck by them [2]. The vast majority of images in publicly available skin cancer datasets are of patients with lighter skin, leaving patients with darker skin at risk of lower performance [3]. There is an urgent need for technical solutions that can predict NN performance, and NN harm, in novel operating domains.

Prior work has investigated how to improve NN performance in novel operating domains under different research objectives, e.g., domain generalization and adaptation, NN robustness, out-of-distribution (OOD) rejection, and adversarial attacks. The vast majority of research focuses on training, i.e., improving how NNs learn to achieve better NN performance. But the definition of “better performance" is typically very limited, i.e., better average success on a predefined dataset. From the average performance it is not clear in what conditions performance will be better and for whom performance will improve.

The domain generalization community proposed that NN performance changes when the data environment changes, e.g., changes in the appearance of buildings and the structure of side-walks across cities, changes in medical scans across different scanner manufacturers, etc.; see Section 2.2 for more details. For NNs that generalize better, work in domain generalization separates the data back into the original environments so that the NN can be trained to perform well across the distinct environments, e.g., unshuffling training

data back into the original data environments [4]. But, prior work relies on multiple datasets or known, observed environment contexts [5] [6].

For many practical applications of NNs, identifying the different environments that impact NN performance is challenging. Large datasets with thousands or millions of examples are used in training and testing where each example may be complex, e.g., robot task demonstrations, high resolution images, medical scans, etc. It is often not scalable to manually inspect failure cases to determine what environment factors lead to changes in NN performance. Failures are typically attributed to needing “more data", but identifying specifically what additional examples are needed is not obvious. Identifying different contexts within a dataset and relating those contexts back to actionable, interpretable attributes without a priori knowledge of what impacts NN performance is an open problem.

In this thesis, we propose that the key to predicting NN performance in a novel operating domain is to discover what environment contexts impact NN performance. We derive an algorithm to predict NN performance in a novel operating domain from a fixed, finite test set by identifying different environment contexts of interest and estimating NN performance in these contexts. We demonstrate this leads to accurate NN performance predictions leveraging environment contexts from labeled metadata of interest or from the NN embedding space.

1.1 Thesis Statement

NN generalization in a novel operating domain can be predicted by identifying structure in the environment context or in the NN embedding that is informative for NN performance.

1.2 Outline

In Chapter 2 we outline our problem formulation and discuss prior works in domain generalization. In Chapter 3 we develop methods for predicting NN performance in a novel operating domain, a task we denote Network Generalization Prediction (NGP). In Section 3.1.1 we propose the metrics ML Dependability, Task Undependability, and Harmful Undependability to measure the probability of success, the probability of failing at the task but not causing harm, and the probability of causing harm, respectively, in a novel operating domain. We begin by assuming that the factors that impact the environment context, denoted context features, are both known and observed. In Section 3.1.2 we derive methods for NGP from a finite test set using the known context features. In Section 3.2 we demonstrate accurate NGP for a robot controller trained in simulation via Reinforcement Learning. The material in Chapter 3 was published as a workshop paper:

- O'Brien, Molly, William Goble, Greg Hager, and Julia Bukowski. "Dependable neural networks for safety critical tasks." In Engineering Dependable and Secure Machine Learning Systems: Third International Workshop, EDSMLS 2020, New York City, NY, USA, February 7, 2020,

Revised Selected Papers, pp. 126-140. Springer, Cham, 2020.

In general, the environment context that impacts NN performance is unknown. In Chapter 4 we propose an algorithm to identify which context features are informative for NGP. These informative context features form the Context Subspace; see Section 4.2.2. Using the Context Subspace, we accurately predict the pedestrian recall for a pedestrian detector deployed in novel operating conditions; see Section 4.3. The material in Chapter 4 was published as a conference paper:

- O'Brien, Molly, Mike Medoff, Julia Bukowski, and Greg Hager. "Network Generalization Prediction for Safety Critical Tasks in Novel Operating Domains." In Winter Conference on Applications of Computer Vision. 2022.

In Chapter 5 we do not assume access to known context features; instead we leverage the NN embedding. We map the NN Embedding Subspace to define local regions that are informative for the NN performance; see Section 5.1. We demonstrate accurate NGP across a variety of NN architectures and image classification tasks including pedestrian classification and melanoma classification; see Section 5.2. The material in Chapter 5 is under review:

- O'Brien, Molly, Julia Bukowski, Mathias Unberath, Aria Peseshk*, and Greg Hager*. "Mapping DNN Embedding Manifolds for Network Generalization Prediction." Under review. 2022.

In Chapter 6 we explore extensions of the subspaces beyond NGP. First, we investigate adding safety functions, external logic outside the NN to improve

safety. In Section 6.2 we find that safety functions substantially increase the overall safety for a robot control task. In Section 6.3 we use the Embedding Subspace to reject operating images that are likely misclassified and find a significant increase in safety. The motivation for this thesis is safety, e.g., how to measure the safety of a NN, how to predict whether a NN will be safe in a novel operating domain, etc., but our work has implications broader than safety critical applications. We define the Context Subspace and the Embedding Subspace to enable NGP, but we find these subspaces are useful for other tasks. The dimensions of the context subspace are interpretable and provide actionable information. In Section 6.5 we leverage the Context Subspace to determine if a trained NN is expected to be robust. The material in Section 6.5 was published as a conference paper:

- O'Brien, Molly, Julia Bukowski, Greg Hager, Aria Peseshk, and Mathias Unberath. "Evaluating Neural Network Robustness for Melanoma Classification using Mutual Information." In International Society for Optics and Photonics Conference on Medical Imaging. 2022.

1.3 Contributions

The contributions of this thesis are:

1. Performance metrics to measure both the probability of success and the probability of harm (Chapter 3).
2. Methods for Network Generalization Prediction (NGP) (Chapter 3).
3. Identifying relevant context features for NGP (Chapter 4).

4. Mapping the NN embedding space for NGP (Chapter 5).

5. For the first time, we propose safety functions for NNs and demonstrate significantly safer performance (Chapter 6).

6. We demonstrate that inspecting the context features that impact NN performance to determine whether the NN is expected to be robust in novel operating domains (Chapter 6).

Chapter 2

Background

In this thesis, we focus on NN generalization in changing environments. Changing environments are often described as “distribution shifts" in the literature. In low dimensional data, e.g., tabular data, the notion of distribution shift is straight-forward, i.e., the distribution of values across given attributes changes. But many of the safety critical applications of NNs consider high dimensional data, e.g., images, and for high-dimensional data modalities the notion of distribution shift becomes harder to quantify. To help frame the problem, let us consider the task of image classification. Imagine an image formation process, see Figure 2.1, where the image is impacted by the label and the context. The image label describes the class of the primary subject in the image. The context describes other information about the image that is not the task label. In Figure 2.1 the available information about the context, hereafter referred to as context features, are the time of day, and the weather. Distribution shift in images could refer to changes in the distribution of different labels in the dataset, or distribution shift could refer to other shifts in the dataset, i.e., shifts in context. Compensating for changes

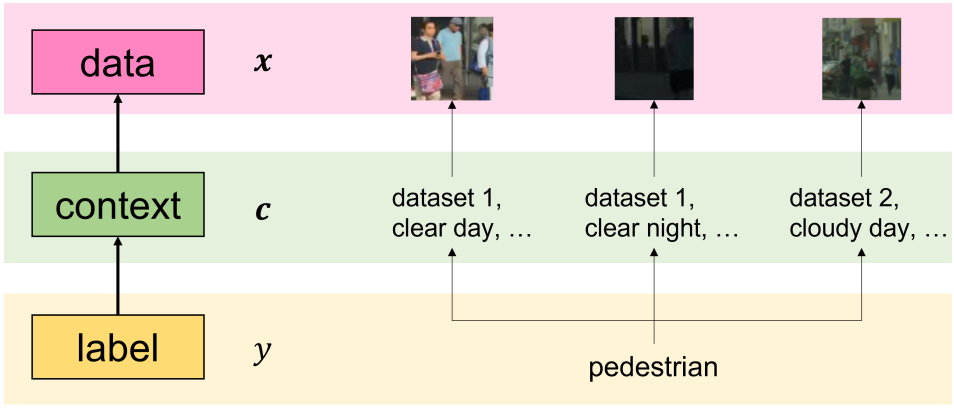

Figure 2.1: Context diagram.

in label distribution is standard practice in ML, e.g., uniform sampling of different label classes during training. Understanding and compensating for distribution shifts in context is more challenging.

It is well known that NN performance often degrades when the context shifts, e.g., in novel environments [7]. But prior works in domain generalization sidestep rigorous context specification by relying on different datasets to implicitly capture context shift [5], [8]–[12] or by only considering one context feature that has a spurious correlation with the label [6]. Relying on recording more and more data to implicitly capture context shifts is expensive in both time and money, it does not provide insight into what impacts NN performance, nor does it guarantee that the shifts in context the NN will observe in a novel operating domain are captured by previous data collection. Relying on one known, observed context feature that impacts performance is an over simplification of most interesting applications. See Section 2.2.2 for a more

detailed description of how prior works identify different contexts. Furthermore, while many domain generalization algorithms have been proposed in the last decade, none has consistently out-performed standard Empirical Risk Minimization (ERM) [13].

Distribution shift is tied to context shift for high dimensional data modalities, and yet prior work does not address specifying or understanding context shift in high dimensional data like image classification. In this chapter, we present our problem formulation and formalize the notation that will be used in this thesis. We then outline the prior work pertaining to NN generalization in unconstrained environments. Finally, we describe the fundamental technical gap that is bridged by this thesis.

2.1 Problem Formulation

We consider a feed-forward NN, f, where f is composed of convolutional, linear, and non-linear layers. Let $(\mathbf{x}, y, \mathbf{c})$ be a sample from the internal dataset that is used for training and testing, where $\mathbf{x}$ is the input data, e.g., an image, y is the label, and $\mathbf{c}$ is the available context. Recall, $\mathbf{c}$ captures additional information about $\mathbf{x}$ that is not the label y, but it is not known whether or how $\mathbf{c}$ impacts the performance of f. If the internal dataset is used for both training and testing, it is partitioned into a training set $(\overline{\mathbf{X}}, \overline{\mathbf{y}}, \overline{\mathbf{C}}) = \{(\overline{\mathbf{x}}_i, \overline{y}_i, \overline{\mathbf{c}}_i)\}_{i=1}^{\overline{N}}$ and a test set $(\mathbf{X}, \mathbf{y}, \mathbf{C}) = \{\mathbf{x}_i, y_i, \mathbf{c}_i\}_{i=1}^{N}$ such that the training set and the test set are disjoint partitions of the internal dataset. During training, the training set is partitioned into training data and validation data.

Let $\ell(f(\mathbf{x}), y)$ indicate a loss function; the mathematical loss for f on the

internal training set is:

$$\mathbb{E}[\ell(f(\overline{\mathbf{X}}), \overline{\mathbf{y}})] \tag{2.1}$$

We define $\mathcal{L}^{\overline{\mathbf{X}}}$ as the empirical loss on training data $\overline{\mathbf{X}}$.

$$\mathcal{L}^{\overline{\mathbf{X}}} = 1/\overline{N} \sum_{i=1}^{\overline{N}} \ell(f(\overline{\mathbf{x}}_i), \overline{y}_i) \tag{2.2}$$

Standard NN training is performed via ERM [14], where f is trained to minimize $\mathcal{L}^{\overline{\mathbf{X}}}$. We define $g(\mathbf{c})$ to be the empirical loss in context $\mathbf{c}$. Let $\mathbb{I}(a, b)$ be an indicator function that is equal to 1 if $a = b$ and 0 otherwise. $g(\mathbf{c})$ can be computed as:

$$g(\mathbf{c}) = \frac{\sum_{i=1}^{\overline{N}} \mathbb{I}(\overline{\mathbf{c}}_i, \mathbf{c}) * \ell(f(\overline{\mathbf{x}}_i), \overline{y}_i)}{\sum_{i=1}^{\overline{N}} \mathbb{I}(\overline{\mathbf{c}}_i, \mathbf{c})} \tag{2.3}$$

The empirical likelihood of $\mathbf{c}$ in $\overline{\mathbf{X}}$, $p_{\overline{\mathbf{X}}}(\mathbf{c})$, can be computed as:

$$p_{\overline{\mathbf{X}}}(\mathbf{c}) = \frac{\sum_{i=1}^{\overline{N}} \mathbb{I}(\overline{\mathbf{c}}_i, \mathbf{c})}{\overline{N}} \tag{2.4}$$

$\mathcal{L}^{\overline{\mathbf{X}}}$ can equivalently be computed as:

$$\mathcal{L}^{\overline{\mathbf{X}}} = \sum_{\mathbf{c} \in \overline{\mathbf{C}}} p_{\overline{\mathbf{X}}}(\mathbf{c}) g(\mathbf{c}) \tag{2.5}$$

Therefore, in standard NN training, the emphasis in the training loss on context $\mathbf{c}$ is proportional to the frequency of $\mathbf{c}$ in the training dataset $\overline{\mathbf{X}}$.

In general, the average performance of f on the test set, $\mathbf{X}$, is used as an indicator for how f will perform on unseen data. The empirical test loss is computed as:

$$\mathcal{L}^{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}_i), y_i) \tag{2.6}$$

The empirical likelihood of $\mathbf{c}$ in $\mathbf{X}$, $p_{\mathbf{X}}(\mathbf{c})$, can be computed as:

$$p_{\mathbf{X}}(\mathbf{c}) = \frac{\sum_{i=1}^{N} \mathbb{I}(\mathbf{c}_i, \mathbf{c})}{N} \tag{2.7}$$

The loss for the test set, $\mathcal{L}^{\mathbf{X}}$, can be computed as:

$$\mathcal{L}^{\mathbf{X}} = \sum_{\mathbf{c} \in \mathbf{C}} p_{\mathbf{X}}(\mathbf{c}) g(\mathbf{c}) \tag{2.8}$$

Typically, $p_{\overline{\mathbf{X}}}(\mathbf{c}) \approx p_{\mathbf{X}}(\mathbf{c})$. Therefore, testing with $\mathbf{X}$ may not reveal important limitations of f, e.g., f may have poor performance in a context $\mathbf{c}$ that is rare in the training set and the test set, the same spurious correlations between $\mathbf{c}$ and y may exist in the training set and the test set, etc.

We are interested in the performance of f when it encounters data $\hat{\mathbf{X}}$ from a novel operating domain. There is often a context distribution shift between the internal dataset and the novel operating domain, i.e., $p_{\hat{\mathbf{X}}}(\mathbf{c}) \neq p_{\overline{\mathbf{X}}}(\mathbf{c})$. The expected empirical loss in the novel operating domain is:

$$\mathcal{L}^{\hat{\mathbf{X}}} = \sum_{\mathbf{c} \in \mathbf{C}} p_{\hat{\mathbf{X}}}(\mathbf{c}) g(\mathbf{c}) \tag{2.9}$$

When the context distribution of $\hat{\mathbf{X}}$ is different from the distribution of $\overline{\mathbf{X}}$, the loss of f is not optimized for performance in this domain and it is likely that the performance of f will degrade; domain generalization aims to address this problem.

2.2 Domain Generalization

NN performance degrades when the data distribution changes from $\overline{\mathbf{X}}$ to $\hat{\mathbf{X}}$ [7]. Broadly, domain generalization techniques identify different environment

contexts within $\overline{\mathbf{X}}$ and then modify standard ERM optimization so that the performance of f is consistent across environments. First we describe the optimization techniques that have been proposed, then we outline the different ways that environments within $\overline{\mathbf{X}}$ have been identified.

2.2.1 Optimization Techniques

2.2.1.1 Invariant Risk Minimization

One of the primary domain generalization techniques is Invariant Risk Minimization (IRM). IRM takes inspiration from causal inference to train NNs that exploit causal relationships in the task of interest instead of learning spurious correlations present in the training data [5]. IRM considers multiple training datasets originating from different environments with the goal of learning f "which performs well across a large set of unseen but related environments" [5]. The NN being optimized, f, is decomposed into two components: a feature extractor, ϕ, and a classifier or predictor, w. $\phi(\mathbf{x})$ indicates projecting the input data into some embedding space, where $w(\phi(\mathbf{x}))$ is equivalent to passing example x through the entire NN, i.e., $w(\phi(\mathbf{x})) = f(\mathbf{x})$.

The authors assume that multiple datasets are available and each dataset is considered one environment context, i.e., $\overline{\mathbf{c}}_1 \rightarrow (\overline{\mathbf{X}}^1, \overline{\mathbf{y}}^1)$, $\overline{\mathbf{c}}_2 \rightarrow (\overline{\mathbf{X}}^2, \overline{\mathbf{y}}^2)$, etc. IRM aims to find a robust feature extractor, ϕ, across all environments while maintaining a perfect classifier, w by minimizing the following objective function:

$$\begin{aligned} \min \quad & \sum_{\mathbf{c} \in \overline{\mathbf{C}}} \ell(w(\phi(\overline{\mathbf{X}})), \overline{\mathbf{y}} | \mathbf{c}) \\ s.t. \quad & w \in \arg\min_{\overline{w}} \ell(\overline{w}(\phi(\overline{\mathbf{X}})), \overline{\mathbf{y}} | \mathbf{c}) \quad \forall \mathbf{c} \in \overline{\mathbf{C}} \end{aligned} \tag{2.10}$$

Note that this is equivalent to

$$\begin{aligned} \min \quad & \sum_{\mathbf{c} \in \overline{\mathbf{C}}} g(\mathbf{c}) \\ s.t. \quad & w \in \arg\min_{\overline{w}} \ell(\overline{w}(\phi(\overline{\mathbf{X}})), \overline{\mathbf{y}}|\mathbf{c}) \quad \forall \mathbf{c} \in \overline{\mathbf{C}} \end{aligned} \tag{2.11}$$

Effectively, instead of weighting each environment context, $\mathbf{c}$, by how frequently it was seen in training, IRM weights each context uniformly.

2.2.1.2 Group Distributionally Robust Optimization

Like IRM, Group Distributionally Robust Optimization (DRO) [6] aims to minimize the worst-case loss of f. With DRO, Sagawa et al. refer to different groups of data; this is analogous to the different environment contexts we consider. Recall, $p_{\overline{\mathbf{X}}}(\mathbf{c})$ denotes the probability of environment context $\mathbf{c}$ in the training data. Let $p_{\overline{\mathbf{X}}}(\overline{\mathbf{C}})$ indicate the overall probability distribution of contexts in the training set $\overline{\mathbf{X}}$. Consider the expected loss over the training data:

$$\mathcal{L}^{\overline{\mathbf{X}}} = \mathbb{E}[\ell(f(\overline{\mathbf{X}}), \overline{\mathbf{y}}|\mathbf{c})], \quad \mathbf{c} \sim p_{\overline{\mathbf{X}}}(\overline{\mathbf{C}}) \tag{2.12}$$

$\overline{\mathbf{X}}$ is assumed to contain a mixture of different contexts, $\mathbf{c}$, determined by the environment and the example label y. Now consider a set of distributions $\mathcal{G}$ with $g \in \mathcal{G}$, where $p_g(\overline{\mathbf{C}})$ is one context distribution of interest. The worst case performance of f is:

$$\max_{g \in \mathcal{G}} \mathbb{E}[\ell(f(\overline{\mathbf{X}}), \overline{\mathbf{y}}|\mathbf{c})], \quad \mathbf{c} \sim p_g(\overline{\mathbf{C}}) \tag{2.13}$$

The objective function that DRO minimizes is:

$$\arg\min_{f}\{\max_{g\in\mathcal{G}}\mathbb{E}[\ell(f(\overline{\mathbf{X}}),\overline{\mathbf{y}}|\mathbf{c})],\quad \mathbf{c}\sim p_g(\overline{\mathbf{C}})\} \tag{2.14}$$

which minimizes the worst case loss.

2.2.1.3 Meta-Learning for Domain Generalization

Meta-Learning for Domain Generalization (MLDG) aims to train models that generalize well by synthesizing potential test domains in training [8]. MLDG assumes access to multiple environment contexts, denoted source domains in the paper. The internal training set and test set are selected so that some environment contexts are used only in training and some environment contexts are used only in testing, i.e., if $\mathbf{c}\in\overline{\mathbf{C}}\rightarrow\mathbf{c}\notin\mathbf{C}$, and if $\mathbf{c}\in\mathbf{C}\rightarrow\mathbf{c}\notin\overline{\mathbf{C}}$. The training loss is computed as:

$$\mathcal{L}^{\overline{\mathbf{X}}}=\frac{1}{|\overline{\mathbf{C}}|}\sum_{\mathbf{c}\in\overline{\mathbf{C}}}\mathbb{E}[\ell(f(\overline{\mathbf{X}}),\overline{\mathbf{y}}|\mathbf{c})] \tag{2.15}$$

The test loss is computed as:

$$\mathcal{L}^{\mathbf{X}}=\frac{1}{|\mathbf{C}|}\sum_{\mathbf{c}\in\mathbf{C}}\mathbb{E}[\ell(f(\mathbf{X}),\mathbf{y}|\mathbf{c})] \tag{2.16}$$

As we have seen in other domain generalization algorithms, the training and test loss weight each environment context uniformly. Let Θ indicate the parameters in f. We use the notation $\mathcal{L}(\Theta)$ to denote the loss for given NN parameters Θ. After the training loss is computed, Θ is updated to be Θ'. The meta-learning objective function takes into account both the training set loss

and the test set loss:

$$\arg\min_{f} \mathcal{L}^{\overline{\mathbf{X}}}(\Theta) + \beta \mathcal{L}^{\mathbf{X}}(\Theta') \tag{2.17}$$

where β is a constant.

2.2.1.4 Robust Feature Extraction

Domain-Adversarial Training of Neural Networks (DANN) was proposed as a representation learning approach to learn f so that it is robust to environment context changes, denoted domains by the authors [11]. Let $\overline{\mathbf{X}}$ and $\overline{\mathbf{y}}$ be the training data and labels respectively and let $\hat{\mathbf{X}}$ be a set of unlabeled operating data. As in Section 2.2.1.1, we decompose the NN f into a feature extractor, ϕ, and a linear classifier, w, where $w(\phi(\mathbf{x})) = f(\mathbf{x})$. With DANN, it is proposed to add an additional component to the NN architecture, d, where $d(\phi(\mathbf{x}))$ aims to predict the context from which the example x originated. The contexts are defined based on the datasets, e.g., $\mathbf{c}_1 \to \overline{\mathbf{X}}$, $\mathbf{c}_2 \to \hat{\mathbf{X}}$, etc. The loss associated with the task is computed as $\mathcal{L}^{\phi,w}$.

$$\mathcal{L}^{\phi,w} = \mathbb{E}[\ell(w(\phi(\overline{\mathbf{X}})), \overline{\mathbf{y}})] \tag{2.18}$$

The loss associated with the context classification is computed as $\mathcal{L}^{\phi,d}$. Let $\mathcal{X}$ denote the union of $\overline{\mathbf{X}}$ and $\hat{\mathbf{X}}$ and let $\mathbf{C}$ denote the domains associated with $\mathcal{X}$.

$$\mathcal{L}^{\phi,d} = \mathbb{E}[\ell(d(\phi(\mathcal{X})), \mathbf{C})] \tag{2.19}$$

The NN is trained by optimizing the adversarial objective function:

$$\arg\min_{\phi,w,d} \mathcal{L}^{\phi,w} - \gamma \mathcal{L}^{\phi,d} \tag{2.20}$$

where γ is a constant.

Deep CORAL (CORAL) was proposed to align features for labeled training data $\overline{\mathbf{X}}$ and unlabeled operating data $\hat{\mathbf{X}}$ [12]. Let $Cov(\phi(\mathbf{X}))$ indicate the covariance matrix for the feature embedding of data $\mathbf{X}$. The NN is trained to both minimize the error on the task of interest, and minimize the difference of feature covariances across the training data and the operating data.

$$\min \mathbb{E}[\ell(w(\phi(\overline{\mathbf{X}})), \overline{\mathbf{y}})] + \gamma ||Cov(\phi(\overline{\mathbf{X}})) - Cov(\phi(\hat{\mathbf{X}}))||_F^2 \tag{2.21}$$

where $|| \bullet ||_F^2$ indicates the Frobenius norm.

Other methods to learn robust features include Deep Domain Generalization via Conditional Invariant Adversarial Networks (CDANN) [9] and Adversarial Autoencoders with Maximum Mean Discrepancy (MMD) [10]. CDANN extends DANN to deeper NNs and minimizes how $P(\overline{\mathbf{X}}|\overline{\mathbf{y}})$ changes across contexts in order to make $P(\overline{\mathbf{y}})$ constant across contexts. MMD takes advantage of an autoencoder architecture and an adversarial loss function to align the feature distributions across different environment contexts. If there is a known bias in the training data, other previous work has proposed that robust features can be learned by training the NN to minimize the Mutual Information between the NN embedding and the known bias [15]. Permuting images during training has been proposed to implicitly train ϕ in a way that generalizes to new environment contexts [16].

2.2.2 Identifying Contexts

The proposed domain generalization techniques aim to train f so that it generalizes to new environment contexts, and the environment contexts are defined in two ways: 1. data from multiple environments is available and the NN is trained to generalize across the environments, or 2. there exists one known, observed attribute that impacts NN performance, and training addresses changes in this attribute. IRM [5], MLDG [8], CDANN [9], and MMD [10] require labeled training data from multiple environment contexts. DANN [11] and CORAL [12] require labeled training data from one environment context and unlabeled operating data from another environment context. In the experiments for DRO, there is one known and observed attribute that has a spurious correlation with the label; this knowledge is used to form environment contexts based on different groups of data [6]. Note that none of these works address the challenge of identifying different environments, associated with different levels of performance, within one dataset.

The limitations of this approach are being addressed by a new research direction within domain generalization: hidden-stratification, i.e., the idea that there are sub-classes within an environment context where the NN performs poorly. Hidden stratification can lead to harm if the task is safety critical, e.g., medical image analysis [17]. Sohoni et al. propose the framework GEORGE that identifies subgroups of data by clustering examples in the NN embedding space and training classifiers that demonstrate robust performance across subgroups, z, where $z \in \mathbf{c}$ and $\mathbf{c} \in \overline{\mathbf{C}}$ [18]. Let $\mathcal{F}$ be the set of possible NNs. GEORGE aims to train f so that it performs well across subgroups within the

different environment contexts:

$$\arg\max_{f\in\mathcal{F}}\min_{z\in\mathbf{c}}\quad\sum_{\mathbf{c}\in\overline{\mathbf{C}}}\ell(f(\overline{\mathbf{X}}),\overline{\mathbf{y}}|z) \tag{2.22}$$

Work in hidden stratification is a promising direction to further improve NN generalization, but it also underscores that relying on known attributes or available datasets is insufficient to truly understand what impacts NN performance and why a NN fails to generalize in a specific application.

2.3 Relating to Generalization

We have framed the problem of NNs in unconstrained environments in terms of domain generalization, but there are other areas of research that relate to NN generalization, including domain adaptation, style transfer, few-shot learning, out-of-distribution detection (OOD), adversarial attacks, robustness, and fairness of generalization. Underspecification can lead network performance to degrade when deployed in operating domains different from the training domains [19]. Prior work has also investigated how to ensure safety during network training [20], [21].

2.3.1 Domain Adaptation

Domain adaptation seeks to adjust a trained network to new operating domains. See [22] for a survey of visual domain adaptation techniques. An adaptation of DRO, Class-conditional Subgroup Robustness (SGDRO), has been proposed to address the performance of subgroups within each context by first, learning a GAN to transform samples from one subgroup to another

and second, balancing subgroup performance via regularization [23]. Domain Mixup has been proposed to interpolate between available domains to improve generalization [24] RoyChowdhury et al. propose a method to leverage unlabeled data in a new operating domain to fine-tune a trained network and show an increase in pedestrian detection over baseline for a network trained using sunny images from the Berkely Deep Drive Dataset (BDD100K) [25] and adapted to rainy, overcast, snowy day, and night images [26]. Liu et al. address Open Domain Adaptation (generalizing to an unseen target domain) and Compound Domain Adaptation (generalizing to combined target domains) and demonstrate results on a compound target of rainy, cloudy and snowy and an open target of overcast images [27].

2.3.2 Style Transfer

In perception, style transfer is used to render images from one domain as if they were from another; see [28] for a survey. Style transfer can be used in safety critical tasks to render a novel scenario in a known style. CycleGANs have achieved impressive results rendering photographs as if they were painted by different artists and transferring the style of similar animals, e.g., rendering a horse as a zebra [29]. Gong et al. extend CycleGANs for continuous style generation flowing from one domain to another and demonstrate results transferring styles between object detection datasets [30].

2.3.3 Few-Shot and Zero-Shot Learning

Few-shot (zero-shot) learning aims to learn a task for given operating conditions with little (no) labeled training data. James et al. use a task embedding to leverage knowledge from previously learned, similar tasks [31] and demonstrate that a robot can learn new tasks with only one real-world demonstration. See [32], [33] for surveys of zero-shot learning.

2.3.4 Out-of-Distribution Detection

NNs are trained in limited environments, e.g., to classify dogs and cats. If a NN is used in an unconstrained environment, an input sample from outside of that limited training environment, e.g., an image of a bird, standard NNs will provide an incorrect answer because neither dog nor cat can be correct. Automatically recognizing OOD samples is a broad area of research that is relevant to safely deploying NNs in unconstrained environments. Many prior works use the NN embedding, i.e., the output from the penultimate NN layer, or the softmax scores to detect OOD samples [34]–[38]. The baseline in [34] uses the softmax scores to predict whether an image is misclassified in addition to OOD detection. Previous work has investigated input sample Euclidean or Mahalanobis distance from training data in the embedding space to identify OOD and adversarial examples [39]–[41]. Recent work proposed the Multi-level Out-of-distribution Detection (MOOD) framework for computationally efficient OOD [42]. Previous work can detect if an input is from an unseen Operating Domain [43], [44].

2.3.5 Adversarial Attacks

NNs are sensitive to noise and common corruptions [45] [46], Adversarial Attacks can change a NN prediction by modifying only a few pixels [47], NNs can make errors with high confidence from imperceptible noise and even on natural images **hendrycks2019natural** or rotated natural images [48]. Adversarial attacks are also relevant for cyber-physical tasks [49]. For automated driving, physical stickers or graffiti on street signs can change NN perception predictions [50]. Adversarial-shape objects can infer with LiDAR-based autonomous driving navigation systems [51].

2.3.6 Robustness

Robustness in ML is used to refer to maintaining task performance in the presence of dataset shift and noise or corruptions. See [52] for a recent survey of robustness in ML. Subbaswamy et al. propose to evaluate a model's robustness to distribution shifts with one fixed evaluation set [53] and are able to identify which shifts in distribution would lead to the worst-case performance. Cygert et al. investigate NN pedestrian detection robustness to different image distortions [54].

2.3.7 Fairness of Generalization

Training DNNs that are fair to different subpopulations is essential to safely deploy DNNs in unconstrained environments. ImageNet is one of the most commonly used datasets in computer vision, the dataset alone has been cited over 35,000 times in the last ten years. But it has been shown that ImageNet

contains offensive language to describe people in the dataset and biased representation of different groups of people, e.g., programmer, banker, etc. [55] There is evidence that both pedestrian detection and melanoma classification can have lower performance for some subpopulations, particularly people with darker skin tones. Wilson et al. investigated pedestrian detection with the BDD dataset and found poorer pedestrian prediction for darker skin tones that is not explained by confounding variables like time of day or occlusion [2]. Wen et al. performed a systematic review of publicly available skin image datasets and found a substantial under-representation of darker skin types [3]. It has been shown that DRO [6] can control the risk to minority groups, e.g., non-native speakers in speech recognition tasks, without requiring the identity of the groups [56].

2.3.8 Explainability

Recent work also aims to explain the NN prediction. See [57] for a recent survey on explainable Deep Learning. Generative Adversarial Network (GAN) inversion can be used to visualize changes in the embedding space when the task involves GANs for image generation [58].

2.3.9 Performance Prediction

NN generalization hinges on how well a NN performs in a novel operating domain. We have outlined the many techniques proposed to improve NN generalization, but, strikingly, very little work has focused on predicting exactly how well a NN will generalize. A similar task, Detection Performance

Modeling, was proposed in [59] where Ponn et al. train a random forest on image attributes, e.g., pedestrian occlusion, bounding box size, presence of rain, etc., to predict whether a specific pedestrian in an image will be detected. This method requires significant labeled metadata and only makes predictions for one given pedestrian.

2.4 The Gap

While significant research effort has focused on improving NN performance in unconstrained environments, very little attention has been paid to predicting what the NN performance will be in a novel operating domain. State of the art research focuses training NNs with *better* performance without proposing techniques to determine if performance is *good enough* for a given application or a given operating environment. Determining whether NN performance is sufficient is essential for safely deploying NNs in unconstrained environments; this question can only be answered by predicting how a NN will generalize to a novel operating domain.

Separately, the different environment contexts considered in domain generalization research are derived from separate datasets or one known, observed attribute. In general, some context features may be available, but, it is not clear which of these context features are related to performance degradation. Additionally, there are usually context shifts between that are not captured by the context features.

We bridge both of these gaps in this thesis by first proposing a methodology to predict NN performance in a novel operating domain using one fixed test set

(Chapter 3) and then discovering the relevant environment contexts *within* a test dataset based on interpretable metadata (Chapter 4) or the NN embedding space (Chapter 5).

Chapter 3

Network Generalization Prediction with a Known Context Space

In this chapter, we address the challenge of Network Generalization Prediction (NGP), i.e., predicting NN performance in a novel operating domain from a fixed test set. We perform NGP for a robot manipulation task where the environment context is described by a small set of known, observable context features. In the following chapters, we propose methods to identify the environment context either from interpretable metadata, see Chapter 4, or from the NN embedding space, see Chapter 5. To the best of our knowledge, NGP is a previously unaddressed problem. Additionally, we distinguish between failures that do not violate safety constraints, which we denote task failures, and failures that violate safety constraints (whether or not the task is completed), which we denote harmful failures. We denote the probability of task success without causing harm as ML Dependability. The contents of this chapter are as follows:

1. We define ML Dependability[1] as the probability of completing a task without harm. We define Task Undependability and Harmful Undependability to distinguish failures by the consequences: task failures do not cause harm, harmful failures cause harm.

2. We propose a NGP algorithm to predict the NN performance in novel operating conditions by *re-weighting* known test results with knowledge of the novel operating condition probabilities.

3. We accurately predict the ML Dependability, Task Undependability, and Harmful Undependability of a NN trained to perform a simulated robot manipulation task in novel operating conditions using test results.

3.1 Methods

3.1.1 Machine Learning Dependability

We consider the performance of a trained, deterministic NN, f, performing a safety critical task; see Table 3.1 for the notation used in this chapter. A finite test set, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, is available where $\mathbf{x}_i$ is one test example and $\mathbf{y} = \{y_i\}_{i=1}^{N}$ indicates the test labels. A NN may be used iteratively within one test example, e.g., a controller moving a robot incrementally towards a goal, or used once, e.g., a classifier labelling a sensor reading as valid or faulty. For each test example, the NN attempts to complete a task without causing harm. The outcome of deploying a NN in an example is the observed behavior mode. We define three behavior modes: success, task failure, and harmful failure. A NN

[1] This is distinct from software Dependability defined in [60].

Table 3.1: Notation

f	the trained Neural NN
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$	the finite test set
$\mathbf{y} = \{y_i\}_{i=1}^N$	the labels for the finite test set
$f(\mathbf{x})$	the NN prediction for $\mathbf{x}$
$s(f(\mathbf{x}), y)$	success indicator for f in example $\mathbf{x}$
$t(f(\mathbf{x}), y)$	task failure indicator for f in example $\mathbf{x}$
$h(f(\mathbf{x}), y)$	harmful failure indicator for f in example $\mathbf{x}$
$\hat{\mathbf{X}}$	the operating domain
$p_{\mathbf{X}}(\mathbf{C}), p_{\hat{X}}(\mathbf{C})$	the probability distribution describing all possible examples during testing, operation (respectively)
$D_{\hat{\mathbf{X}}}(f)$	the dependability of f in conditions $\hat{\mathbf{X}}$
$T_{\hat{\mathbf{X}}}(f)$	the task undependability of f in conditions $\hat{\mathbf{X}}$
$H_{\hat{\mathbf{X}}}(f)$	the harmful undependability of f in conditions $\hat{\mathbf{X}}$
v	the obstacle velocity [inches/second] in the robot simulation experiments
τ	the obstacle start time [seconds] in the robot simulation experiments
z	the robot goal position [inches] in the robot simulation experiments

is successful if it accomplished the task without causing harm. A task failure occurs when the NN failed to complete the specified task but did not cause harm. Any example where the NN caused harm is labeled a harmful failure, whether or not the task was completed. We propose three indicator functions to identify the behavior modes: $s(f(\mathbf{x}), y)$ to indicate success, $t(f(\mathbf{x}), y)$ to indicate task failure, and $h(f(\mathbf{x}), y)$ to indicate harmful failure.

The environment context or context space of the test set is denoted by $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^N$ where $\mathbf{c}_i$ describes the environment context for x_i. Note that $\mathbf{C}$ can include multiple context features and in general $\mathbf{c}$ is a vector. The input

space of a NN is defined as the information the NN observes. The input may include context features, but need not include context features.

The probability of different contexts within the test set is described as $p_{\mathbf{X}}(\mathbf{C})$. Let $\hat{\mathbf{X}}$ indicate data from a novel operating domain, but we do not assume that data $\hat{\mathbf{X}}$ is available. Instead, information about the context of the novel operating domain, $p_{\hat{\mathbf{X}}}(\mathbf{C})$, is available. As a motivating example, imagine a NN detecting pedestrians for an autonomous vehicle. The NN is trained and tested in Palo Alto but will operate in Seattle. Information, e.g., weather patterns, population statistics, can be used to estimate the probability of different examples during testing and operation without recording or labelling data in the operating conditions.

We define Machine Learning Dependability as the probability that a model will succeed when used in the specified operating domain. We aim to estimate $D_{\hat{\mathbf{X}}}(f)$: the ML Dependability of model f deployed under the operating conditions described by $p_{\hat{\mathbf{X}}}(\mathbf{C})$, where $p_{\hat{\mathbf{X}}}(\mathbf{C}) \neq p_{\mathbf{X}}(\mathbf{C})$. Note that the ML Dependability of f under testing conditions, $D_{\mathbf{X}}(f)$, is equal to the NN accuracy or the fraction of successful tests: $D_{\mathbf{X}}(f) = \frac{1}{N}\sum_{i=1}^{N} s(f(\mathbf{x}_i), y_i)$.

For this analysis, it is assumed that $\mathbf{C}$ is numerical, that $p_{\mathbf{X}}(\mathbf{C})$ and $p_{\hat{\mathbf{X}}}(\mathbf{C})$ are known, and that while $p_{\hat{\mathbf{X}}}(\mathbf{C}) \neq p_{\mathbf{X}}(\mathbf{C})$, the test set covers all contexts $\mathbf{c}$ that are possible in $p_{\hat{\mathbf{X}}}(\mathbf{C})$.

3.1.2 Derivation

3.1.2.1 Discrete-Bounded Context Space

To perform NGP from a finite test set, we relate the test examples, $\mathbf{X}$, and their observed behavior modes to the environment context, $\mathbf{C}$. To begin, we assume $\mathbf{C}$ is discrete with finite M possible values, $\dot{\cup}_{m=1}^{M} \mathbf{c}_m = \mathbf{C}$. Let $\mathbb{I}(a, b)$ be an indicator function that is equal to 1 if $a = b$ and 0 otherwise. The likelihood of a given context $\mathbf{c}$ in the test set can be computed as

$$p_{\mathbf{X}}(\mathbf{c}_m) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\mathbf{c}_i, \mathbf{c}_m) \tag{3.1}$$

where the probability distribution describing examples during testing is

$$p_{\mathbf{X}}(\mathbf{C}) = \{p_{\mathbf{X}}(\mathbf{c}_m)\}_{m=1}^{M} \tag{3.2}$$

The probability distribution describing examples during operation is

$$p_{\hat{\mathbf{X}}}(\mathbf{C}) = \{p_{\hat{\mathbf{X}}}(\mathbf{c}_m)\}_{m=1}^{M} \tag{3.3}$$

Recall that $p_{\hat{\mathbf{X}}}(\mathbf{C})$ can be estimated without labeled operating data using prior knowledge and available statistics about the operating domain.

The ML Dependability of f operating in conditions $\hat{\mathbf{X}}$ is defined as the probability that model f succeeds when deployed in a context $\mathbf{c}$ randomly sampled from the operating conditions $\mathbf{c} \sim p_{\hat{\mathbf{X}}}(\mathbf{C})$.

$$D_{\hat{\mathbf{X}}}(f) = E[s(f(\mathbf{x}), y|\mathbf{c})], \quad \mathbf{c} \sim p_{\hat{\mathbf{X}}}(\mathbf{C}) \tag{3.4}$$

To simplify notation, let $s(\mathbf{c})$ denote the expected probability of success in

context $\mathbf{c}$.

$$s(\mathbf{c}) = \mathbb{E}[s(f(\mathbf{X}), \mathbf{y}|\mathbf{c})] = \frac{\sum_{i=1}^{N} s(f(\mathbf{x}_i), y_i)\mathbb{I}(\mathbf{c}_i, \mathbf{c})}{\sum_{i=1}^{N} \mathbb{I}(\mathbf{c}_i, \mathbf{c})} \tag{3.5}$$

$D_{\hat{\mathbf{X}}}(f)$ can equivalently be written as:

$$D_{\hat{\mathbf{X}}}(f) = \sum_{m=1}^{M} p_{\hat{\mathbf{X}}}(\mathbf{c}_m) s(\mathbf{c}_m) \tag{3.6}$$

$p_{\hat{\mathbf{X}}}(\mathbf{C})$ is known; $s(\mathbf{c})$ must be evaluated via testing. If the context subspace of the NN is truly discrete and $M < \infty$, then the NN can be exhaustively tested with M tests. (Note, if M is finite but large it may be infeasible to exhaustively test the NN. This case may be treated as discrete-unbounded.) In most applications, the context subspace is discrete-unbounded or continuous so the NN cannot be tested exhaustively.

3.1.2.2 Discrete-Unbounded or Continuous Context Space

We approximate discrete-unbounded or continuous $\mathbf{C}$ as discrete-bounded by partitioning $\mathbf{C}$ into M partitions, with $M < \infty$. Let the m^{th} partition be defined as the contiguous region r_m of $\mathbf{C}$, such that $\dot{\cup}_{m=1}^{M} r_m = \mathbf{C}$. The reader is reminded that N test examples are drawn from $p_{\mathbf{X}}(\mathbf{C})$ as $\{\mathbf{x}_i\}_{i=1}^{N}$. N_m examples lie in each partition where $\{\mathbf{x}_i^m\}_{i=1}^{N_m} \in r_m$ denotes the examples in partition m. We require the partitions are defined so that at least one test example lies within each partition, $N_m > 0, \forall m \in [0, M]$. $p_{\hat{\mathbf{X}}}(\mathbf{C})$ is equivalently described by:

$$p_{\hat{\mathbf{X}}}(\mathbf{C}) = \{p_{\hat{\mathbf{X}}}(r_m)\}_{m=1}^{M} \tag{3.7}$$

where $p_{\hat{\mathbf{X}}}(r_m)$ is computed as: $p_{\hat{\mathbf{X}}}(r_m) = \sum_{\mathbf{c} \in r_m} p_{\hat{\mathbf{X}}}(\mathbf{c})$ for discrete-unbounded contexts, or $p_{\hat{\mathbf{X}}}(r_m) = \int_{r_m} p_{\hat{\mathbf{X}}}(\mathbf{c}) d\mathbf{c}$ for continuous contexts. $s(r_m)$ can be estimated as:

$$s(r_m) \approx \frac{\sum_{i=1}^{N_m} s(f(\mathbf{x}_i), y_i)}{N_m} \tag{3.8}$$

The overall ML Dependability can now be approximated as:

$$D_{\hat{\mathbf{X}}}(f) \approx \sum_{m=1}^{M} p_{\hat{\mathbf{X}}}(r_m) * s(r_m) \tag{3.9}$$

3.1.2.3 Estimating Undependability

In a similar manner, we can estimate the undependability of the model f in the operating conditions $\hat{\mathbf{X}}$. $t(f(\mathbf{x}), y) = 1$ when the task is not completed but no harm is done, and $t(f(\mathbf{x}), y) = 0$ otherwise. The Task Undependability, $T_{\hat{\mathbf{X}}}(f)$, is the probability that the model will fail to complete the desired task without causing harm in conditions $\hat{\mathbf{X}}$. We can compute the Task Undependability as:

$$T_{\hat{\mathbf{X}}}(f) = E[t(f(\mathbf{x}), y|\mathbf{c})], \quad \mathbf{c} \sim p_{\hat{\mathbf{X}}}(\mathbf{C}) \tag{3.10}$$

Let $t(r_m)$ denote the expected value of $t(f(\mathbf{x}), y|\mathbf{c})$ in r_m. $T_{\hat{\mathbf{X}}}(f)$ is approximated by:

$$T_{\hat{\mathbf{X}}}(f) \approx \sum_{m=1}^{M} p_{\hat{\mathbf{X}}}(r_m) * t(r_m) \tag{3.11}$$

Similarly, $h(f(\mathbf{x}), y) = 1$ in the event of a harmful failure, and is zero otherwise. The Harmful Undependability of the model, $H_{\hat{\mathbf{X}}}(f)$, is the probability that the model will cause harm when operated in conditions $\hat{\mathbf{X}}$, whether or

not the task is completed. The Harmful Undependability is computed as:

$$H_{\hat{\mathbf{X}}}(f) = E[h(f(\mathbf{x}), y|\mathbf{c})], \quad \mathbf{c} \sim p_{\hat{\mathbf{X}}}(\mathbf{C}) \tag{3.12}$$

Let $h(r_m)$ denote the expected value of $h(f(\mathbf{x}), y|\mathbf{c})$ in r_m. $H_{\hat{\mathbf{X}}}(f)$ is approximated by:

$$H_{\hat{\mathbf{X}}}(f) \approx \sum_{m=1}^{M} p_{\hat{\mathbf{X}}}(r_m) * h(r_m) \tag{3.13}$$

Note that success, task failure, and harmful failure are mutually exclusive, so $D_{\hat{\mathbf{X}}}(f) + T_{\hat{\mathbf{X}}}(f) + H_{\hat{\mathbf{X}}}(f) = 1$.

3.2 Experiments

We evaluated the performance of a NN agent trained via Reinforcement Learning to move a simulated robot in the presence of an obstacle that moves at a constant velocity, v, starting at time τ. The obstacle moves from right to left in the scene with its bottom edge 25 inches from the robot base. The robot's task is to reach or exceed a goal position, z, while avoiding the obstacle, see Figure 3.1. The context space, $\mathbf{C}$, is defined as $v \in [0, 10]$ inches/second, $\tau \in [0, 10]$ seconds, and $z \in [0, 50]$ inches. $\mathbf{C}$ is bounded, continuous, and fully observed. The robot starts at 0 inches and is constrained to be within [0, 50] inches[2]. The simulations last 100 seconds and the NN moves the robot forward 5 inches or back 5 inches every second. The robot moves for the entire 100 second simulation, even after the goal position is reached. A simulation only terminates before 100 seconds if the robot collides with the obstacle.

[2]If the robot tries to move outside this region, the position is clipped. There is no penalty for trying to move outside the valid region.

To succeed, the robot must reach or exceed the goal position before the end of the simulation and avoid the obstacle for the entire simulation. A simulation is a task failure if the robot does not reach the goal position but avoids collision with the obstacle. Any simulation where the robot collides with the obstacle is a harmful failure. In the following results, the behavior modes are denoted with the following colors: success is indicated with green, task failure with blue, and harmful failure with pink.

The NN consists of two linear layers separated by a Rectified Linear Unit (ReLU) and is trained using a modified version of the PyTorch Q-Learning tutorial [61]. Each second, the NN observes the position of the obstacle, the position of the robot, the speed of the obstacle, and the robot goal. Timing information is not input to the NN. Zero-mean Gaussian noise with a standard deviation of 0.1, 0.1, 0.5 for v, τ, and z, respectively, is added to the inputs to simulate sensor noise. The reward function for the NN was designed so reaching the goal results in a reward of 30 points and colliding with the obstacle results in a penalty of -50 points. Before reaching the goal position, the NN receives a small reward of 5 points for moving towards the goal or a penalty of -5 points for moving away from the goal. Before the obstacle has passed the robot, the NN receives a reward of 2 points for each time step it is below the obstacle and a penalty of -2 points each time step it is in the path of the obstacle. The point values for reaching the goal (+30 points) and collision (-50 points) were chosen to prioritize safety over task completion. Likewise, the intermediate rewards were chosen so that moving towards the goal ($\pm$5 points) was prioritized above a potential, future collision ($\pm$2 points).

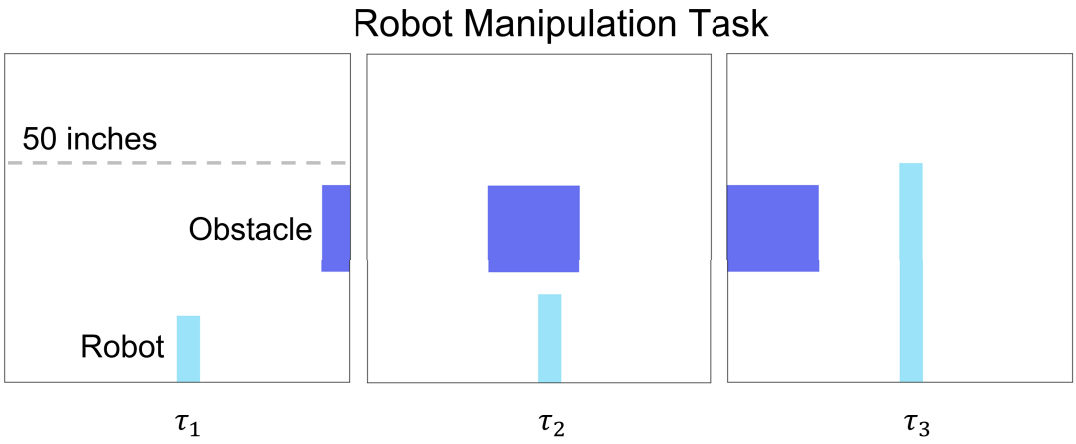

Figure 3.1: The simulated robot manipulation task. To succeed, the robot must avoid the obstacle, which moves at a constant velocity v from right to left, starting at time τ, and reach or exceed a goal location, z, between 0 and 50 inches. τ_1: the obstacle has started moving. τ_2: the robot is avoiding collision with the obstacle. τ_3: the robot has successfully reached and/or exceeded its goal position without colliding with the obstacle.

3.2.1 Performance during Testing

C is bounded and continuous. We sample 100,000 test examples uniformly from **C**:

$$P_\tau(X) : v \sim U(0,10), \tau \sim U(0,10), z \sim U(0,50)$$

where $U(a,b)$ indicates a uniform probability distribution from a to b. We deployed the trained NN in each test example to evaluate the NN performance. The NN had an ML Dependability of **90.35%**, a Task Undependability of **4.18%**, and a Harmful Undependability of **5.47%**. See Figure 3.2 for a plot of observed failures by test example.

Task failures (shown in blue in Figure 3.2) occurred when the obstacle speed was less than or equal to 0.80 inches/second. Inspection revealed that the NN learned to wait for the obstacle to pass before moving forward. In

many cases the robot moved as far forward as it could, exceeding the input robot goal. When the obstacle moved very slowly, this strategy did not give the NN enough time to reach the goal. Harmful failures (shown in pink in Figure 3.2) occurred when the robot goal was greater than or equal to 38.47 inches.

We partition each dimension of $\mathbf{C}$ into 10 equal regions to obtain 1,000 voxels in $\mathbf{C}$. v and τ are divided into regions 1 inch/second and 1 second wide (respectively). z is divided into regions 5 inches wide. We use these voxels to predict the model performance in new operating conditions.

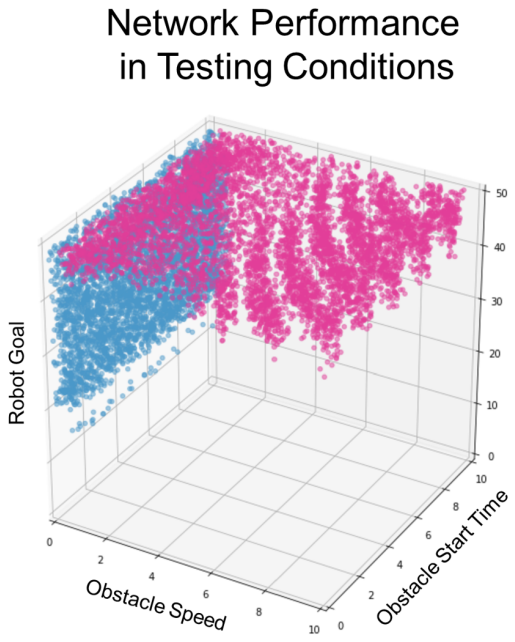

Figure 3.2: The observed failures during testing, best viewed in color. Blue indicates a task failure. Pink indicates a harmful failure. The task failures (along the left 'wall' of the figure) occurred when the obstacle speed was less than or equal to 0.80 inches/second. The harmful failures (along the 'ceiling' of the figure) occurred when the robot goal was greater than or equal to 38.47 inches.

3.2.2 Predicting Model Performance in Novel Operating Conditions

We demonstrate that our method can predict the performance of a NN when deployed in novel operating domains. We define four novel operating domains in Figure 3.3, left. The harmful failures in testing occurred for robot goals greater than or equal to 38.47 inches. We selected Operating Conditions 1 to simulate safe conditions: $z \in [0, 30]$ inches. Operating Conditions 2 simulate dangerous conditions: $z \in [30, 50]$ inches. We also selected distributions other than uniform (the testing distribution) to make the prediction task more challenging. We selected Operating Conditions 3 to introduce a Gaussian context distribution and focus the obstacle velocity v towards slower speeds

	Obstacle Speed v	Obstacle Start Time τ	Robot Goal z
OC 1	$U(0,10)$	$U(0,10)$	$U(0,30)$
OC 2	$U(0,10)$	$U(0,10)$	$U(30,50)$
OC 3	$\mathcal{N}(3,2^2)$	$U(0,10)$	$U(30,50)$
OC 4	$\mathcal{N}(3,2^2)$	$U(0,10)$	$\mathcal{N}(35,10^2)$

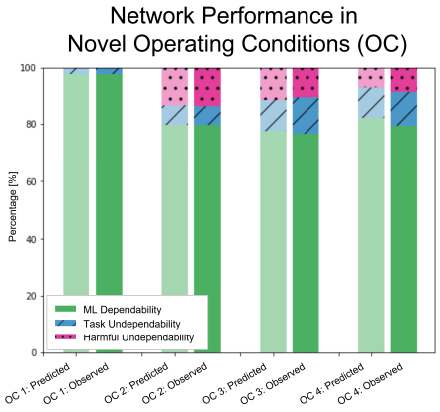

Figure 3.3: Predicted and observed performance of the trained NN in Novel Operating Conditions (OC). **Left:** OC Specification. $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian with a mean of μ and a standard deviation of σ. The sampled examples $x \sim \mathcal{N}(\mu, \sigma^2)$ are clipped to lie within the specified context $\mathbf{C}$. τ is not listed because $\tau \sim U(0, 10)$ for all conditions. **Right:** Predicted and observed performance of the trained NN in OCs. OC predicted performance shown left in light colors. Observed performance shown right in bold colors. ML Dependability $D_{\hat{\mathbf{X}}}(f)$ is shown as solid green, Task Undependability $T_{\hat{\mathbf{X}}}(f)$ is shown as blue hatched, and Harmful Undependability $H_{\hat{\mathbf{X}}}(f)$ is shown as pink dotted bars.

to target the area where task failures occurred. Operating Conditions 4 are the most challenging to predict with Gaussian distributions in v and z focused towards observed task failures and harmful failures.

We used the partitions defined in Subsection 3.2.1 to predict the model performance. To confirm our predictions, 100,000 simulations were run for each set of operating conditions. A comparison of our predicted NN performance with the observed performance is shown in Figure 3.3, above. We accurately predicted the ML Dependability, Task Undependability, and Harmful Undependability within 2% of observed results.

3.3 Discussion

3.3.1 Robot Manipulation Task

We see in Figure 3.2 that the NN performance varies by region in **C**. Partitioning **C** enables these regional variations to emerge when we predict the NN performance in novel operating conditions.

Overall, we accurately predict the performance of the NN in novel operating domains. Across the four proposed operating domains and three performance metrics, the error between the predicted and observed performance percentage was within 2%. The prediction is poorer for operating domains with Gaussian distributions as compared to those with uniform distributions. Finer partitioning of **C** is expected to lead to better predictions and may be necessary as the operating domain distributions become more complex. Both failure modes of the NN, task failure and harmful failure,

relate to timing. The current time step was not an input to the NN; subsequently the NN did not learn to make decisions based on timing. The NN ML Dependability could be improved in the future by adding a timing input.

Task failures occurred when the obstacle speed was less than or equal to 0.80 inches per second. The NN learned to wait for the obstacle to pass the robot before moving past the obstacle, towards the goal. When the obstacle moved slowly this strategy did not give the robot enough time to reach the goal. But, in these examples the NN had ample time to reach the robot goal before the obstacle passed the robot. Adding a timing input could allow the NN to learn more sophisticated timing strategies.

Harmful failures occurred when the robot goal was greater than or equal to 38.47 inches. The NN learned an incorrect trade-off between moving towards the goal and avoiding the obstacle. In most of the examples that were harmful failures in testing, the robot had enough time to avoid collision and reach the goal before the end of the simulation. But the strategy learned by the NN did not time the robot's approach correctly. Interestingly, the reward function was specifically designed to weight safety over task completion: a collision resulted in a penalty of -50 points whereas reaching the goal resulted in a reward of 30 points. While we do not claim that it would be impossible to craft a reward function to perfectly complete this task without harm, this example illustrates that designing a reward function that appropriately weights task requirements and safety constraints is not trivial. See Section **??** for a Safety Functions that was proposed for this task that is an explainable alternative to hand crafting reward functions and guarantee a degree of safety for a NN.

3.3.2 Dependable NNs in Practical Applications

We make several key assumption in our analysis. The implications of these assumptions determine how this work can be applied in practical applications. We assume that $\mathbf{C}$ is numerical. Many applications have numerical contexts such as force sensors and distance sensors, e.g., lidar.

We assume the context is fully observed. A context space may be fully observed in a constrained, industrial setting. But as learned NNs move into unconstrained, dynamic environments, it is not possible to assume the context space is fully observed. In partially observed context spaces, the key change is that we do not assume one context $\mathbf{c}$ maps to exactly one behavior mode. When we modeled discrete-unbounded and continuous fully observed context spaces, we modeled the performance of a NN in a region as $s(r_m) = \mathbb{E}[s(f(\mathbf{X}), \mathbf{y}|\mathbf{c})], \quad \mathbf{c} \sim r_m$. This can be extended in the future to model the distribution of outcomes observed from examples $\mathbf{x} \sim \mathbf{c}$ when the context is only partially observed. The quality of the performance predictions will vary by how well the partially observed context describes the full context.

We assume $p_{\mathbf{X}}(\mathbf{C})$ and $p_{\hat{\mathbf{X}}}(\mathbf{C})$ are known. As stated earlier, $p_{\mathbf{X}}(\mathbf{C})$ and $p_{\hat{\mathbf{X}}}(\mathbf{C})$ can be estimated empirically from statistical data or context knowledge. Lastly, we assume both distributions cover the same CS $\mathbf{X}$ and that the number of test samples in each partition is greater than zero. This assumption requires some care when designing the partitions.

3.4 Conclusions

We define and derive the metrics ML Dependability, Task Undependability, and Harmful Undependability to predict a trained NN's performance in novel operating conditions. We demonstrate that our metrics can predict the performance of a trained NN in novel operating conditions within 2% of observed performance for a simulated robot manipulation task. In Chapter 4 we address the challenge of a partially observed context space and a high dimensional problem where it is not known a priori which context features impact NN performance.

Chapter 4

Identifying the Context Subspace

In this chapter we identify the context subspace, $\mathbf{C}^{s_K}$, from a set of available context features. We rank the context features by how much information they provide about the NN performance; we then select how many context features should be included in the context subspace for the most accurate performance predictions. This renders the NGP approach proposed in Chapter 3 tractable for high complexity problems with partially observed context spaces. Our contributions in this chapter are as follows:

1. We introduce the concept of a context subspace, a low-dimensional space, encoding the context features most informative about the network performance.

2. We propose a greedy feature selection algorithm for identifying the context subspace by 1) ranking the context features by the information they provide about the network loss, and 2) selecting the subspace dimensionality that leads to accurate NGP.

3. We leverage a context subspace for accurate NGP for pedestrian detection in diverse operating domains, with a prediction error from 0.5% to 2% for not safety critical pedestrians (pedestrians not in the road), and a prediction error from 2% to 5% for safety critical pedestrians (pedestrians in the road).

4. We demonstrate that the context subspace identified for the Berkeley Deep Drive Dataset (BDD) can be used to predict pedestrian recall in completely unseen datasets, the JAAD and Cityscapes Datasets, with a prediction bias of 10% or less.

4.1 Feature Selection

Selecting which context features to consider for NGP is a feature selection problem. Feature selection algorithms aim to select a subset of the available features, typically to use the features as input to train a model for a given task. Feature selection algorithms can be classified as filter methods, i.e., features are scored according to their association with the task label, wrapper methods, i.e., features are selected to minimize task error, and embedded methods, i.e., features are selected in the model training process [62]. The Mutual Information [63] is often used in filter methods to measure the information between a given feature and the desired label [64]. As exhaustive feature selection search is typically intractable, greedy feature selection algorithms are often used [65], [66], [67]. Note, greedy feature selection is related to matching pursuit in the sparse approximation literature [68] and has applications in compressed sensing [69].

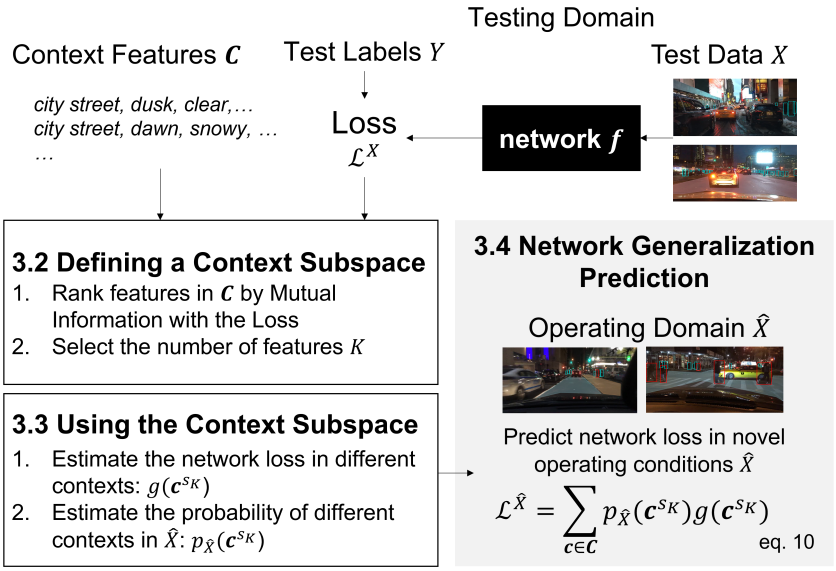

Figure 4.1: Overview of Network Generalization Prediction.

4.2 Methods

4.2.1 Problem Formulation

It is well known that in supervised learning, a network, f, is trained to produce a label, y_i, from data, $\mathbf{x}_i$, and a loss function, $\ell(f(\mathbf{x}_i), y_i)$, is used to drive training. In NGP, we are not training f. Instead, we aim to predict the performance of a fixed network f, trained via supervised learning, when deployed in an operating domain, $\hat{\mathbf{X}}$, that differs from the testing domain, $\mathbf{X}$; see Figure 4.1. The performance of f is measured using test data, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, and test labels, $\mathbf{y} = \{y_i\}_{i=1}^N$, via a loss function $L = \{\ell(f(\mathbf{x}_i), y_i)\}_{i=1}^N$, where the elements of L are assumed to be discrete and bounded, e.g., an object detection flag, whether a safety criteria was satisfied, or a discretized classification error.

$\mathbf{X}$ is described via J context features, $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^N$, where $\mathbf{c}_i$ indicates a J

Notation	
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$	The test set
$\mathbf{y} = \{y_i\}_{i=1}^N$	The test labels
$\hat{\mathbf{X}}$	The operating domain
f	The trained network
$L = \{\ell(f(\mathbf{x}_i), y_i)\}_{i=1}^N$	The test set loss
$\mathbf{C}$	The context features
$\mathbf{c} \in \mathbf{C}$	A context vector
$g(\mathbf{c})$	The expected loss of f in $\mathbf{c}$
$\mathbf{C}^{S_K}$	The context subspace
$p_{\mathbf{X}}(\mathbf{c})$, $p_{\hat{\mathbf{X}}}(\mathbf{c})$	The probability of $\mathbf{c}$ in $\mathbf{X}$, $\hat{\mathbf{X}}$
$\mathcal{L}^{\mathbf{X}}$, $\mathcal{L}^{\hat{\mathbf{X}}}$	The observed loss in $\mathbf{X}$, $\hat{\mathbf{X}}$

Table 4.1: Notation.

dimensional context vector associated with $\mathbf{x}_i$. $p_{\mathbf{X}}(\mathbf{c})$ denotes the probability of encountering $\mathbf{c}$ in $\mathbf{X}$. $\hat{\mathbf{X}}$ is described by the probability of encountering $\mathbf{c}$ in $\hat{\mathbf{X}}$, $p_{\hat{\mathbf{X}}}(\mathbf{c})$. In many practical applications, the likelihood of encountering a context may be known without annotated data, e.g., there is a 25% chance of snow in Boston, etc. Note, labeled test data from $\hat{\mathbf{X}}$ is not required. We assume that while the distribution of contexts shifts between the testing and operating domains, i.e., $p_{\mathbf{X}}(\mathbf{c}) \neq p_{\hat{\mathbf{X}}}(\mathbf{c})$, the expected network performance in context $\mathbf{c}$ is stable for both the testing and operating domains. Table 4.1 describes the Notation used in the Methods Section.

As is typical, we approximate the posterior expected loss in $\mathbf{X}$, $\mathcal{L}^{\mathbf{X}}$, using the empirical loss:

$$\mathcal{L}^{\mathbf{X}} = E[\ell(f(\mathbf{X}), \mathbf{y})] = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}_i), y_i) \tag{4.1}$$

We define $g(\mathbf{c}) = E[\ell(f(\mathbf{X}), \mathbf{y}|\mathbf{c})]$. Let $\mathbb{I}(\mathbf{a}, \mathbf{b})$ be an indicator function that is

equal to 1 if $\mathbf{a} = \mathbf{b}$ and 0 otherwise. $g(\mathbf{c})$ can be computed as:

$$g(\mathbf{c}) = \frac{\sum_{i=1}^{N} \mathbb{I}(\mathbf{c}_i, \mathbf{c}) * \ell(f(\mathbf{x}_i), y_i)}{\sum_{i=1}^{N} \mathbb{I}(\mathbf{c}_i, \mathbf{c})} \tag{4.2}$$

$\mathcal{L}^X$ can equivalently be computed as:

$$\mathcal{L}^{\mathbf{X}} = \sum_{\mathbf{c} \in \mathbf{C}} p_{\mathbf{X}}(\mathbf{c}) g(\mathbf{c}) \tag{4.3}$$

Likewise, we can now express the NGP, $\mathcal{L}^{\hat{\mathbf{X}}}$, as:

$$\mathcal{L}^{\hat{\mathbf{X}}} = \sum_{\mathbf{c} \in \mathbf{C}} p_{\hat{\mathbf{X}}}(\mathbf{c}) g(\mathbf{c}) \tag{4.4}$$

This formulation holds theoretically for any number of context features J. However, as J grows linearly, computing Eqn. 4.4 requires exponentially more test samples to cover every possible $\mathbf{c} \in \mathbf{C}$. Thus, we introduce the context subspace, $\mathbf{C}^{S_K}$, a low-dimensional space, encoding the context features most informative about the network performance.

4.2.2 Defining a Context Subspace

We are interested in selecting the K context features that provide the most information about the network loss, to include these features in $\mathbf{C}^{S_K}$. Let $S_K = \{s_k\}_{k=1}^{K}$ be the indices of context features of interest and $\mathbf{C}^{S_K} = \{C^{s_k}\}_{k=1}^{K}$, where $C^{s_k} = \{c_i^{s_k}\}_{i=1}^{N}$ are the annotated attributes for each example in the test set for context feature s_k. To select the context features to include in $\mathbf{C}^{S_K}$, we 1) rank the context features by how much information they provide about the network loss, 2) select the $\mathbf{C}^{S_K}$ dimensionality K to enable accurate NGP.

4.2.2.1 Ranking Context Features

Recall, the Mutual Information is often used to rank features in filter feature selection algorithms and is computed as $I(L, C^j)$ for loss L and context feature C^j:

$$I(L, C^j) = \sum_{\ell \in L} \sum_{c \in C^j} p(\ell, c) log(\frac{p(\ell, c)}{p(\ell)p(c)}) \tag{4.5}$$

where $p(\ell, c)$ indicates the joint probability of ℓ and c, and $p(\ell)$ and $p(c)$ indicate the marginal probabilities for ℓ and c, respectively. The Interaction Information is a generalization of the Mutual Information to K features. The Interaction Information between L and the context features $C^{s_1}, ..., C^{s_K}$ is defined as:

$$I(L, C^{s_1}, ..., C^{s_K}) \quad = \quad I(L, C^{s_1}, ..., C^{s_{K-1}}) \; - \; I(L, C^{s_1}, ..., C^{s_{K-1}} | C^{s_K}) \tag{4.6}$$

For two features, this becomes:

$$I(L, C^{s_1}, C^{s_2}) = I(L, C^{s_2}) - I(L, C^{s_2} | C^{s_1}) \tag{4.7}$$

Where $I(L, C^{s_2} | C^{s_1})$ can be computed as:

$$I(L, C^{s_2} | C^{s_1}) \quad = \quad \sum_{\ell \in L} \sum_{c_2 \in C_2} \sum_{c_1 \in C_1} p(\ell, c_2, c_1) \; \times \; log \left(\frac{p(\ell, c_2, c_1)}{p(\ell, c_1) p(c_2, c_1)} \right) \tag{4.8}$$

The computational complexity of $I(L, C^{s_1}, ..., C^{s_K})$ grows combinatorially with K. We are interested in ranking the context features by the Interaction Information, but computing the exact Interaction Information becomes intractable as K grows. To make computation tractable, we propose $\Delta I(L, C^{s_1}, ..., C^{s_K})$ to

Algorithm 1 Greedy ΔI Context Selection

1: $S_K = \{\}$
2: **for** $k = 1 : K$ **do**
3: $\quad s_k^* \leftarrow argmax_j[I(C^j, L) - \sum_{s_k \in S_K} I(C^j, C^{s_k})]$
4: $\quad \forall j \in J \setminus S_K$
5: $\quad S_K = S_K \cup s_k^*$
6: **end for**
7: $g(\mathbf{c}^{S_K}) = E[\ell(f(\mathbf{X}), \mathbf{y}|\mathbf{c}^{S_K})]$

approximate how much more information including context feature C^{s_K} in the context subspace provides about L.

$$\Delta I(L, C^{s_1}, ..., C^{s_K}) = I(L, C^{s_K}) - \sum_{k=1}^{K-1} I(C^{s_k}, C^{s_K}) \tag{4.9}$$

Intuitively, $\Delta I(L, C^{s_1}, ..., C^{s_K})$ subtracts the redundant information in C^{s_K}, that is $\sum_{k=1}^{K-1} I(C^{s_k}, C^{s_K})$, from the information it provides about the loss, $I(L, C^{s_K})$. Note that the computational complexity of computing $\Delta I(L, C^{s_1}, ..., C^{s_K})$ grows linearly with K. Like the Interaction Information, $\Delta I(L, C^{s_1}, ..., C^{s_K})$ can be positive or negative. In Section 4.2.5, we show that for independent features in the context subspace, $\Delta I(L, C^{s_1}, C^{s_2})$ approaches $I(L, C^{s_1}, C^{s_2})$ as C^{s_2} approaches perfect information on L. We propose a greedy algorithm to iteratively select the K most informative features from the context; see Algorithm 1.

4.2.2.2 Selecting the Context Subspace Dimensionality

Selecting the number of features, K, to include in $\mathbf{C}^{S_K}$ is not trivial. Recall, the context features are discretized. If each feature has n unique values, the total number of unique contexts is n^K: as the number of context features, K, increases, the number of unique contexts to describe with the test data

increases exponentially. Including more features can lead to a more descriptive $\mathbf{C}^{S_K}$ but can also lead to many untested contexts in $\mathbf{C}^{S_K}$. To select K, we compute the expected prediction error for a given subspace dimensionality, ϵ_K. Using the K most informative context features, $g(\mathbf{c}^{S_K}) = E[\ell(f(X), \mathbf{y}|\mathbf{c}^{S_K})]$ can be computed according to Eqn. 4.2. where $\mathbf{c}^{S_K}$ is a K dimensional feature vector in $\mathbf{C}^{S_K}$. We iteratively compute the prediction error within the test set, ϵ_K, to estimate the expected prediction error, $\tilde{\epsilon}_K$, see Algorithm 2. First, we randomly partition the test set into a fit set and a val set: $\mathbf{X}^{fit}$, $\mathbf{y}^{fit}$, $\mathbf{C}^{fit}$ with N^{fit} samples and $\mathbf{X}^{val}$, $\mathbf{y}^{val}$, $\mathbf{C}^{val}$ with N^{val} samples respectively. We estimate $g^{fit}(\mathbf{c}^{S_K})$ using the fit set. We compute the observed loss from the val set, $\mathcal{L}^{val}$. Let $p^{val}(\mathbf{c}^{S_K})$ indicate the probability of encountering context $\mathbf{c}^{S_K}$ in $\mathbf{C}^{val}$. The prediction error, ϵ_K, is the difference between the observed validation loss, $\mathcal{L}^{val}$, and the predicted validation loss using $g^{fit}(\mathbf{c}^{S_K})$. This procedure can be iterated multiple times, and the subsequent ϵ_K's averaged, to estimate the expected prediction error, $\tilde{\epsilon}_K$, for different random fit and val partitions of the test set. We select the K that minimizes $\tilde{\epsilon}_K$.

$\tilde{\epsilon}_K$ measures the expected prediction error within X. When the context is informative about the loss, we expect $\tilde{\epsilon}_K$ to decrease as K increases until an optimal K^* is reached, then $\tilde{\epsilon}_K$ will begin to rise as K increases and there are many untested contexts. If $\tilde{\epsilon}_K$ is flat or increasing as K increases, it indicates that the context features available are not informative about the loss.

After we have ranked the context features and selected the number of features to include in the subspace, we can form $\mathbf{C}^{S_K}$. The K most informative context features form the axes of the subspace. Recall, we assumed the context

Algorithm 2 Context Subspace Dimensionality Selection

$\tilde{\epsilon}_K = \{\}$
for $K = 1 : J$ **do**
  $\epsilon_K s = []$
  for *iteration* **do**
    split test set into fit and val set
    $g^{fit}(\mathbf{c}^{S_K}) = E[\ell(f(\mathbf{X}^{fit}), \mathbf{y}^{fit} | \mathbf{c}^{S_K})]$
    $\mathcal{L}^{val} = \frac{1}{N^{val}} \sum_{i=1}^{N^{val}} \ell(f(\mathbf{x}_i^{val}), y_i^{val})$
    $\epsilon_K = |\mathcal{L}^{val} - \sum_{\mathbf{c}^{S_K} \in \mathbf{C}^{S_K}} p^{val}(\mathbf{c}^{S_K}) g_k^{fit}(\mathbf{c}^{S_K})|$
    $\epsilon_K s.append(\epsilon_K)$
  end for
  $\tilde{\epsilon}_K = mean(\epsilon_K s)$
end for
$K \leftarrow argmin_K \quad \tilde{\epsilon}_K$

features are categorical or numerical and discrete, this yields a finite set of context partitions, $\mathbf{c}^{S_K} \in \mathbf{C}^{S_K}$.

4.2.3 Using the Context Subspace

We use $\mathbf{C}^{S_K}$ to describe the expected network loss in different contexts, $g(\mathbf{c}^{S_K})$, and to describe the probability of encountering a context in the operating domain, $p_{\hat{\mathbf{X}}}(\mathbf{c}^{S_K})$. We can compute $g(\mathbf{c}^{S_K})$ using Eqn. 4.2, note we use the entire test set to compute $g(\mathbf{c}^{S_K})$ once we have selected the subspace dimensionality K. Recall, we do not assume to have labeled test data in $\hat{\mathbf{X}}$, but we do assume to know $p_{\hat{\mathbf{X}}}(\mathbf{c}^{S_K})$. Individual context feature probabilities can be multiplied to obtain a joint probability distribution if the context feature probabilities are assumed to be independent.

4.2.4 Network Generalization Prediction

We can now perform NGP, where $\mathcal{L}^{\hat{\mathbf{X}}}$ is the predicted loss in $\hat{\mathbf{X}}$:

$$\mathcal{L}^{\hat{\mathbf{X}}} = \sum_{\mathbf{c}^{S_K} \in \mathbf{C}^{S_K}} p_{\hat{\mathbf{X}}}(\mathbf{c}^{S_K}) g(\mathbf{c}^{S_K}) \tag{4.10}$$

Recall, we selected a small number of informative context features so that it would be practical to describe the unique contexts $\mathbf{c}^{S_K} \in \mathbf{C}^{S_K}$, but there may be untested contexts in $\mathbf{C}^{S_K}$. For conservative predictions, we assume the maximum loss in untested contexts. The maximum loss may correspond to a binary failure or a large expected error. Leveraging $\mathbf{C}^{S_K}$ renders NGP practical for interestingly complex applications, like perception for autonomous vehicles. The interested reader is directed to Section 4.2.5 to examine the difference between ΔI and I. All other readers may proceed to the Experiments in Section 4.3.

4.2.5 Comparing ΔI and I

We propose $\Delta I(L, C^{s_1}, ..., C^{s_K})$ to approximate the Interaction Information between K context features and the network loss L, $I(L, C^{s_1}, ..., C^{s_K})$. The computational complexity of computing $\Delta I(L, C^{s_1}, ..., C^{s_K})$ grows linearly with K, as compared to the computational complexity of computing $I(L, C^{s_1}, ..., C^{s_K})$ which grows combinatorially with K. We investigate the difference between $I(L, C^{s_1}, ..., C^{s_K})$ and $\Delta I(L, C^{s_1}, ..., C^{s_K})$. To simplify the notation, we denote C^{s_1} as C^1 and C^{s_2} as C^2. It is trivial to compute the Mutual Information between the context features and L and select C^1 to be the feature most informative about the loss. We assume C^1 has been selected and we compare $I(L, C^1, C^2)$

and $\Delta I(L, C^1, C^2)$.

$$I(L, C^1, C^2) = I(L, C^2) - I(L, C^2|C^1) \tag{4.11}$$

$$\Delta I(L, C^1, C^2) = I(L, C^2) - I(C^1, C^2) \tag{4.12}$$

The difference between $I(L, C^1, C^2)$ and $\Delta I(L, C^1, C^2)$ is

$$I(L, C^1, C^2) - \Delta I(L, C^1, C^2) = I(C^1, C^2) - I(L, C^2|C^1) \tag{4.13}$$

As we would like the context features in $\mathbf{C}^{S_K}$ to be roughly independent, let us assume that C^1 is not informative of C^2, i.e., $I(C^1, C^2) = 0$.

$$I(L, C^1, C^2) - \Delta I(L, C^1, C^2) = -I(L, C^2|C^1) \tag{4.14}$$

The reader is reminded that the conditional mutual information is computed as:

$$I(L, C^2|C^1) \quad = \quad \sum_{\ell \in L} \sum_{c_1 \in C_1} \sum_{c_2 \in C_2} p(\ell, c_1, c_2) \times log\left(\frac{p(c_1)p(\ell, c_1, c_2)}{p(\ell, c_1)p(c_1, c_2)}\right) \tag{4.15}$$

For simplicity, let us consider the point wise conditional mutual information at ℓ, c_1, and c_2:

$$log\left(\frac{p(c_1)p(\ell, c_1, c_2)}{p(\ell, c_1)p(c_1, c_2)}\right) \tag{4.16}$$

Recall, it was assumed that C^1 and C^2 are independent, thus $p(c_1, c_2) = p(c_1)p(c_2)$. The joint probability $p(\ell, c_1, c_2)$ can also be factored as $\frac{1}{Z}\psi(\ell, c_1)\psi(\ell, c_2)$. Thus the point wise conditional mutual information at ℓ, c_1, and c_2 can be

rewritten as:

$$log\left(\frac{p(c_1)\psi(\ell,c_1)\psi(\ell,c_2)}{Zp(\ell,c_1)p(c_1)p(c_2)}\right) \tag{4.17}$$

The term $p(c_1)$ occurs in the numerator and denominator; therefore, this expression simplified to

$$log\left(\frac{\psi(\ell,c_1)\psi(\ell,c_2)}{Zp(\ell,c_1)p(c_2)}\right) \tag{4.18}$$

Note $\psi(\ell,c_1) \propto p(\ell,c_1)$ and $\psi(\ell,c_2) \propto p(\ell,c_2)$. Thus, the difference between the proposed ΔI and the Interaction Information is proportional to

$$\propto log\left(p(\ell|c_2)\right) \tag{4.19}$$

If we consider only combinations of ℓ and c_2 that exist in the test set, $p(\ell|c_2) > 0$. As the new context feature becomes more informative, $p(\ell|c_2) \to 1$ and the difference $log\left(p(\ell|c_2)\right) \to 0$. This demonstrates that, if the context features are informative about the loss, ΔI is a good approximation of the Interaction Information.

4.3 Experimental Results

4.3.1 Pedestrian Detection Generalization

Perception for autonomous vehicles is an active area of research, and systems that use deep networks to detect and avoid obstacles, like pedestrians, while driving are commercially available. Some of these commercial systems can be used in any driving conditions, at the user's discretion, and the operating domains can vary significantly in terms of the lighting conditions, e.g.,

daytime compared to night, road conditions, e.g., dry roads in clear weather compared to slippery roads in rainy or snowy weather, and obstacle density, e.g., a residential street compared to a restricted access highway. It would be impractical for autonomous vehicle developers to test a perception system in every possible operating domain, but it is also imperative to know whether it is safe to use a perception system in a given operating domain. We perform experiments analogous to an autonomous vehicle developer: we test a fixed network in one testing domain, $\mathbf{X}$, and predict the network's performance in novel operating domains, where the distribution of context features vary significantly from $\mathbf{X}$. Our goal is to accurately predict the observed network performance when the network is used in a novel operating domain, $\hat{\mathbf{X}}$.

We test a pretrained Faster RCNN [70] object detector for pedestrian detection, where the objects detected as *person* are used as pedestrian detections. In our analysis, we consider pedestrians whose ground truth bounding box area is ≥ 300 pixels. We evaluate the network performance at the pedestrian level. Pedestrians correctly detected with an $IoU > 0.5$ and a confidence score > 0.5 are assigned a loss of 0; pedestrians that are not detected are assigned a loss of 1[1]. Pedestrians in images with multiple people are considered independently; images with no pedestrians present are not assigned any loss.

BDD Dataset [25] was recorded across the continental US and includes data from varying times of day (daytime, dawn/dusk, or night), weather conditions (clear, partly cloudy, overcast, rainy, foggy, or snowy), and scene

[1]We are predicting the network's recall. We do not assign a loss for false positive detections; this same methodology can be used to predict network precision if that is of interest. We focus on recall because failing to predict a pedestrian who is truly present in the scene is a higher safety risk than trying to avoid a pedestrian who is not present.

types (city street, residential, or highway). BDD images are of size 720×1280. We use $10,000$ images from the BDD Dataset for testing, denoted the BDD Test Set. We use the remaining $70,000$ images in the BDD Dataset, denoted the BDD Operating Set, to define novel operating domains. The BDD Test Set and BDD Operating Set correspond to the BDD “Validation" and “Train" folds, respectively.

4.3.2 Defining the Context Subspace

We evaluate the network performance at the pedestrian level; therefore, context features are assigned to individual pedestrians. We do not know a priori which pedestrian attributes are informative about the network loss, so we include all available context features. The BDD dataset includes metadata on the image time of day, weather, and scene type. We include the metadata as context features. We also include the image brightness and the pedestrian bounding box brightness. We define the road(s) to be the safety critical (SC) region(s) in the images. Pedestrians in the road are labeled SC, pedestrians outside the road, e.g., on the sidewalk, are labeled not safety critical (NSC). The road is defined using the drivable area annotations. Whether a pedestrian is SC, denoted the safety critical flag, is included as a context feature. To capture information about the obstacle density in the scene, we include the total number of pedestrians, the number of SC pedestrians, and the number of NSC pedestrians in the image as context features.

4.3.2.1 Ranking Context Features

We use Algorithm 1 to rank the context features by how much information they provide about the network loss. When computing the mutual information for a numerical feature with more than 10 unique values, we uniformly partition the feature into 10 discrete bins. Categorical features are labeled discretely with their assigned labels. See Figure 4.2 left for the ΔI computed for the first three iterations of Algorithm 1. The six most informative features were found to be: 1) image brightness, 2) safety critical flag, 3) scene , 4) number SC pedestrians, 5) time of day, and 6) bounding box brightness.

4.3.2.2 Selecting the Context Subspace Dimensionality

To select the number of features to include in the context subspace, we compute $\tilde{\epsilon}_K$ for values of K from 1 to 6. For each dimensionality, K, we compute ϵ_K 50 times by randomly partitioning the test data into 50% for fitting $g(\mathbf{c}^{S_K})$ and 50% for validation. We select the K with the minimum expected prediction error $\tilde{\epsilon}_K$ over the 50 iterations. $K = 3$ was found to be optimal, with an average prediction error of 0.63%; see Figure 4.2 right. We subsequently define the context subspace with three dimensions: 1) image brightness, 2) safety critical flag, and 3) scene.

The image brightness is a continuous feature; we uniformly partition the image brightness into 10 bins. The safety critical flag and the scene type are discrete and categorical features with 2 and 3 possible values, respectively. This results in a context subspace, $\mathbf{C}^{S_K}$, with 60 discrete contexts, $\mathbf{c}^{S_K}$.

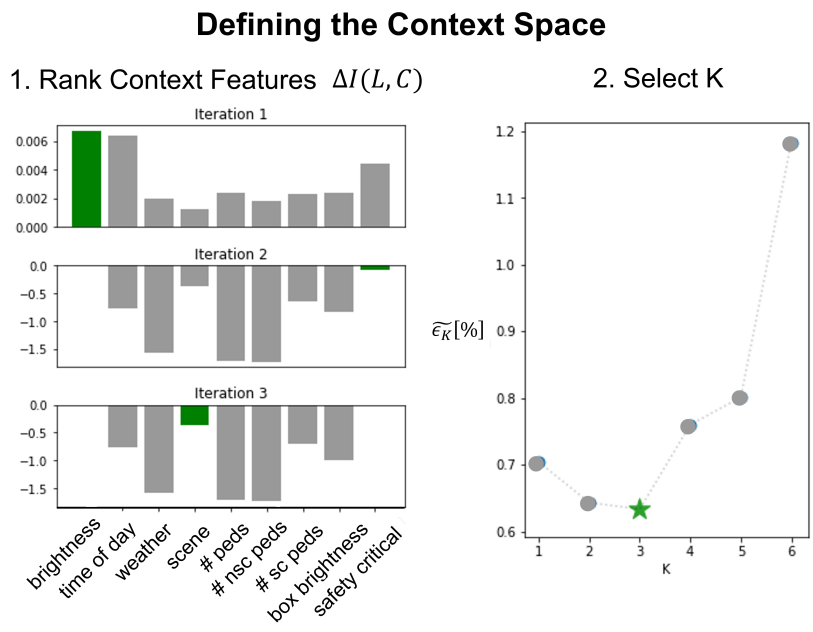

Figure 4.2: Defining the context subspace. 1) Rank Context Features: The $\Delta I(L,C)$ between different context features and the loss in the BDD Test Set for the first three rounds of Algorithm 1. Note that in iteration one, $\Delta I(L,C) = I(L,C)$ so the features' scores are non-negative. 2) Select K: We estimate the expected prediction error for different context subspace dimensionalities, K, and choose the dimensionality with the lowest expected prediction error: in this case, $K = 3$. We form the context subspace with the three most informative context features: brightness, safety critical flag, and the scene type.

4.3.3 Using the Context Subspace

We use $\mathbf{C}^{S_K}$ to estimate the expected network loss in a context, $g(\mathbf{c}^{S_K})$, and to describe the probability of encountering a context in $\hat{X}$, $p_{\hat{X}}(\mathbf{c}^{S_K})$. For all tested contexts, $g(\mathbf{c}^{S_K})$ is computed according to Eqn. 4.2. All untested contexts are assigned an expected loss of 1, i.e., a 100% chance of failing to detect a pedestrian. The BDD Operating Set is used to define four novel operating domains: 1) day, small groups; 2) day, large groups; 3) night, small groups;

Figure 4.3: BDD Novel Operating Domains. We define operating domains based on the time of day and the number of pedestrians in an image. Images with fewer than 5 (N)SC pedestrians fall under small groups. Images with 5 or more (N)SC pedestrians fall under large groups. Sample images from the operating domains are shown. NSC pedestrians are outlined in blue. SC pedestrians are outlined in red. Drivable area is shown in random transparent colors.

and 4) night, large groups; see Figure 4.3. The time of day annotated in the images was used to assign "day" or "night". The NSC and SC pedestrians are considered independently. Pedestrians in images with fewer than 5 (N)SC pedestrians are categorized as small groups; pedestrians in images with 5 or more (N)SC pedestrians are categorized as large groups, i.e., in an image with 15 NSC pedestrians and 2 SC pedestrians, the NSC pedestrians would be labeled 'large group' and the SC pedestrians would be labeled 'small group'. We compute $p_{\hat{X}}(\mathbf{c}^{S_K})$ for each $\hat{X}$ by counting the number of pedestrians that fall into each $\mathbf{c}^{S_K} \in \mathbf{C}^{S_K}$ and dividing by the total number of pedestrians.

4.3.4 Pedestrian Detection Generalization Prediction

We predict the network loss in the novel operating domains defined in 4.3.3 using Eqn. 4.10. Our network loss is equivalent to the fraction of pedestrians

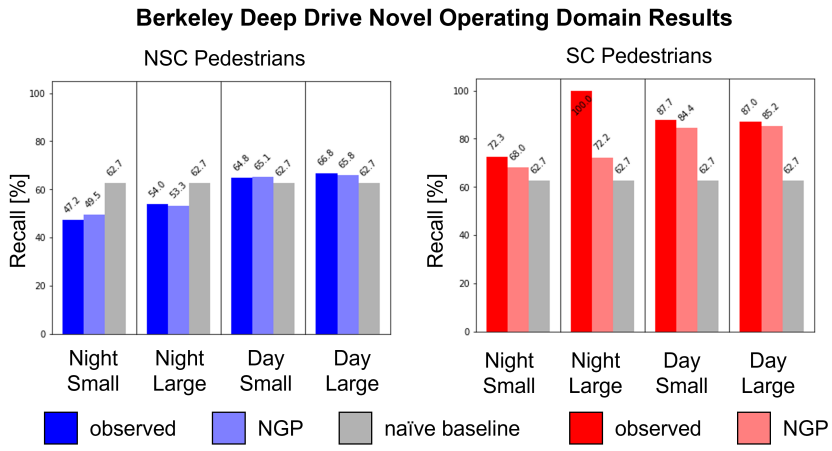

Figure 4.4: Network Generalization Prediction (NGP) Results. NSC pedestrian recall and SC pedestrian recall are shown separately. The observed pedestrian recall for the images of each operating domain are shown in bright blue or red. The NGP predicted recall is shown in light blue or red. The naïve baseline indicates the average recall over all pedestrians in the Test Set. Note that the naïve baseline is the same for every operating domain.

that are not detected by the network; we convert the predictions into the predicted network recall by subtracting the fraction of pedestrians that are not detected from 1; see Figure 4.4. For reference, we include the average percent of pedestrians detected in the Test Set as a naïve baseline. We then pass the BDD Operating Set through the network; the observed recall is computed as the fraction of pedestrians that were correctly detected. Figure 4.4 illustrates that our predictions are accurate with NGP accuracy between 0.5% and 2.5% for NSC pedestrian recall and 2% and 5% for SC pedestrian recall. All the SC predictions underpredict the observed recall; this demonstrates that our predictions are conservative. Note, the only prediction with significant error is for night, large group SC pedestrians. Only one image in the BDD Operating Set falls into this category, so the observed performance is based on minimal

Figure 4.5: Unseen Dataset Novel Operating Domains. Sample images from the unseen datasets. NSC pedestrians outlined in blue. SC pedestrians outlined in red.

data.

4.3.5 Generalization Prediction for Unseen Datasets

As a preliminary study, we investigate whether the context subspace, $\mathbf{C}^{S_K}$, defined using the BDD Test Set and the network loss, $g(\mathbf{c}^{S_K})$, estimated from the BDD Test Set provide information about completely unseen datasets. Unseen datasets include shifts in the context feature distributions, as well as changes in camera parameters and physical setup that are not captured by the test set. As such, we expect predictions for unseen dataset to contain bias, i.e., the prediction error for an unseen dataset will have a consistent non-zero offset. We are interested in determining the magnitude of this prediction bias to evaluate the usefulness of NGP across datasets. We perform NGP for the JAAD Dataset [71], and the Cityscapes Dataset with the gtFine labels [72]; see Figure 4.5 for sample images. For both datasets, the (N)SC pedestrian image brightness distribution is computed from the images.

The JAAD Dataset was recorded in North America and Europe; it includes primarily daytime images from residential and city streets in varying weather conditions. JAAD images are of size 1080×1920. For the JAAD Dataset,

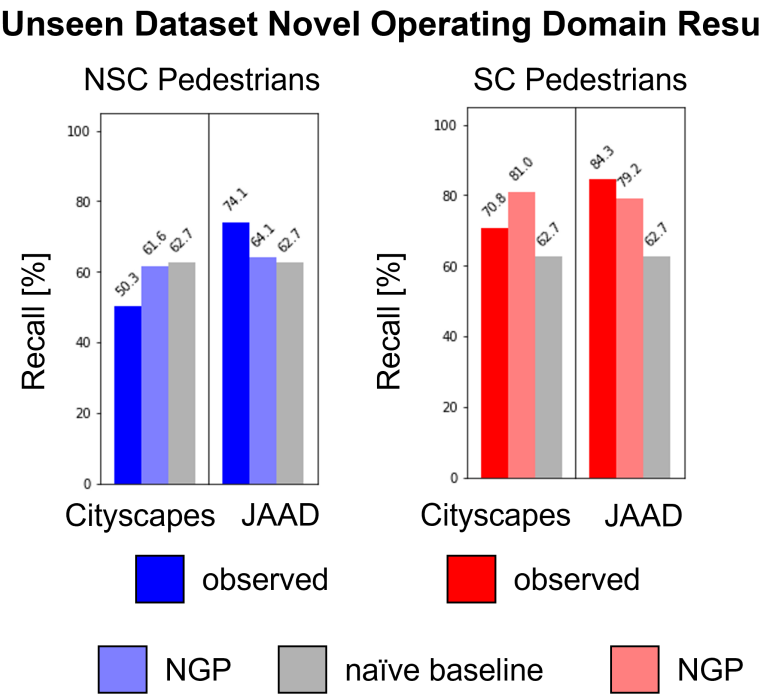

Figure 4.6: Network Generalization Prediction for Unseen Datasets. NSC pedestrian recall and SC pedestrian recall are shown separately. The observed pedestrian recall for the images of each novel dataset is shown in bright blue or red. The NGP predicted recall is shown in light blue or red. The naïve baseline indicates the average recall over all pedestrians in the Test Set. Note that the naïve baseline is the same for both unseen datasets.

we sampled images every three seconds from the videos to limit temporal correspondence between frames; this resulted in 1,031 images. Pedestrians in the road were manually annotated as SC, all others were labeled NSC. Scene annotations are not available for the JAAD dataset. To estimate the probability distribution of scenes, the scene type was annotated for a subset of 100 images, we assume the distribution holds for the entire dataset. The marginal (N)SC image brightness distributions and scene type distribution are multiplied to obtain the joint probability distributions for the JAAD Dataset.

The Cityscapes Dataset contains 3,475 images recorded in 50 cities across Germany in the daytime during fair weather conditions. Cityscapes images are of size 1024×2048. We defined the pedestrian bounding boxes using the outermost edges of the labeled person instance segmentations, and we used the semantic segmentation of the road to define the SC region in the image. For Cityscapes, the scene type is known to be "city street".

We make NGPs for the JAAD and Cityscapes Datasets using $g(\mathbf{c}^{S_K})$, estimated using the BDD Test Set; see Figure 4.6. The prediction bias is approximately 10%, with a minimum prediction error of 5% for SC pedestrian recall in the JAAD Dataset. We underpredict pedestrian recall for the JAAD Dataset and we overpredict pedestrian recall for the Cityscapes Dataset.

4.4 Discussion

We make accurate NGPs for the BDD Operating Set, where the observed recall varies from 47% to 87%. This demonstrates that a fixed test set can be used to predict a network's performance in diverse, novel operating domains. The observed recall for SC pedestrians is about 20% higher than for NSC pedestrians. This makes intuitive sense, as SC pedestrians tend to be central in the image and closer to the vehicle. This is encouraging, because the performance of perception systems for autonomous vehicles will ultimately be determined by how well they detect SC pedestrians and obstacles. However, in the BDD Test Set there are many more examples of NSC pedestrians, 11,169, than SC pedestrians, 484. This leads to more untested contexts for the SC pedestrians, which in turn leads to the slight underprediction of SC recall.

For unseen datasets, we find a NGP bias of 10%; we believe these results are promising and that the results indicate the context subspace identified for one dataset, e.g., one camera setup and one physical setup, can be informative for unseen datasets. Investigating how network performance changes between datasets is a challenge we address in Chapter 5.

NGP can be used to link network behavior in novel operating domains to required levels of performance. The context subspace can be leveraged for quasi-white box testing by testing the network across variations in context features that are known to impact network behavior. The context subspace also makes the network behavior interpretable by elucidating where failure is more likely. In addition to making the NGP tractable, we believe the context subspace can be used during network training to extract features that are robust to changes in the context subspace. The context subspace can also be used for online error monitoring, e.g., an autonomous vehicle could notify the driver if it detects the surrounding scene is a context with subpar expected performance. We believe the context subspace is a tool that can make network performance more interpretable during training, testing, and deployment.

4.5 Conclusions

We propose the task NGP and leverage a context subspace to render NGP tractable with scarce test samples. We identify the context subspace automatically and demonstrate accurate NGP for Faster RCNN used for pedestrian detection in diverse operating domains. We show that the context subspace identified for the BDD Dataset is informative for completely unseen datasets.

We believe that accurate NGP, with an interpretable context subspace, is a step towards bridging the gap between the high performance of deep networks and the verification required for safety critical systems. The context subspace provides actionable information about what context features impact the NN performance; however, annotated context features are not always available. In Chapter 5 we map the NN embedding space and demonstrate that a sparse subset of the NN embedding dimensions can be used for NGP.

Chapter 5

Mapping the Embedding Subspace

In Chapter 4 we identified the context features that impact the NN performance, but labeled context features are not always available, and distributions describing the novel operating data are not always known. In this chapter, we explore the NN embedding space and map regions of the embedding space that are associated with different NN performance. We leverage this embedding map to predict NN performance in a novel operating domain without labeled context features, the first NGP algorithm that makes predictions based solely on how unlabeled operating images map in the NN embedding space. We demonstrate accurate NGP across diverse image classification tasks, i.e., pedestrian classification, melanoma classification, and animal classification, and are able to outperform our NGP based on the context subspace proposed in Chapter 4. This chapter proceeds as follows:

1. We fit a decision tree to the NN embedding space that efficiently maps the manifold of the labeled test data.

2. We extend the NGP algorithm proposed in Chapter 3 to predict NN

performance in a novel operating domain based on how the unlabeled operating domain images map into the NN embedding space.

3. We evaluate our NGP method on pedestrian, melanoma, and animal classification tasks and demonstrate accurate NGP across different NN architectures and classification tasks. Additionally, we show that our NGP method can identify misclassified images when the NN performance is poor.

5.1 Methods

5.1.1 Problem Formulation

We consider a trained, feed-forward NN, f, where $f(x)$ denotes the NN prediction; see Figure 5.1. The layers of f, excluding the final layer, are a feature extractor, denoted ϕ, that projects the input image $\mathbf{x}$ into a D dimensional NN embedding space (embedding space), $\phi(\mathbf{x}) \in R^D$; see Figure 5.1. The NN f is tested with images from an internal test set, i.e., a test set drawn from the same distribution as the training data. The images in the internal test set are denoted $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and are labeled $\mathbf{y} = \{y_i\}_{i=1}^N$. Images from an external operating set $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i\}_{i=1}^M$ are analogous data from a new distribution. However, for the external operating set we assume that labels $\hat{\mathbf{y}}$ are unknown.

We are interested in finding structure in the embedding space that provides information about the NN performance; specifically, we aim to link the embedding space to the NN outcome. The NN outcome, o, can be a function of both the label and the NN loss. Generally, let $\ell(f(\mathbf{x}), y)$ denote the loss

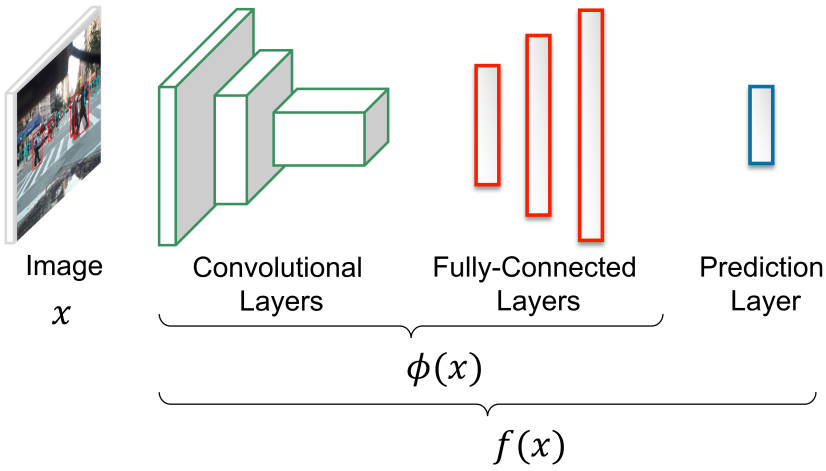

Figure 5.1: Components of a typical feed-forward Deep Neural Network (NN): convolutional layers, fully connected layers, and the prediction layer. The prediction layer is also a fully-connected layer that projects the final embedding, $\phi(\mathbf{x})$, into the prediction dimension.

associated with the NN prediction $f(\mathbf{x})$ and label y. The outcome is denoted by:

$$o(\ell(f(\mathbf{x}), y), y) \triangleq o(f(\mathbf{x}), y) \tag{5.1}$$

For simplicity, we use the notation $o(f(\mathbf{x}), y)$ to denote the outcome. Depending on the task, the outcome of interest could be determined by the loss, e.g., success or failure, or by the loss and the label, e.g., for melanoma classification, in addition to modeling success and failure we may want to model misclassifying a malignant image separately from misclassifying a benign image because misclassifying a malignant image is dangerous for the patient. In the experiments in Section 5.2 we examine binary classification where there are four possible outcomes, i.e., true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

5.1.2 Decision Tree in Embedding Space

The internal test set embeddings, $\phi(\mathbf{X})$, lie on some manifold in the high-dimensional embedding space, see Figure 5.2 Test (1). Decision trees are able to identify a sparse set of the most informative features given high-dimensional feature data. We fit a decision tree on the D-dimensional test set embeddings, $\phi(\mathbf{X})$, so that the decision tree can predict the observed test outcomes, $o(f(\mathbf{X}), \mathbf{y})$. We set a maximum depth of the decision tree to prevent the decision tree from overfitting. The decision tree recursively selects the dimension of the embedding feature that maximizes the information gain about the NN outcome. After the decision tree is fit, each node in the tree corresponds to a hyper-plane in the embedding space; see Figure 5.2 Test (2). Each leaf node in the tree corresponds to a contiguous region in embedding space identified using a sparse subset of the embedding dimensions that give the most information (in a greedy sense) about the NN outcome. We refer to the fitted decision tree as our embedding map, because it maps regions in the embedding space to observed NN outcomes.

5.1.3 Approximating Internal Test Set Manifold

The embedding map found in Section 5.1.2 contains L leaf nodes that define contiguous regions in the embedding space, where leaf l is identified using a sparse subset $\mathbf{d}^l << D$ of the embedding space dimensions. Note that the number of dimensions in $\mathbf{d}^l$ is less than or equal to the maximum depth of the decision tree. Using the embedding map, we can partition the internal test

samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ by the leaf to which each sample maps, i.e.,

$$\mathbf{X} = \cup_{l=1}^{L} \mathbf{X}^l, \qquad \mathbf{X}^i \cap \mathbf{X}^j = \emptyset \quad \forall i \neq j \tag{5.2}$$

where $\mathbf{X}^l$ is the set of N^l test samples that map to leaf l.

$$\mathbf{X}^l = \{\mathbf{x}_i^l\}_{i=1}^{N^l} \tag{5.3}$$

We want to link the embedding space to the outcomes observed in testing and identify the tested regions of the embedding space. Given the contiguous region in the embedding space defined by leaf l and the test samples $\mathbf{X}^l$ that map to leaf l, we can define the tested region in the embedding space as a convex hull, H^l, around $\phi(\mathbf{X}^l)$. Let $\phi(\mathbf{X})[d]$ indicate the d^{th} dimension of the embedding feature. Then H^l is given by:

$$H^l = [min(\phi(\mathbf{X}^l)[d]), max(\phi(\mathbf{X}^l)[d])] \quad \forall d \in \mathbf{d}^l \tag{5.4}$$

See Figure 5.2 Test (3) for an illustration of convex hulls around the test samples in the embedding space.

The internal test set $\mathbf{X}$ has associated labels $\mathbf{y}$. Let $\mathbf{y} = \cup_{l=1}^{L} \mathbf{y}^l$ be the test set labels partitioned by the leaf to which each sample maps. $\mathbf{y}^l = \{y_i^l\}_{i=1}^{N^l}$ are the labels for the test samples that map to leaf l. Let $\mathbb{I}(\mathbf{a}, \mathbf{b})$ be an indicator function that is equal to 1 if $\mathbf{a} = \mathbf{b}$ and 0 otherwise. Assuming each test sample is equally likely, the probability of outcome a in leaf l can be computed as:

$$p(a|l) = \frac{1}{N^l} \sum_{i=1}^{N^l} \mathbb{I}(o(f(\mathbf{x}_i^l), y_i^l), a) \tag{5.5}$$

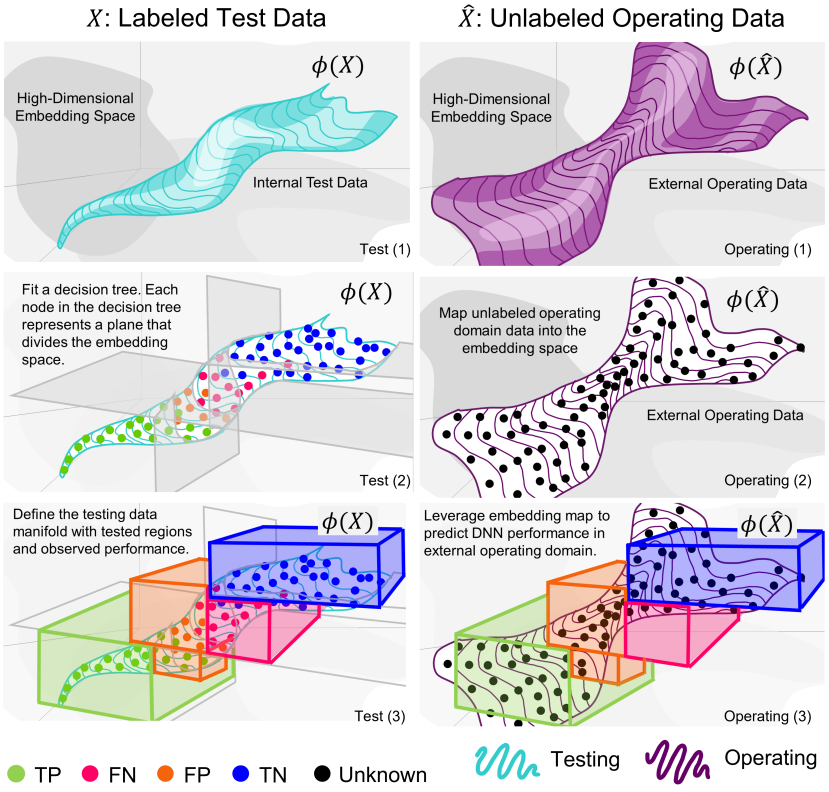

Figure 5.2: An illustration of the decision tree for mapping NN embeddings. Test data lie on a manifold in the embedding space. We identify structure in the embedding space as it relates to the NN outcome. For binary classification the possible outcomes are true positive (TP), false negative (FN), false positive (FP) and true negative (TN). The structure identified using labeled test data can be leveraged to predict the NN's performance on unlabeled operating data, where the outcome is unknown. Best viewed in color.

The boxes in Figure 5.2 Test (3) are colored to match the most likely test outcome in the leaf region to illustrate linking a region of embedding space to the NN outcomes observed in testing.

5.1.4 Inference on External Operating Data

We leverage the embedding map on unlabeled, external operating data; see Figure 5.2 Operating (1). The external operating samples can be mapped into the NN embedding space as $\phi(\hat{\mathbf{X}})$; see Figure 5.2 Operating (2). For the operating samples that map to leaf l, the sample is deemed "inside" the testing domain if it lies inside the convex hull of the test samples observed at that leaf node, H^l:

$$\phi(\hat{\mathbf{x}})[d] \in [min(\phi(\mathbf{X}^l)[d]), max(\phi(\mathbf{X}^l)[d])] \quad \forall d \in \mathbf{d}^l \tag{5.6}$$

The external operating samples that do not map inside the tested regions are assigned to leaf $L+1$ where the outcome is unknown. The external operating samples can then be partitioned to the $L+1$ leaf nodes:

$$\hat{\mathbf{X}} = \cup_{l=1}^{L+1} \hat{\mathbf{X}}^l, \qquad \hat{\mathbf{X}}^i \cap \hat{\mathbf{X}}^j = \emptyset \quad \forall i \neq j \tag{5.7}$$

where $\hat{\mathbf{X}}^l$ is the set of M^l external operating samples that map to leaf l.

$$\hat{\mathbf{X}}^l = \{\hat{\mathbf{x}}_i^l\}_{i=1}^{M^l} \tag{5.8}$$

5.1.5 Network Generalization Prediction

The goal of Network Generalization Prediction is to predict NN performance for an unlabeled, external operating set from which labeled test data are not

available. The probability that a sample in the external operating set maps to leaf l can be approximated by the fraction of the operating samples that map to leaf l:

$$p(l) = \frac{M^l}{M} \tag{5.9}$$

The probability of encountering outcome a in the operating domain is:

$$p(a) = \sum_{l \in L} p(a|l)p(l) \tag{5.10}$$

$p(a)$ can be computed for each outcome a observed in testing (assuming we have discrete outcomes and outcome possibilities). The probability of an unknown outcome can be computed as the probability that external operating samples map outside the tested regions, i.e., M^{L+1}/M.

5.2 Experiments

In this section we demonstrate that our embedding map can be used to accurately predict NN performance for unlabeled, external operating datasets for three binary image classification tasks: pedestrian classification, melanoma classification, and animal classification. Figure 5.3 shows examples of the images for each of these tasks.

5.2.1 Pedestrian Classification

One of the most safety-critical tasks for autonomous perception systems in self-driving vehicles is to detect and avoid pedestrians. We consider pedestrians from three driving perception datasets: Berkeley Deep Drive 100k (BDD) [25], Cityscapes [72], and Joint Attention in Autonomous Driving (JAAD)

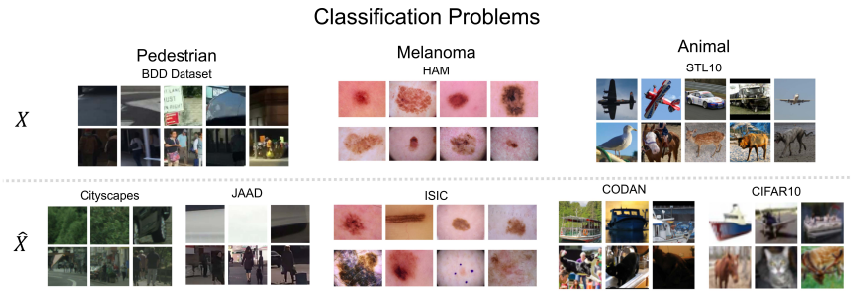

Figure 5.3: Classification tasks. X indicates the internal dataset that is used to train the NN classifier and fit the embedding decision tree. $\hat{X}$ indicates the unlabeled, external operating dataset. For each dataset, the top row shows a random sampling of negative examples, and the bottom row shows a random sampling of positive examples.

[71]. From the images in these datasets, for positive examples, we cropped square patches containing pedestrians, with area of greater than 300 pixels. For negative examples, we cropped random, square image patches of 100 $\times$ 100 pixels. The pedestrian image patches were resized to 100 $\times$ 100 pixels. We use BDD as the internal dataset versus Cityscapes and JAAD that are used as external datasets. The BDD dataset was recorded in different settings across the US in varying weather conditions, day and night times. Cityscapes was recorded in 50 cities in Germany during the day in fair weather conditions. JAAD was recorded in North America and Europe in mainly daytime settings with clear weather. Between the internal and external datasets, we expect changes in image statistics like brightness and saturation as well as some structural changes in the size and location of pedestrians in the image due to changes in how roads and sidewalks are laid out in different cities and countries.

5.2.2 Melanoma Classification

Skin cancer is one of the most common cancers, affecting one in five Americans throughout their lifetimes [73]. Melanoma is the deadliest type of skin cancer across all age groups, and early detection is crucial for effective treatment of this disease. Melanomic skin lesions demonstrate a wide variety in terms of disease expression such as lesion border, asymmetry, actual color, and diameter. Melanoma incidence varies with age and sex: melanoma is more common in women than men in patients younger than 50; however, in older patients melanoma is more common in men [74]. Images of skin lesions can exhibit additional variation that comes from the image acquisition and subsequent processing. For instance, lighting conditions, image capture modalities (dermascopes, phone cameras, handheld cameras, etc.), focus, motion blur, color correctness, and image compression can each affect the appearance and characteristics of an image taken from a lesion.

Due to its high prevalence, skin cancer detection is a popular application area for machine learning researchers. Similar to how dermatologists visually inspect the lesions on patients to make a diagnosis, pictures of skin lesions taken using cameras can be used to train NNs to classify benign versus malignant, i.e., melanomic, lesions. In the last five years, multiple methods in the literature have described NNs for classification of skin lesions and reported results that match or exceed those from dermatologists [73], [75]. Further, there are now some phone applications that claim to automatically classify skin lesions based on image(s) uploaded by the patient. Generally, NNs are black box systems that do not provide an understanding of why or how they

produce a particular prediction. At the same time, it is well known that NNs are prone to latching onto irrelevant attributes or characteristics of input data and moreover their performance can degrade due to minor differences in image characteristics, e.g., changes in image brightness, saturation, noise, etc., or due to differences in the distribution of sub-populations in the data. For instance, Hosseini et al. have shown that simple impulse noise can break Google's Cloud Vision Application Programming Interface [46]. However, recent studies [2-4] have demonstrated that a typical deep neural network that contains millions of free parameters can easily distinguish between images sourced from different origins, even after careful pre-processing pipelines [76]–[78]. However, deep neural network classifiers exhibit unexpected instability even for simple perturbations [45], [46], especially in domains that suffer from small sample size problem such as medical imaging.

We address the task of classifying an image of a skin lesion as melanoma (the positive class), or benign (the negative class). We use the Human Against Machine 10000 (HAM) dataset [79] for our internal dataset and the SIIM-ISIC Melanoma Classification (ISIC) dataset [80] for our external dataset. HAM images are 450×600 pixels. We resized ISIC images to be the same size, i.e., 450×600 pixels. For melanoma classification, the ISIC operating data has more variation in image appearance, e.g., image saturation and hue, than the internal dataset.

5.2.3 Animal Classification

General animal and object classification is a standard computer vision task that could have safety implications in the future, e.g., household autonomous robots will need to differentiate pets from inanimate objects in order to safely navigate around changing environments. We aim to classify an image as an animal (the positive class) or an object (the negative class), with STL10 [81] as the internal dataset. The STL10 images are 96×96 pixels and they include animals: birds, horses, deer, dogs, and cats, and objects: planes, cars, trucks, and boats. For external datasets we use the Common Objects Day and Night (CODaN) [82] and CIFAR-10 [83] datasets.

In our NGP experiments, we only include the external dataset images of classes seen during training. The CODaN animal classes considered are dogs and cats, and the CODaN object classes considered are cars and boats. The CODaN dataset was compiled from other existing datasets; we exclude images taken from ImageNet because the NN classifiers we fine-tuned were originally trained on ImageNet. The CODaN dataset includes night and day images that are 256×256 pixels; we resized the images to 96×96 pixels. The CIFAR-10 animal classes considered are birds, horses, dogs, cats, and deer. The CIFAR-10 object classes considered are boats, cars, and trucks. CIFAR-10 images are 32×32 pixels, which we resized to 96×96 pixels. Note that for animal classification, both external datasets present some significant change in image distribution: the CODaN dataset includes night images that were not present in the internal dataset and the CIFAR-10 dataset has images with $1/3$ the resolution of the internal dataset.

5.2.4 Experimental Setup

For each classification task, we fine-tune three classifiers with different NN architectures: VGG [84], AlexNet [85], and DenseNet [86]; the pre-trained models are available in the PyTorch library. Each round of training considers 100 batches of images with a batch size of 8, where the images are sampled with a uniform probability for each class. The VGG and AlexNet models are trained with 10 rounds of training, a learning rate of $1e-6$ and a weight decay of $1e-3$. The DenseNet models are trained with 4 rounds of training, a learning rate of $1e-4$, and a weight decay of $1e-3$. VGG and AlexNet have an embedding space of $4,096$ dimensions. DenseNet projects into an embedding space of $W \times H \times 1664$ where W and H depend on the initial image size. Like the full DenseNet architecture, we use a Global Average Pooling (GAP) layer to convert from the 3D embedding to a 1664 dimensional vector for each image. For each architecture in each task, we fit a decision tree with a maximum depth of 10 to distinguish four outcomes: TP, FP, FN, and TN. We refer to the fitted decision tree as our embedding map.

5.2.5 Network Generalization Prediction

We pass the external dataset through the NN to obtain the embeddings, $\phi(\hat{\mathbf{X}})$. We subsequently map the external embeddings to leaves in the embedding map, $\hat{\mathbf{X}} = \cup_{l=1}^{L+1} \hat{\mathbf{X}}^l$. For each outcome, TP, FN, FP, and TN, we compute the probability of observing the outcome in the external dataset according to equation 5.10. If the external dataset has images that map outside the tested regions, i.e., outside the convex hulls defined as H^l for each leaf, the expected

outcome is unknown. The true results are the classification results when the NN is evaluated using the external dataset labels. We present the NGP results in two ways: numerically using an F1 score and visually.

5.2.6 Numerical Network Generalization Prediction Results

To facilitate numerical comparison, we show NGP results with an F1 score [87] where we compare the predicted probability of a correct outcome and predicted probability of failure with the true probability of a correct outcome and the true probability of failure. For our NGP algorithm, it is ambiguous whether the unknown outcomes will be correct or a failure. To address this, we compute the F1 score in two ways: assuming the unknown outcomes are correct classifications, denoted “Ours" in Table 5.1, and assuming the unknown outcomes are failures, denoted “Ours+" in Table 5.1. The two assumptions reflect optimistic and conservative failure probability predictions. The F1 score shows whether the NGP algorithm accurately predicts overall success and failure of the NN. We compare against the NGP approach proposed in Chapter 4, denoted CS NGP. The CS NGP requires distributions of each class and the distribution of context features to make predictions. For pedestrian classification we use image brightness, scene type, weather, and time of day as available context features. For melanoma classification we use average image hue, saturation, value, patient age, sex, and lesion location as possible context features. Labeled metadata are not available for the animal classification task so we cannot include a comparison to CS NGP.

In Table 5.1 we show the numerical F1 NGP results. Our proposed NGP

Operating Domain	NGP	Architecture VGG	AlexNet	DenseNet
Cityscapes	Ch. 4	0.984	0.964	**0.956**
	Ours	**0.985**	**0.984**	0.947
	Ours+	0.972	0.954	0.945
JAAD	Ch. 4	0.958	0.927	0.954
	Ours	**0.988**	**0.986**	**0.987**
	Ours+	0.959	0.927	0.948
ISIC	Ch. 4	0.908	0.923	**0.971**
	Ours	**0.935**	**0.970**	0.800
	Ours+	**0.935**	0.940	0.929
CODaN	Ours	0.950	0.901	0.942
	Ours+	**0.993**	**0.984**	**0.951**
CIFAR-10	Ours	0.959	0.928	0.951
	Ours+	**0.982**	**1.000**	**0.960**

Table 5.1: NGP numerical F1 results for pedestrian, melanoma, and animal classification tasks with different architectures.

approach can robustly predict performance over different NN architectures, classification tasks, and different external dataset distributions. The proposed NGP algorithm consistently makes more accurate predictions than the CS NGP baseline in Chapter 4 and does not require knowledge about class or context distributions. Our approach is state of the art in 13 of the 15 examples while the CS NGP baseline is more accurate only when predicting performance for Cityscapes, DenseNet and ISIC, DenseNet.

5.2.7 Graphical Network Generalization Prediction Results

Second, we present the full granularity of the NGP results visually in Figure 5.4 and in Figure 5.5. The stacked bar graphs show the predicted probability of TP, TN, failure, and an unknown outcome. The goal is to have the predicted

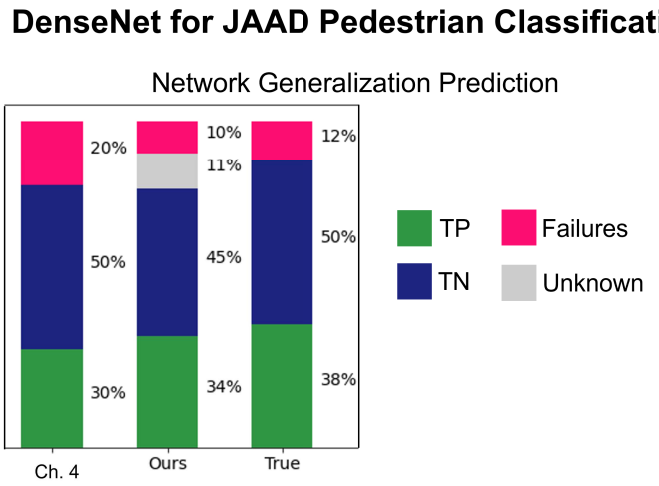

Figure 5.4: DenseNet JAAD visualized results for Network Generalization Prediction.

results match the 'True' results, i.e., the classification results when the NN is evaluated using the external dataset images and labels. It is common to use the internal test results to predict NN generalization, so we show the average test set results as a naive baseline for performance prediction, denoted Test Results in Figure 5.4 and Figure 5.5.

In Figure 5.4 we show a sample of the NGP visualized results for the JAAD external dataset with the DenseNet architecture. We show a subset of the NGP results in Figure 5.4 so that the results can be clearly explained. The totality of the results are shown in Figure 5.5. The NGP visualized results for all operating domains and NN architectures are shown in Figure 5.5. In Figure 5.4 the pink bars represent the probability of a failure, i.e., FN or FP. For our algorithm, the gray bar represents the probability of an unknown outcome. In Table 5.1 the predictions marked "Ours" correspond to predicting that the failures shown in pink are failures and the unknown outcomes are correct.

In Table 5.1 the predictions marked “Ours+" correspond to predicting that the failures shown in pink are failures and the unknown outcomes are also failures. Note that for JAAD, DenseNet the predicted probability of failure is 10%, which is close the true probability of failure of 12%, while the probability of failure or an unknown outcome is 21%, which over-predicts failure. Correspondingly “Ours" is the most accurate prediction for JAAD, DenseNet. Note that the true probability of failure lies between the predicted probability of failure and the predicted probability of failure plus the probability of an unknown outcome. In Figure 5.5 we show the NGP results for all tasks, all external operating domains, and all NN architectures. Note that our proposed NGP approach can robustly predict performance over different NN architectures, classification tasks, and different external dataset distributions. Our NN performance predictions are consistently more accurate than both baseline methods, CS NGP and the average test results, respectively. Looking at Figure 5.5, it can be seen that this is true for all tasks and all architectures except VGG for animal classification where we slightly under-predict failure.

The CS NGP method requires labeled distributions for the probability of the positive and negative classes where our method does not. Note that from the embedding space we are able to capture not only the probability of failure but also the overall class distribution for the external domain whether or not those distributions are the same. In the pedestrian and animal examples, the probability of positive and negative samples is roughly equivalent. On the other hand, in the melanoma example it is far more likely to have a benign image than an image of melanoma. In all tasks we capture accurate overall

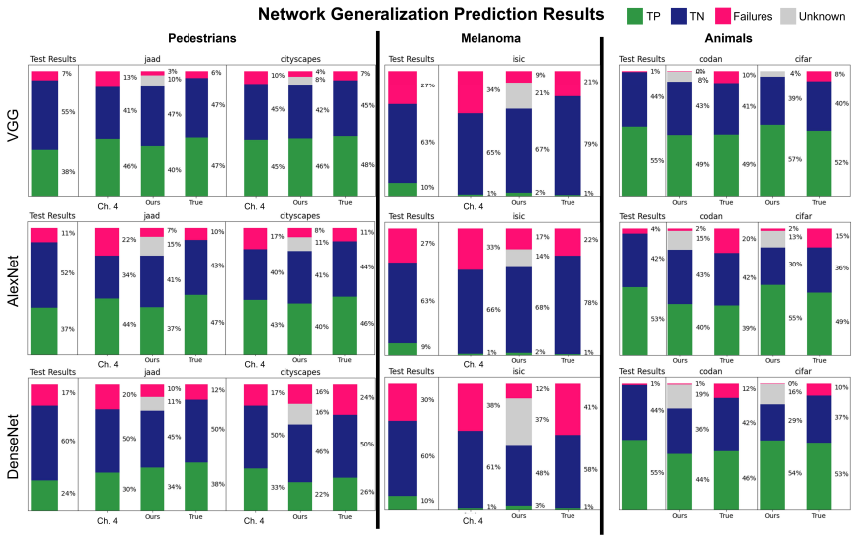

Figure 5.5: Network Generalization Prediction results for pedestrian classification, melanoma classification, and animal classification. We show results for three NN architectures: VGG, AlexNet, and DenseNet.

class distribution for the external operating domain.

5.3 Discussion

We demonstrate that mapping the structure in the NN embedding space can lead to powerful prediction of NN performance in external datasets across three high-complexity perception tasks. We consider pedestrian classification and melanoma classification, both of which are tasks where NNs must perform well in unconstrained environments and where commercial products are available at present. Techniques that can accurately predict NN performance in new operating domains without requiring labeled data, such as our proposed technique, are essential for both the safety and the fairness of NNs.

NN generalization depends on the internal training and testing data, the

NN, and the operating domain. Our proposed method does not require labeled operating data, can predict accurately how a NN will generalize, and can improve overall performance when the NN performance is poor. The percent of the operating data where the outcome is unknown can be used to determine whether it is appropriate to deploy the NN in the novel operating domain and answer questions like *has the NN been sufficiently tested?* and *have the right tests been performed?*

Our proposed approach is not restricted to binary classification problems, and is applicable for other feed-forward supervised learning problems, such as, multi-class classification and object detection. We leverage the structure in the NN embedding space for NGP, but this structure is likely useful for other tasks such as OOD rejection. Other directions for future work include further investigation on the decision tree structure. We began our experimentation with a tree depth of 10 arbitrarily and obtained exceedingly good results. The decision trees converged without finding the maximum number of leaves possible, so we did not perform extensive experiments with different tree depths. Future work could investigate different decision tree depths or random forests.

We identify the tested regions in the embedding space with convex hulls. It is not clear that a convex hull is always an appropriate choice. It may be useful to measure the distance from the test samples to determine if samples are inside the tested region instead of a binary decision based on whether the operating samples are inside or outside the convex hull. Additionally, the convex hull may not be able to capture all kinds of corruptions or shifts of

interest. For example, we consider NN classifiers that have been pre-trained with ImageNet; an adversarial sample that is a perturbed ImageNet image may still map within a convex hull if it maintains characteristics from the original dataset.

5.4 Conclusions

We propose a NGP method that can predict NN performance in a novel operating domain without requiring labeled data, context distributions, or class distributions. We demonstrate the robustness of the method over three classification tasks, three NN architectures, and five different realistic external datasets. More broadly, we believe that NGP is a task that warrants more attention to enable targeted performance prediction towards a specific operating domain. We believe this is a promising direction for further research and can be a step towards dependable and practical NNs for safety critical tasks in unconstrained environments.

Chapter 6

Extensions of NGP Subspaces

We have developed context spaces and embedding spaces to perform NGP, but these spaces are useful for other tasks. In this chapter, we leverage the context space (Chapter 3) and embedding subspace (Chapter 5) to improve the safety of the NN by designing safety functions; see Section 6.1. We also demonstrate that the context subspace (Chapter 4) can be used to predict whether the NN performance is robust without requiring information about the operating domain. This chapter contributes extensions of the previous technical chapters as follows:

1. Chapter 3: We design a safety function to reduce harmful failures in the simulated robot manipulation task under testing conditions. We reduce the harmful failures, in one example, by a factor of 700 while maintaining a high probability of success.

2. Chapter 5: We leverage the embedding subspace to identify regions where testing found a high probability of failure and design a safety function to reject unseen operating images that map to these failure

embedding regions. Our error rejection safety function outperforms a baseline when the NNs has an overall probability of failure over 10% in testing.

3. Chapter 4: We inspect the context subspace found for a melanoma classifier and determine that saturation, which is not a clinically relevant feature, impacts the NN performance. We predict that the NN will not be robust in novel operating domains; further testing confirms that the NN performance degrades significantly when used on skin lesion images from an unseen dataset.

In Section 6.1 we present background information about safety functions. In Section 6.2 and Section 6.3, we present safety functions for the work in Chapter 3 and Chapter 5, respectively. In Section 6.5 we present our work predicting robustness based on the context feature ranking algorithm from Chapter 4.

6.1 Safety Functions for Neural Networks

Safety functions are external functions used in functional safety when the system, e.g., a chemical plant, has an unacceptable level of risk inherent to the task [88]. When the system encounters a hazardous condition, e.g., high pressure, the safety function, e.g., a pressure relief valve, acts to bring the system to a safe state, e.g., the pressure relief valve allows some material to move to another chamber and the overall pressure returns to a safe level. The internal decision-making structure of a NN is obscured by thousands or millions of learned weights that cannot be inspected. It is highly likely that

NNs, when used in safety critical tasks, have an unacceptable inherent level of risk. For the first time, we design safety functions for NNs performing safety critical tasks. Using the context space, see Section 6.2, or the embedding subspace, see Section 6.3, we identify hazardous conditions for the NN and intervene to improve the safety of the NN.

6.2 Safety Function in the Context Space

In Chapter 3 we proposed a NGP algorithm for problems with a known, observed context space. In Section 3.2 we evaluated our NGP algorithm on a simulated robot control problem where a NN moved a robot arm linearly towards a goal location across the path of a moving obstacle. The context space of the problem was defined by the speed of the moving obstacle, v, the time the obstacle started moving, τ, and the robot goal location, z. See Section 3.2 for the full details of the robot control problem.

Testing revealed that harmful failures only occurred with robot goals, z, greater than or equal to 38.47 inches. We designed a safety function to reduce harmful failures by clipping the robot goal input to the network to be between $[0, 38.47 - \delta]$ inches. We chose $\delta = 0.5$ inches. The reader is reminded that the NN continues to move the robot after the goal position is reached, until the simulation ends at 100 seconds. Clipping the robot goal *input* to the NN was intended to make the NN behave more conservatively[1]; it was still possible for the robot to exceed the clipped goal and reach the original goal position. The safety function did not change the conditions for success: for a simulation

[1]This is a similar idea to Control Governors [89].

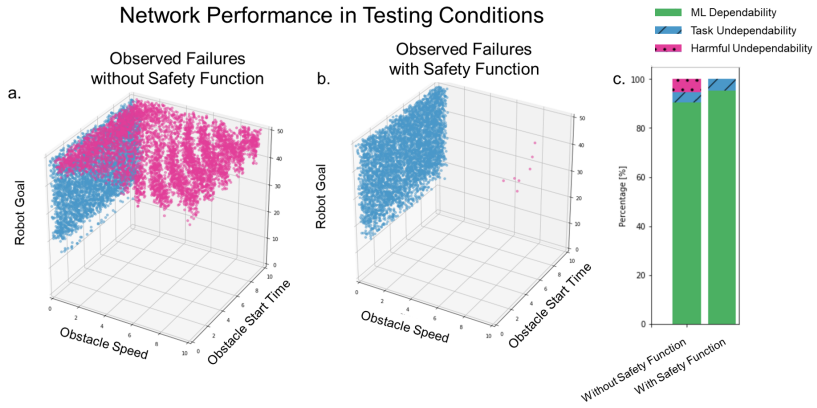

Figure 6.1: A comparison of the NN performance without the safety function and with the safety function. Task failures are indicated in blue. Harmful failures are indicated in pink. (a) a reprint of Figure 3.2 to facilitate comparison. (b) the observed failures in Testing Conditions with the safety function. (c) a comparison of the NN ML Dependability, Task Undependability, and Harmful Undependability with and without the safety function. Note, the Harmful Undependability is reduced from 5.47% to 0.007% with the safety function.

to be successful the robot had to reach the original goal position. 100,000 new test scenarios were sampled from the Testing Conditions and run with the safety function. With the safety function, the NN had a ML Dependability of **95.19%**, a Task Undependability of **4.81%**, and a Harmful Undependability of **0.007%**. Figure 6.1, above, offers a side-by-side comparison of observed failures and NN performance with and without the safety function.

6.2.1 Safety Function in the Context Space Results

The safety function results, see Figure 6.1, reveal that in most of the examples that were harmful failures in testing, the robot had enough time to avoid collision and reach the goal before the end of the simulation. But the strategy learned by the NN did not time the robot's approach correctly. Interestingly,

the reward function was specifically designed to weight safety over task completion: a collision resulted in a penalty of -50 points whereas reaching the goal resulted in a reward of 30 points. The NN learned an incorrect trade-off between moving towards the goal and avoiding the obstacle. While we do not claim that it would be impossible to craft a reward function to perfectly complete this task without harm, this example illustrates that designing a reward function that appropriately weights task requirements and safety constraints is not trivial. Safety functions are an explainable alternative to hand crafting reward functions and guarantee a degree of safety for a NN.

The safety function reduced the number of harmful failures by a factor of 700. With our safety function, the only harmful failures that remain occur when the robot goal and the obstacle speed reach the maximum limit seen in the simulation, i.e., the harmful failures occur at the outer boundary of the scenarios seen in training. Surprisingly, even though our safety function clipped the input robot goal, it converted many harmful failures into successes. Clipping the robot goal made the NN behave more conservatively, i.e. the NN waited for the obstacle to pass before moving as far forward as it could. In general, we expect safety functions to reduce the probability of harmful failures, but we do not expect them to increase the probability of success.

6.3 Safety Function in the Embedding Space

In Chapter 5 we map the NN embedding space based on the test outcomes, e.g., false positives, true negatives, etc. We define regions in the embedding space associated with the different outcomes and perform NGP based on

how unlabeled operating images map into the embedding regions. When we perform NGP in Chapter 5, we predict the probability of failure in the novel operating domain, i.e., the fraction of operating domain images that will have incorrect predictions. We can also leverage the embedding map to predict whether individual images have been misclassified; this task is called error prediction in the literature. For each leaf node in the embedding map, there are N^l test samples that map to the leaf. We identify leaf nodes where more than 50% of their associated test samples were failures as failure leaves. In the error prediction task, we have M unlabeled operating domain samples. When an operating sample maps to a failure leaf, we predict the operating sample will be misclassified, resulting in m rejected samples from the operating domain. We define the safety function so that the NN does not make a prediction for the m rejected samples but the NN makes a prediction for the $M - m$ remaining samples. As it is ambiguous whether the unknown outcomes will be correct or a failure, we assume that images that map to an unknown outcome will be correctly classified. To compare against a baseline, we use the ranking in [34] and predict that the m lowest scoring samples will be misclassified. We report error prediction results using the F1 score, where we denote correct predictions as the positive class and incorrect predictions as the negative class.

6.3.1 Safety Function in the Embedding Subspace Results

We present the error prediction results in Table 6.1 for the pedestrian classification, melanoma classification, and animal classification tasks presented

Operating Domain	Error Pred.	Architecture VGG	AlexNet	DenseNet
Cityscapes	NN	**0.962**	**0.944**	0.863
	[34]	0.954	0.932	0.809
	Ours	0.949	0.920	**0.902**
JAAD	NN	**0.971**	**0.949**	0.935
	[34]	0.961	0.932	0.901
	Ours	0.952	0.915	**0.939**
ISIC	NN	0.885	0.879	0.741
	[34]	0.845	0.834	0.646
	Ours	**0.905**	**0.928**	**0.822**
CODaN	NN	**0.949**	**0.891**	**0.938**
	[34]	0.946	0.880	0.936
	Ours	0.945	0.867	0.923
CIFAR-10	NN	**0.958**	**0.918**	**0.949**
	[34]	0.957	0.910	0.943
	Ours	0.954	0.899	0.933

Table 6.1: Error prediction F1 scores for pedestrian, melanoma, and animal classification tasks.

in Section 5.2. Note, neither our proposed error prediction approach nor the baseline outperform keeping all the NN predictions in many experiments. However, our proposed error prediction approach outperforms both the NN and the baseline specifically when the overall NN performance is poor.

In Table 6.2 we reprint the results from Table 6.1 for the operating domains and the NN architectures where the observed probability of failure throughout the operating domain is greater than or equal to 15%. When the NN performance is poor, our error prediction approach outperforms the NN and the baseline in three of five experiments. However, for the animal classification task, there are two examples where the NN performance is poor and our NGP algorithm does not predict a high probability of failure: CODaN, AlexNet

OD	Cityscapes	ISIC			CODaN	CIFAR-10
Arch.	DenseNet	VGG	AlexNet	DenseNet	AlexNet	AlexNet
NN	0.863	0.885	0.879	0.741	**0.891**	**0.918**
[34]	0.809	0.845	0.834	0.646	0.880	0.910
Ours	**0.902**	**0.905**	**0.928**	**0.822**	0.867	0.899

Table 6.2: Error prediction F1 scores for classification tasks where the probability of failure in the operating domain (OD) and architecture (Arch.) is greater than or equal to 15%.

(20% probability of failure) and CIFAR-10, AlexNet (15% probability of failure). In these examples we predict an unknown outcome with a probability of 15%, and 13%, respectively (see Figure 5.5 for the NGP predictions for all tasks and architectures). The external datasets for animal classification represent two of the largest distribution shifts we encounter: CODaN includes day and night images where the internal dataset only includes daytime images. CIFAR-10 images are originally 32×32 pixels but we resize them to be 96×96 to match the size of the internal dataset images; this is a dramatic reduction in image resolution for the NN. It is not surprising that most of the failures stem from images that map outside the tested region, but this means that our proposed error prediction method is not as effective for very large domain shifts.

Interestingly, for operating domains and NNs with a probability of failure below 15%, neither the baseline nor our proposed method improves on the NN performance when all predictions are kept; see Table 6.1. This makes sense, because if the NN is performing well, we want to take advantage of all of its predictions.

6.4 Safety Functions Discussion

We present two examples of safety functions for NNs: one based on hazards identified in the context space, Section 6.2, and one based on hazards identified in the NN embedding space, Section 6.3. In the safety function designed for the robot control context space, our safety function was hand-crafted. Targeted safety functions could prove a scalable approach for ensuring safety in dynamic environments, and may be more feasible than retraining the NN for different operating conditions. The safety function for error prediction in image classification was not learned, but it took advantage of the feature projection learned by the NN. We compare against the baseline in [34], which performs error projection based on the softmax scores from the final NN layer. Our embedding map leverages information about the structure of the embedding space. This is fundamentally different than leveraging information from the softmax scores and can be a complementary source of information. In the future, we expect safety functions for NNs to become a key tool to render NNs practical for safety critical tasks.

6.5 Predicting Robustness from the Context Subspace

In Chapter 4 we proposed a context feature ranking algorithm, Algorithm 1, to rank the available context features by how much information they provide about the NN performance. We propose that this ranking algorithm can be useful to inspect what impacts NN performance. It is well known that NNs

can fail to generalize if during training the NN learns to extract features that are not discriminative for input data with a different distribution. For instance, a NN may learn features that are based on spurious correlations within the training data if the samples used in training exhibit a bias in terms of one or more characteristics. NN robustness is particularly important in medical applications. For Melanoma classification, lesion rotation in the image and image hue have been identified as factors that can impact classification performance [90]. NNs detecting MRI scan location even after careful post-processing [78], [77]. It has been shown that NNs can classify skin lesions based solely on the surrounding skin *without* information about the skin lesion [91]. Gentian violet skin markings have been shown to increase the false positives for melanoma classification with NNs [92]. This is an problem that may not be easily caught since the test set is often a subset of the same overarching dataset from which the training data was derived and therefore has the same distribution, including the spurious correlations. In this section, we leverage Algorithm 1 to inspect which context features impact the performance of a melanoma classifier for the purpose of evaluating if the NN is relying on clinically relevant or spurious features. See Section 5.2.2 for an overview of melanoma classification. We utilize a combination of context features that are either part of the dataset (e.g. patient age) or can be calculated directly based on images (e.g. image brightness or hue) for this purpose.

6.5.1 Predicting Robustness Results

The HAM10000 (HAM) dataset [79] contains 7,470 unique lesions over 10,015 images of size 450 × 600 pixels, including 1,113 images of melanoma from 614 unique lesions. We split the HAM dataset into three folds: 3,137 images for training, 357 images for validation, and 4,544 images for testing. The images are randomly assigned to folds, and all images belonging to the same lesion are assigned to the same fold to avoid data leakage. We fine-tune a pretrained VGG [93] network for melanoma classification with the training fold of the HAM dataset; training images are sampled with replacement so that 50% of training images are benign and 50% are malignant. (Pre)cancerous, non-melanoma samples were specifically excluded from training and testing in the HAM dataset[2]. The melanoma classifier achieves an AUC of 0.852 on the HAM test images[3].

The SIIM-ISIC Melanoma Classification (SIIM-ISIC) Dataset includes images captured over 22 years from six different centers taken with or without polarized light using a contact or noncontact dermascope [80]. The SIIM-ISIC dataset contains 33,126 lesion images from 2,056 patients, including 584 images of melanoma. Images are labeled as benign or melanoma. The images vary in size from 480 × 640 pixels to 4000 × 6000 pixels. We resize each image in the SIIM-ISIC dataset to 450 × 600 pixels to conform to the input image size from the HAM dataset. See Figure 6.2 for a visualization of images sampled

[2]The dataset labels 'vasc', 'bkl', 'df', 'nv' were assigned benign label. 'mel' was assigned malignant. (Pre)cancerous, non-melanoma samples 'bcc' and 'akiec' were excluded.

[3]We consider this performance adequate for the robustness analysis presented here as it is reasonably close to the AUC of the winning team in the 2018 MICCAI HAM Disease Classification Challenge who achieved an AUC of 0.885 on the more challenging 7-disease classification task [94].

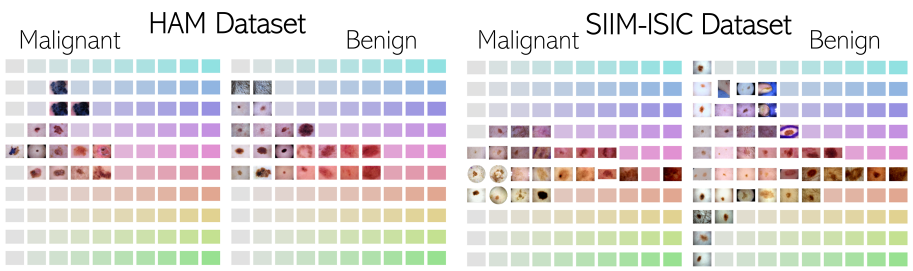

Figure 6.2: Sampling of images from the HAM and SIIM-ISIC Skin Lesion datasets separated by malignant/benign labels. For each category the images are shown according to their hue and saturation with consistent ranges across datasets. Hue/saturation ranges without available images are shown as solid colors. Note that the malignant images in the HAM dataset exhibit a smaller spread over saturation compared to the HAM benign images or the SIIM-ISIC images.

from both datasets. Note that the malignant images in the HAM dataset exhibit a smaller spread over hue and saturation compared to the HAM benign images or the SIIM-ISIC images. The classifier trained on the HAM dataset achieves only an AUC of 0.599 on the SIIM-ISIC dataset. This suggests that the classifier has latched onto some spurious correlation in the HAM dataset.

We aim to identify which context features impact the NN performance. In the HAM dataset, the metadata available are the patient age, the patient sex, and the lesion location. We are also interested in context features that describe the image properties. The pixels in digital images are described using a color space; Red-Green-Blue (RBG) is the most common color space, but the Hue-Saturation-Value (HSV) color space is easier to interpret because it considers hue (color), saturation, and value (brightness) independently. Therefore, we include the average image hue, saturation, and value as context features. We consider patient age, patient sex, lesion location, image hue, image saturation, and image value as the context features with Algorithm 1.

We rank the context features according to Algorithm 1. The resulting ranking of the context features is as follows: 1. image saturation, 2. patient sex, 3. patient age, 4. lesion location, 5. image brightness, 6. image hue. As mentioned in Section 5.2.2, the incidence rate of melanoma is related to patients' sex and age. However, image saturation is not a clinically relevant feature. We investigate the number of benign and malignant training images indexed by image saturation and the classifier sensitivity and specificity indexed by image saturation; see Figure 6.3. Note that all the melanomic training images are with lower saturation, $0\% - 38\%$, while the benign training images have saturation from $0\% - 70\%$. In the lower saturation region, the classifier has a high sensitivity and low specificity. From the HAM dataset it is not clear what the sensitivity is for images with saturation $> 40\%$. This indicates that the classifier has not learned to distinguish melanomic and benign lesions in a meaningful way. We examine the classifier's performance on the SIIM-ISIC dataset indexed by saturation and find that the classifier has higher sensitivity and lower specificity in low saturation images, but has significantly worse specificity on the SIIM-ISIC dataset than the HAM dataset, see Figure 6.3.

We define sub-populations according to the three context features that are most indicative of the NN performance: image saturation, patient sex, and patient age. Most of the HAM images have a saturation between 0% and 50%, so we define low saturation as $< 25\%$ and high saturation $\geq 25\%$. The incidence of melanoma for men and women changes around 50 years, so we define younger as < 50 years and older as ≥ 50 years. Based on the breakdown of performance shown in Figure 6.4 for the HAM dataset, we can

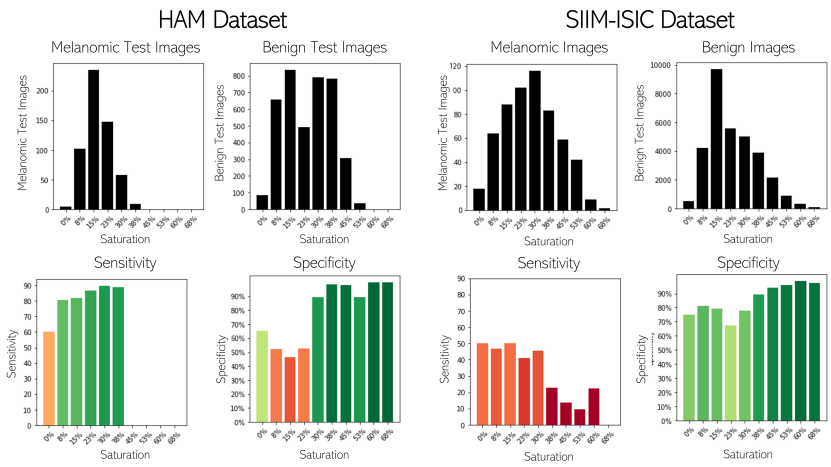

Figure 6.3: Left Top: HAM Dataset test image count indexed by image saturation, note the histogram vertical axes are not at the same scale. Left Bottom: classifier Sensitivity and Specificity indexed by image saturation. Right Top: SIIM-ISIC Dataset test image count indexed by image saturation, note the histogram vertical axes are not at the same scale. Right Bottom: classifier Sensitivity and Specificity indexed by image saturation. Sensitivity and Specificity bars are colored green for high performance and red for poor performance.

observe that the network has non-uniform performance across the different sub-populations. In particular, as previously shown in Figure 6.3 most of the melanoma images fall into the low-saturation bin, and we see that the classifier AUC is ≤ 0.74 on the low-saturation images compared to ≥ 0.93 across the high-saturation images.

6.5.2 Predicting Robustness Discussion

In this section we leverage Algorithm 1 in a novel way which allows us to inspect whether a trained NN is utilizing clinically relevant features. This in turn informs us about whether the classifier is expected to be robust when deployed on real world data with a different distribution. We demonstrated

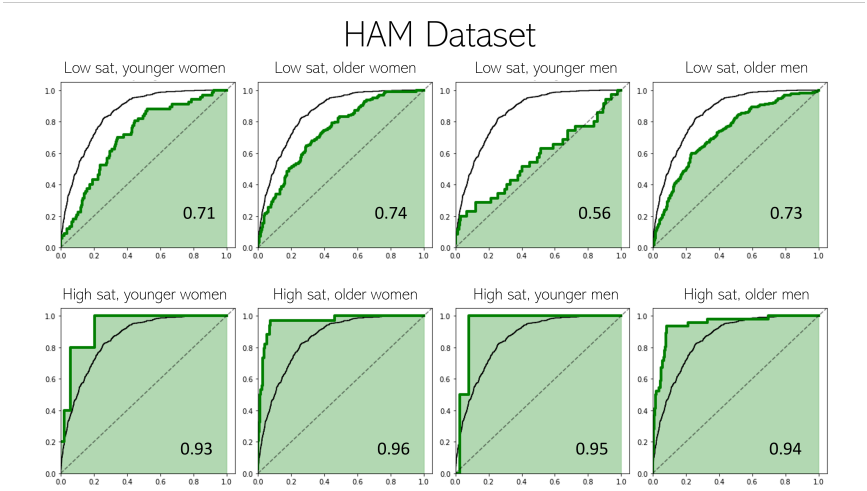

Figure 6.4: Classifier AUC on sub-populations of the HAM test images. For each plot, the True Positive Rate is shown on the y-axis and the False Positive Rate is shown on the x-axis. The overall AUC curve is shown in black in each plot, the sub-population AUC curve is shown in green. We define low saturation as $< 25\%$ and high saturation $\geq 25\%$. We define younger as < 50 years and older as ≥ 50 years.

that our melanoma classifier relied chiefly on image saturation, instead of clinically relevant features, thereby precluding the classifier from generalizing in a robust or fair way. We have proposed a method to reveal what context features the NN may be using when producing a classification. This information can help better evaluate the robustness and fairness of a trained network, and identify the root-cause for potential degradation of performance across different datasets or subpopulations of data. Our method can be used to identify gaps in the training data or problems with the NN that should be addressed prior to deployment.

6.6 Conclusions

We have shown that the context spaces (Chapter 3 and Chapter 4) and the embedding space (Chapter 5) can be used to improve the NN safety and to predict the NN robustness. The focus of this dissertation is NGP, but we believe the methods we developed for NGP have broader applications. We outline ideas for future work and our conclusions in Chapter 7.

Chapter 7

Conclusions

When NNs perform safety critical tasks in unconstrained environments, predicting how the NNs will generalize is essential for safety. In Chapter 3, we developed an algorithm for Network Generalization Prediction (NGP) from a finite test set to predict NN performance in novel operating domains. Understanding which factors impact NN performance in high-complexity tasks is challenging, so we proposed the context subspace (Chapter 4) and the embedding subspace (Chapter 5) to map NN performance based on interpretable context features or the DNN embedding, respectively. Both the context subspace and the embedding subspace can be used to accurately predict the NN performance in novel operating domains.

The context subspace relates NN performance to interpretable context features and renders NGP tractable for complex problems when it is unknown which factors impact NN performance. In Section 6.5 we demonstrated that the context subspace also has extensions beyond NGP; specifically, we show that the most informative context features for medical image analysis can be inspected to see if they are clinically relevant and thus predict whether the

NN is expected to be robust in novel operating domains. The embedding subspace relates regions of the NN embedding to different outcomes, e.g., false positive, true negative, etc. Finding the embedding subspace does not require the labels and distributions, but it does require unlabeled operating data. The embedding subspace automatically identifies which novel operating samples are unlike those seen before and which operating samples will likely result in incorrect predictions.

For the first time, we propose safety functions for NN, i.e., external functions that can mitigate risk inherent in the NN. In Section 6.2 and Section 6.3 we demonstrate that safety functions for NNs can improve safety in a robot control problem and image classification problems, respectively. As NNs become more integrated in cyber-physical systems that operate in unconstrained environments, e.g., self-driving vehicles, medical image analysis, robot control, etc., techniques like NGP and safety functions will become increasingly important for safety. We believe the work in this thesis can help address a pressing need for techniques to systematically evaluate NNs in high dimensional problems and provide actionable guidance on why a NN is failing, when it is safe to deploy, and how to improve NN performance.

This thesis opens up many directions for future research. Uncertainty estimation is an area of active research in ML. We have developed methods for deterministic NGP, but in the future NGP with uncertainty estimation would have many applications in safety critical tasks. We believe there are also other uses for context and embedding subspaces. The context subspace provides actionable information that can be used to target training or testing

data collection for contexts with poor performance.

NNs are increasingly performing safety critical tasks in unconstrained environments. The work in this thesis provides a strong technical foundation for NNs applied to safety critical tasks and for ML research to better understand and predict NN performance.

Bibliography

[1] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *arXiv preprint arXiv:1910.07738*, 2019.

[2] B. Wilson, J. Hoffman, and J. Morgenstern, "Predictive inequity in object detection," *arXiv preprint arXiv:1902.11097*, 2019.

[3] D. Wen, S. M. Khan, A. J. Xu, H. Ibrahim, L. Smith, J. Caballero, L. Zepeda, C. de Blas Perez, A. K. Denniston, X. Liu, *et al.*, "Characteristics of publicly available skin cancer image datasets: A systematic review," *The Lancet Digital Health*, 2021.

[4] D. Teney, E. Abbasnejad, and A. v. d. Hengel, "Unshuffling data for improved generalization," *arXiv preprint arXiv:2002.11894*, 2020.

[5] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[6] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *arXiv preprint arXiv:1911.08731*, 2019.

[7] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, S. Beery, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," *arXiv preprint arXiv:2012.07421*, 2020.

[8] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 624–639.

[10] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5400–5409.

[11] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[12] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*, Springer, 2016, pp. 443–450.

[13] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[14] V. Vapnik and V. Vapnik, "Statistical learning theory wiley," *New York*, vol. 1, no. 624, p. 2, 1998.

[15] B. Kim, H. Kim, K. Kim, S. Kim, and J. Kim, "Learning not to learn: Training deep neural networks with biased data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9012–9020.

[16] S. A. Taghanaki, M. Havaei, A. Lamb, A. Sanghi, A. Danielyan, and T. Custis, "Jigsaw-vae: Towards balancing features in variational autoencoders," *arXiv preprint arXiv:2005.05496*, 2020.

[17] L. Oakden-Rayner, J. Dunnmon, G. Carneiro, and C. Re, "Hidden stratification causes clinically meaningful failures in machine learning for medical imaging," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, ser. CHIL '20, Toronto, Ontario, Canada: Association for Computing Machinery, 2020, pp. 151–159, ISBN: 9781450370462. DOI: 10.1145/3368555.3384468. [Online]. Available: https://doi.org/10.1145/3368555.3384468.

[18] N. Sohoni, J. Dunnmon, G. Angus, A. Gu, and C. Ré, "No subclass left behind: Fine-grained robustness in coarse-grained classification problems," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 19 339–19 352.

[19] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, *et al.*, "Underspecification presents challenges for credibility in modern machine learning," *arXiv preprint arXiv:2011.03395*, 2020.

[20] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," in *Advances in Neural Information Processing Systems*, 2016, pp. 4312–4320.

[21] Y. Zhang, D. Balkcom, and H. Li, "Towards physically safe reinforcement learning under supervision," *arXiv preprint arXiv:1901.06576*, 2019.

[22] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[23] K. Goel, A. Gu, Y. Li, and C. Ré, "Model patching: Closing the subgroup performance gap with data augmentation," *arXiv preprint arXiv:2008.06775*, 2020.

[24] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang, "Adversarial domain adaptation with domain mixup," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6502–6509.

[25] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[26] A. RoyChowdhury, P. Chakrabarty, A. Singh, S. Jin, H. Jiang, L. Cao, and E. Learned-Miller, "Automatic adaptation of object detectors to new domains using self-training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 780–790.

[27] Z. Liu, Z. Miao, X. Pan, X. Zhan, S. X. Yu, D. Lin, and B. Gong, "Compound domain adaptation in an open world," *arXiv preprint arXiv:1909.03403*, 2019.

[28] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 11, pp. 3365–3385, 2019.

[29] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[30] R. Gong, W. Li, Y. Chen, and L. V. Gool, "Dlow: Domain flow for adaptation and generalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2477–2486.

[31] S. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," *arXiv preprint arXiv:1810.03237*, 2018.

[32] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 13, 2019.

[33] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[34] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *CoRR*, vol. abs/1610.02136, 2016. arXiv: 1610.02136. [Online]. Available: http://arxiv.org/abs/1610.02136.

[35] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," English (US), 2018.

[36] S. Mohseni, M. Pitale, J. Yadawa, and Z. Wang, "Self-supervised learning for generalizable out-of-distribution detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5216–5223, 2020. DOI: 10.1609/aaai.v34i04.5966. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5966.

[37] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 21 464–21 475.

[38] V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, M. Chiang, and P. Mittal, "Analyzing the robustness of open-world machine learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, ser. AISec'19, London, United Kingdom: Association for Computing Machinery, 2019, pp. 105–116, ISBN: 9781450368339. DOI:

10.1145/3338501.3357372. [Online]. Available: https://doi.org/10.1145/3338501.3357372.

[39] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Advances in neural information processing systems*, vol. 31, 2018.

[40] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[41] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[42] Z. Lin, S. D. Roy, and Y. Li, "Mood: Multi-level out-of-distribution detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 313–15 323.

[43] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[44] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[45] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[46] H. Hosseini, B. Xiao, and R. Poovendran, "Google's cloud vision api is not robust to noise," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, 2017, pp. 101–105.

[47] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[48] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1802–1811.

[49] A. K. Shekar, L. Gou, L. Ren, and A. Wendt, "Label-free robustness estimation of object detection cnns for autonomous driving applications," *International Journal of Computer Vision*, pp. 1–17,

[50] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[51] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li, "Adversarial objects against lidar-based autonomous driving systems," *arXiv preprint arXiv:1907.05418*, 2019.

[52] N. Drenkow, N. Sani, I. Shpitser, and M. Unberath, "Robustness in deep learning for computer vision: Mind the gap?" *arXiv preprint arXiv:2112.00639*, 2021.

[53] A. Subbaswamy, R. Adams, and S. Saria, "Evaluating model robustness to dataset shift," *arXiv preprint arXiv:2010.15100*, 2020.

[54] S. Cygert and A. Czyżewski, "Evaluating calibration and robustness of pedestrian detectors," in *International Conference on Multimedia Communications, Services and Security*, Springer, 2020, pp. 98–111.

[55] K. Yang, K. Qinami, L. Fei-Fei, J. Deng, and O. Russakovsky, "Towards fairer datasets: Filtering and balancing the distribution of the people subtree in the imagenet hierarchy," ser. FAT* '20, Barcelona, Spain: Association for Computing Machinery, 2020, ISBN: 9781450369367. DOI: 10.1145/3351095.3375709. [Online]. Available: https://doi.org/10.1145/3351095.3375709.

[56] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, "Fairness without demographics in repeated loss minimization," in *International Conference on Machine Learning*, PMLR, 2018, pp. 1929–1938.

[57] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *arXiv preprint arXiv:2004.14545*, 2020.

[58] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "Gan inversion: A survey," *arXiv preprint arXiv:2101.05278*, 2021.

[59] T. Ponn, T. Kröger, and F. Diermeyer, "Identification and explanation of challenging conditions for camera-based object detection of automated vehicles," *Sensors (Basel, Switzerland)*, vol. 20, no. 13, 2020.

[60] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[61] A. Paszke, *Reinforcement learning (dqn) tutorial, https://pytorch.org/tutorials /intermediate/reinforcement_q _learning.html*, 2019.

[62] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[63] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066 138, 2004.

[64] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, no. 1, pp. 175–186, 2014.

[65] R. Khanna, E. Elenberg, A. Dimakis, S. Negahban, and J. Ghosh, "Scalable greedy feature selection via weak submodularity," in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1560–1568.

[66] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides, "A greedy feature selection algorithm for big data of high dimensionality," *Machine learning*, vol. 108, no. 2, pp. 149–202, 2019.

[67] J. Jiao, Y. Zhu, H. Ye, H. Huang, P. Yun, L. Jiang, L. Wang, and M. Liu, "Greedy-based feature selection for efficient lidar slam," *arXiv preprint arXiv:2103.13090*, 2021.

[68] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[69] G. Braun, S. Pokutta, and Y. Xie, "Info-greedy sequential adaptive compressed sensing," *IEEE Journal of selected topics in signal processing*, vol. 9, no. 4, pp. 601–611, 2015.

[70] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[71] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 206–213.

[72] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[73] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[74] F. Morgese, C. Sampaolesi, M. Torniai, A. Conti, N. Ranallo, A. Giacchetti, S. Serresi, A. Onofri, M. Burattini, G. Ricotti, *et al.*, "Gender differences and outcomes in melanoma patients," *Oncology and therapy*, vol. 8, no. 1, pp. 103–114, 2020.

[75] L. R. Soenksen, T. Kassis, S. T. Conover, B. Marti-Fuster, J. S. Birkenfeld, J. Tucker-Schwartz, A. Naseem, R. R. Stavert, C. C. Kim, M. M. Senna, *et al.*, "Using deep learning for dermatologist-level detection of suspicious pigmented skin lesions from wide-field images," *Science Translational Medicine*, vol. 13, no. 581, 2021.

[76] C. Wachinger, B. G. Becker, A. Rieckmann, and S. Pölsterl, "Quantifying confounding bias in neuroimaging datasets with causal inference," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2019, pp. 484–492.

[77] E. Ferrari, P. Bosco, G. Spera, M. E. Fantacci, and A. Retico, "Common pitfalls in machine learning applications to multi-center data: Tests on the abide i and abide ii collections," in *Joint Annual Meeting ISMRM-ESMRMB*, 2018.

[78] B. Glocker, R. Robinson, D. C. Castro, Q. Dou, and E. Konukoglu, "Machine learning with multi-site imaging data: An empirical study on the impact of scanner effects," *arXiv preprint arXiv:1910.04597*, 2019.

[79] P. Tschandl, C. Rosendahl, and H. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific data*, vol. 5, no. 1, pp. 1–9, 2018.

[80] V. Rotemberg, N. Kurtansky, B. Betz-Stablein, L. Caffery, E. Chousakos, N. Codella, M. Combalia, S. Dusza, P. Guitera, D. Gutman, *et al.*, "A patient-centric dataset of images and metadata for identifying melanomas using clinical context," *Scientific data*, vol. 8, no. 1, pp. 1–8, 2021.

[81] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[82] A. Lengyel, S. Garg, M. Milford, and J. C. van Gemert, "Zero-shot domain adaptation with a physics prior," 2021. arXiv: 2108.05137 [cs.CV].

[83] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[84] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[85] A. Krizhevsky, *One weird trick for parallelizing convolutional neural networks*, 2014. arXiv: 1404.5997 [cs.NE].

[86] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, 2018. arXiv: 1608.06993 [cs.CV].

[87] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, "Thresholding classifiers to maximize f1 score," *ArXiv*, pp. 1402–1892, 2014.

[88] M. Medoff and R. Faller, *Function Safety And IEC 61508 Development Process*. Sellersville, PA: exida, 2014.

[89] E. Garone, S. Di Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306–328, 2017.

[90] X. Du-Harpur, C. Arthurs, C. Ganier, R. Woolf, Z. Laftah, M. Lakhan, A. Salam, B. Wan, F. M. Watt, N. M. Luscombe, *et al.*, "Clinically relevant vulnerabilities of deep machine learning systems for skin cancer diagnosis," *The Journal of investigative dermatology*, vol. 141, no. 4, p. 916, 2021.

[91] A. Bissoto, M. Fornaciali, E. Valle, and S. Avila, "(de) constructing bias on skin lesion datasets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[92] J. K. Winkler, C. Fink, F. Toberer, A. Enk, T. Deinlein, R. Hofmann-Wellenhof, L. Thomas, A. Lallas, A. Blum, W. Stolz, *et al.*, "Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition," *JAMA dermatology*, vol. 155, no. 10, pp. 1135–1141, 2019.

[93] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[94] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, *et al.*, "Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)," *arXiv preprint arXiv:1902.03368*, 2019.