Explaining and Interactively Debugging Deep Models

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
Genehmigte Dissertation von Xiaoting Shao aus China
Tag der Einreichung: 27. Mai 2022, Tag der Prüfung: 20.Juli 2022

1. Gutachten: Prof. Dr. Kristian Kersting
2. Gutachten: Prof. Dr. Stefano Teso
Darmstadt, Technische Universität Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Artificial Intelligence and Machine Learning Lab

Explaining and Interactively Debugging Deep Models

Accepted doctoral thesis by Xiaoting Shao

Date of submission: 27. Mai 2022
Date of thesis defense: 20.Juli 2022

Darmstadt, Technische Universität Darmstadt

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-218689
URL: http://tuprints.ulb.tu-darmstadt.de/21868
Jahr der Veröffentlichung auf TUprints: 2022

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 27. Mai 2022

Xiaoting Shao

author

Acknowledgement

There are many people that I would like to express my deep appreciation to. The first one I want to thank is my supervisor Prof. Dr. Kristian Kersting. Without him, I would not have made it this far and this work would not have been here. During these years of research, he has provided me with a lot of support and guidance, which helped me to develop as a researcher. I highly appreciate the extra effort he put in the early phase of my study to discuss the technical details with me and help me improve my writing. I also highly appreciate the trust he has put in me in the later phase of my study to develop my own research ideas and take more lead in my projects. Besides academic growth, Kristian also cares about each of us as a person and he is always open to communicating. This has positively influenced me in my way of interacting with others and dealing with interpersonal relations. I have learnt so much from him.

I want to thank Prof. Dr. Stefano Teso as well for the interesting discussions we had and for the nice atmosphere he always brings to the work environment. It has always been a pleasure to talk with him. Besides, I highly appreciate that he has always been very responsive and supportive.

My colleague Alejandro Molina has also given me so much guidance and support as a senior researcher. He has also been a dear friend to me, encouraging and helping me when I was feeling lost or frustrated. I would not have made it this far with my study if he was not there. So I would like to thank him as well.

I also want to thank all my other co-authors, Antonio Vergari, Karl Stelzner, Zhongjie Yu, Arseny Skryagin, Tjitze Rienstra, Matthias Thimm, Robert Peharz, Martin Trapp, Thomas Liebig, Patrick Schramowski and Wolfgang Stammer, Andrea Galassi, Marco Lippi, Paolo Torroni for all the inspiring discussions and great teamwork we had. Many of them are not only colleagues, but also friends to me.

Finally, I want to thank my family and friends for sharing both my joy and sorrow with me. Besides, I also appreciate the mental support of my boyfriend Philippe. I would not have pulled through all the hard times without them.

Abstract

Artificial Intelligence (AI) has made a huge impact on our everyday lives. As a dominant branch of AI since the 1990s, Machine Learning (ML) has been applied to a wide range of scenarios, including image recognition, speech recognition, fraud detection, recommendation systems, time series prediction and self-driving cars. Deep learning, backed up by Deep Neural Networks (DNNs), is a major subfield of machine learning. DNNs are good at approximating smooth functions, i.e., learning a mapping from inputs to outputs, which is also known as the predictive or supervised learning approach. Sometimes, one is not interested in a specific predictive task, but rather in finding interesting patterns in the data. In this case, a descriptive or unsupervised learning approach is needed, and the task can be formalized as density estimation. Deep probabilistic models have gained popularity for density estimation because they maintain a good balance between expressivity and tractability, whereas classical probabilistic models face an inherent trade-off.

Deep neural networks and deep probabilistic models are both deep models in the sense that they are composed of multiple layers of computation units. They are essentially computation graphs and consequently, it is hard for humans to understand the underlying decision logic behind their behavior. Despite the representational and predictive power deep models have demonstrated in many complex problems, their opaqueness is a common reason for concern. In this thesis, we provide insights into deep models using high-level interpretations and explanations of why particular decisions are made.

Explanations that contradict our intuitions or prior knowledge on the underlying domain can expose a potential concern, which may imply some desiderata of ML systems are not met. For example, a deep model may obtain high predictive accuracy by exploiting a spurious correlation in the dataset, which can lead to a lack of robustness, or unfairness if the spurious correlation is linked to a protected attribute. Built on the framework of Explanatory Interactive Machine Learning (XIL), we propose to interactively improve deep models based on the explanations we get. This way, we put users in the training loop and take user feedback on explanations as additional training signals. As an effect, the model can learn the rules that align with our intuitions or prior knowledge.

List of Symbols

x	A scalar
X	A random variable
$\mathbf{x}$	A vector of scalar
$\mathbf{X}$	A vector of random variables
$\boldsymbol{X}$	A matrix
x_i	Element i of vector $\mathbf{x}$
$x_{\setminus i}$	All elements of vector $\mathbf{x}$ except for element i
$\mathbf{x}^{(i)}$	The i-th example from a dataset as input
$\mathbf{y}^{(i)}$	The i-th example from a dataset as output
$\mathbf{x}^k$	The input for class k
$\mathbf{y}^k$	The output for class k
X_{ij}	Element in i-th row and j-column of matrix $\boldsymbol{X}$
$\mathbb{R}$	The set of real numbers
$\mathbb{A} \setminus \mathbb{B}$	The set of $\mathbb{A}$ subtracted by the set of $\mathbb{B}$
$\mathcal{G}$	A graph
$\mathcal{D}$	A set
$f(\mathbf{X}; \boldsymbol{\theta})$	A function of $\mathbf{X}$ parameterized with $\boldsymbol{\theta}$
$\|\mathbf{x}\|_1$	L^1 norm of $\mathbf{x}$

$\|\mathbf{x}\|_2$ L^2 norm of $\mathbf{x}$

$\|\mathbf{x}\|_F$ Frobenius norm of $\mathbf{x}$

$|\ x\ |$ Absolute value of x

$\sum_{Y_i Y_j \cdot \mathbf{X}}$ A partial cross-covariance operator of (Y_i, Y_j)

$\log(x)$ Natural logarithm of x

$\exp(x)$ Exponential of x

$\boldsymbol{X}^{-1}$ Inverse of matrix $\boldsymbol{X}$

$\boldsymbol{X}^{\mathrm{T}}$ Transpose of matrix $\boldsymbol{X}$

$\nabla_x y$ Gradient of y with respect to x

$\frac{\partial y}{\partial x}$ Partial derivative of y with respect to x

$\int f(x) dx$ Definite integral over the domain of x

$P(X)$ Probability distribution over random variable X

$P(\mathbf{X})$ Joint probability distribution over random variables $\mathbf{X}$

$p(X)$ Probability mass function (PMF) of discrete random variable X or probability density function (PDF) of continuous random variable X

$p(\mathbf{X})$ Joint probability mass function of discrete random variables $\mathbf{X}$ or a joint probability density function of continuous random variables $\mathbf{X}$

$P(\mathbf{X} \mid \mathbf{Y})$ Conditional probability distribution of random variables $\mathbf{X}$ given $\mathbf{Y}$

$\mathbb{E}[X]$ Expected value of a variable X

$\mathbb{E}_X[L]$ Expected value of L under distribution $P(X)$

$\mathcal{I}(X, Y)$ Mutual information between variable X and Y

$X \sim P$ Random variable X follows distribution P

$\mathbb{D}_{\mathrm{KL}}[P(X) \parallel P(Y)]$ KL divergence between distribution $P(X)$ and $P(Y)$

$X_i \perp\!\!\!\perp X_j$ The variable X_i and X_j are independent

$Y_i \perp\!\!\!\perp Y_j \mid X$ The variable Y_i and Y_j are conditionally independent given X

Contents

List of Figures xv

List of Tables xvii

List of Algorithms xix

I. Fundamentals 1

1. Introduction 3
1.1. Interpreting and Explaining Deep Models . . . 4
1.2. Debugging Deep Models . . . 6
1.3. Outline and Summary of Contributions . . . 7

2. Explainable AI 11
2.1. Explainable Methods . . . 13
2.1.1. Local Approaches . . . 13
2.1.2. Global Approaches . . . 19
2.2. Learning from Explanations . . . 20
2.2.1. Passive Learning . . . 21
2.2.2. Explanatory Interactive Machine Learning (XIL) . . . 24
2.3. Caveats of Explainable AI . . . 28

II. Deep Models 31

3. Deep Neural Networks 35
3.1. Model Representation . . . 35
3.2. Learning . . . 37
3.3. Regularization . . . 38
3.4. Variational Auto-Encoder . . . 40

4. Sum-Product Networks **43**
4.1. Network Polynomial . . . 43
4.2. From Network Polynomial to SPNs . . . 44
4.3. Inference . . . 47
4.4. Learning . . . 49
4.5. Random and Tensorized SPNs . . . 51
4.6. The Connections between SPNs and Other Models . . . 52

5. Conditional Sum-Product Networks **57**
5.1. Conditional Probabilistic Models . . . 57
5.2. Representation . . . 59
5.3. Learning . . . 61
5.4. Empirical Evaluation . . . 64
5.4.1. Traffic Prediction . . . 65
5.4.2. Conditional Density Estimation . . . 66
5.4.3. Neural Conditional Sum-Product Networks with Random Structures 67

III. Explaining and Debugging Deep Models **69**

6. Imposing Interpretable Structure **75**
6.1. Modular Probabilistic Modelling via Conditional Sum-Product Networks . 77
6.2. Sum-Product Variational Auto-Encoders . . . 78

7. Explaining SPNs by Counterfactual Examples **85**
7.1. Related Work . . . 87
7.2. Sum-Product networks Interpretation by Counterfactual Examples . . . 88
7.3. Empirical Evaluation . . . 91
7.3.1. Visual Inspection of Counterfactual Examples . . . 93
7.3.2. Density Evaluation of Counterfactual Examples . . . 95
7.3.3. Computation Time . . . 97
7.3.4. Effectiveness . . . 97

8. Right for the Right Features **101**
8.1. Motivation . . . 101
8.2. Right for Better Reasons . . . 103
8.3. Empirical Evaluation . . . 106
8.3.1. Dataset . . . 107

8.3.2. Experiment Protocol . . . 108
8.3.3. Adversarial Robustness . . . 108
8.3.4. RBR Learns the Right Rules . . . 109
8.3.5. Effectiveness in High-Dimensional Domains . . . 111

9. Right for the Right Latent Factors **117**
9.1. Motivation . . . 117
9.2. Disentangled Representation . . . 119
9.3. Disentangled Representation for Generative Models . . . 120
9.3.1. Disentangling via User Feedback . . . 121
9.3.2. Analysis of Property Guarantees . . . 122
9.3.3. Interactive Data Augmentation . . . 124
9.4. Empirical Evaluation . . . 125
9.4.1. Dataset . . . 126
9.4.2. Experimental Protocol . . . 127
9.4.3. Reconstruction Quality . . . 127
9.4.4. Latent Space Arithmetic . . . 128
9.4.5. Latent Space Traversal . . . 130
9.4.6. Disentanglement Metrics . . . 131
9.4.7. Downstream Accuracy . . . 132

IV. Conclusion **135**

10. Conclusion **137**
10.1. Summary . . . 137
10.2. Lessons Learned . . . 139
10.3. Outlook . . . 140

11. Selected Papers and Contributions **143**

List of Figures

1.1. Illustration of the Trade-off between Accuracy and Interpretability 5

2.1. Saliency Maps 15
2.2. Counterfactual Explanations 17

3.1. The Architecture of Feedforward Networks 36
3.2. The Directed Graphical Model behind VAEs 42

4.1. A Bayesian Network Example 44
4.2. A Canonical Polynomial Example 45
4.3. A Factored Polynomial Example 45
4.4. A SPN Example 47
4.5. The SPN Structure for Classification 53
4.6. An Example of SPN Augmentation 54
4.7. Bayesian Network Representing Dependencies in SPNs 55

5.1. A CSPN Example 60
5.2. Residual Plots of Traffic Prediction 65

6.1. A Bayesian Network and a Corresponding Sum-Product Network 76
6.2. Graphical Illustration of ABCSPNs 77
6.3. Samples Generated by an ABCSPN 77
6.4. Graphical Model of SPVAE 79
6.5. Prior Distributions for SPVAE 81
6.6. Image Reconstructions from VAEs and SPVAEs 82
6.7. Quantitative Evaluation for SPVAEs 83

7.1. Illustration of SPICE 86
7.2. Intuition of SPICE on 2D Datasets 90
7.3. Counterfactual Examples on MNIST 93
7.4. Counterfactual Examples on Caltech-UCSD Birds 94

8.1. Illustration of a Confounding Factor . . . 102
8.2. Explanatory Interactive Learning . . . 103
8.3. Robustness to Adversarial Perturbations . . . 109
8.4. Human Feedback on Machine Explanations . . . 110
8.5. Right for the Right Reasons on the Toy Color Dataset . . . 111
8.6. Machine Explanations on the Decoyed MNIST Dataset . . . 112
8.7. User Feedback and the Counter Examples . . . 114
8.8. Right for the Right Features on PASCAL VOC 2007 . . . 115
8.9. Right for the Right Reasons on the Plant Dataset . . . 116

9.1. Illustration of the Purpose of the Disentanglement Losses . . . 122
9.2. Illustration of the Disentanglement Losses . . . 123
9.3. Reconstructed Images from VAEs . . . 128
9.4. Combinatorial Generalization of Disentangled Representation . . . 129
9.5. Cross-Product of Latent Factors . . . 129
9.6. Latent Space Traversal . . . 130
9.7. Evaluation of Disentanglement Metrics . . . 133
9.8. Accuracies of Downstream Classifiers of Latent Representations . . . 134

List of Tables

5.1. Conditional Log-Likelihood of DACL, CCNs and CSPNs 66
5.2. Conditional Log-Likelihood of MFs, MDNs and CSPNs 67

7.1. Density Evaluation of Counterfactual Examples 96
7.2. Counterfactual Perturbations for the German Credit Dataset 97
7.3. Average Computation Time of Counterfactual Examples 98
7.4. Success Rates of Counterfactual Examples 98

8.1. Classification Accuracies on the Counter Examples and the Random Examples113

9.1. Properties of Disentangling Methods 125
9.2. Summary of the Datasets 126

List of Algorithms

1. Explanatory Interactive Learning 26
2. Stochastic Gradient Descent Update 39
3. LearnSPN 50
4. SPN Augmenting Algorithm 56
5. Feature Splitting Algorithm for CSPNs 63
6. LearnCSPN 64
7. Differentiable Sampling Algorithm for CSPNs 80
8. Obtaining Counterfactual Examples for RAT-SPNs 92
9. Right for Better Reasons 104

Part I.

Fundamentals

1. Introduction

Nowadays, Artificial Intelligence (AI) has received an incredible amount of attention from both academia and industry. Coined by John McCarthy in 1956, the term AI refers to "the science and engineering of making intelligent machines, especially intelligent computer programs" [142]. Since the 1990s, Machine Learning (ML) has become a dominant branch in AI. ML refers to a set of methods that aim to automatically detect patterns in data, which can in turn facilitate automated decision making [158]. Applications of machine learning range from image recognition [236], speech recognition [248, 74], fraud detection [138, 227, 165], recommendation systems [253, 3], time series prediction [79] to self-driving cars [16]. After AlexNet [113] won the ImageNet Large Scale Visual Recognition Challenge by a phenomenally wide margin in 2012, deep learning has boomed and remains a major subfield of machine learning until today. Loosely inspired by neuroscience, artificial neural networks are the backbone of deep learning [69]. Another promising subfield of machine learning is probabilistic reasoning, which provides a formalism for automated reasoning in the presence of uncertainty [167]. Unlike neural networks, probabilistic models allow for a full modelling of the domain instead of a specific input-output mapping, which in turn enables reasoning beyond one specific target variable. Among probabilistic models, probabilistic circuits have gained popularity because they maintain a good balance between expressivity and tractability, whereas classical probabilistic models face an inherent trade-off.

Both probabilistic circuits and neural networks are deep models in the sense that they are computation graphs composed of multiple layers of computation units. They provide scalable solutions for specific tasks by learning heavily parameterized but unstructured models. Despite their success on specific tasks in recent years [113, 80, 67, 134, 245, 55], deep models still face doubts and critics in many ways. A frequently used argument against deep models is that they lack interpretability and explainability. Deep models are to be contrasted with conventional machine learning approaches, which are inherently easy to interpret but lack representational and predictive power to deal with complex problems without prior feature engineering. Take a linear regression model as an example. It learns a linear relation between features and the outputs. The model's weights associated with each feature determine directly their contribution to the prediction, which can easily

be understood by humans, assuming the number of features remains a reasonable size and the features correspond to intuitive concepts. However, such models are limited to learn only linear relations. Other conventional machine learning models are interpretable in different ways. Naive Bayes may be interpreted by their assumption of conditional independence of the features given the class label [151]. Decision trees may be interpreted by their feature splitting rules. However, interpretability is not a binary concept but a spectrum. For instance, decision trees can in fact become less interpretable when their model size grows. This is due to the finite span of our immediate memory [146, 35]. In general, there is an intuitive trade-off between predictive accuracy and interpretability which is illustrated in Figure 1.1.

Interpretability is desired due to the following reasons:

1. It increases the social acceptance of black-box models for daily uses [151].
2. It is helpful for humans to gain knowledge and insights about the underlying domain [48, 151].
3. It can be used to verify auxiliary criteria, other than expected task performance, of machine learning systems [48].

The desire for interpretability of black-box models has not only raised attention from the research community, but also from the public. For instance, the General Data Protection Regulation (GDPR) [246] that came into effect in 2018 requires automated decision-making systems that legally affect EU residents to be able to explain themselves.

1.1. Interpreting and Explaining Deep Models

The definition of interpretability has largely been based on intuitions. One of the most popular definitions is that interpretability is *the degree to which a human can understand the cause of a decision* [14, 147]. Another popular definition is, that it is *the ability to explain or to present in understandable terms to a human* [48]. Interpretability and explainability are often used interchangeably. However, Gilpin et al. [66] and Arrieta et al. [8] explicitly argue that these two concepts should be distinguished. Rudin posits that interpretability is conferred by models that are considered inherently interpretable by construction, and explainability refers to the ability to explain the black-box model based on another post-hoc model [199]. In this work, we follow the terminology adopted by Rudin.

In terms of scope, interpretability and explainability can be considered at a global level and a local level. Global approaches focus on the entire behaviour of a model, which

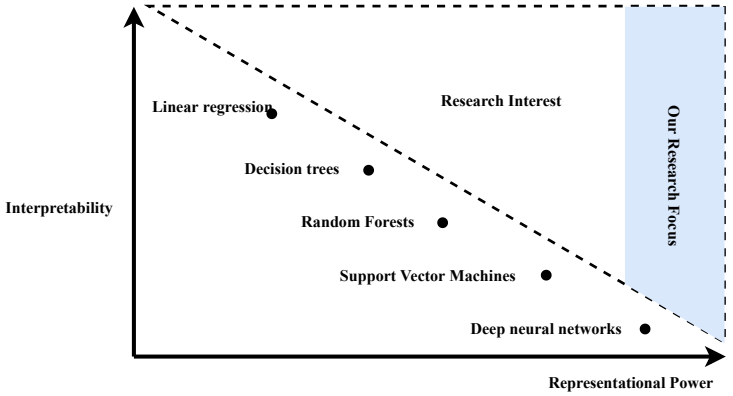


Figure 1.1.: Illustration of the trade-off between accuracy and interpretability, adapted from [251].

can give general insights into the domain. Local methods are supposed to improve our understanding of why a specific decision is made.

Deep models are not inherently interpretable, both on the global level and on the local level. This is attributed to the fact that deep models lack rich structure with semantics and their architecture is typically too large. In this thesis, we are interested in increasing the interpretability and explainability of deep models on both the global level and the local level, whereby we put our emphasis on post-hoc explanations but also touch upon intrinsic interpretability. To induce more interpretable high-level structures for deep models, we impose modular structures on high-dimensional joint distributions. In particular, we propose to use tractable conditional probability distributions as building blocks for joint probabilistic models. Modularity can be formed using the chain rule in probability theory, which can in turn help us obtain a high-level interpretation of the underlying domain and enable the integration of background knowledge or assumptions in the form of conditional independencies in highly complex spaces. As such, deep models are beginning to explore the spectrum between interpretability and representational power. Sometimes a general interpretation of the domain is insufficient and more specific explanations based on the situation are required. For example, knowing that the success of a loan application depends on income, deposit, age and housing state etc. is not as helpful as a specific explanation such as “your application is rejected because your income is too low”. Therefore, we also delve into local explanations for deep models to understand individual inferences.

1.2. Debugging Deep Models

Deep models are good at learning for specialized tasks from a large quantity of data. They do so by optimizing an objective function as performance metric on a task. However, other important criteria, such as fairness, privacy, reliability, robustness, causality, usability, and trust, of machine learning systems can be hard to formalize, which in turn makes optimization and validation hard [48]. Explanations can help us to verify auxiliary criteria, apart from giving insights. In particular, explanations that contradict our intuitions or prior knowledge on the underlying domain can expose a potential concern about the model, which may imply some desiderata are not met [48]. For example, by using Layer-wise Relevance Propagation (LRP) [9] to explain a Fisher vector classifier trained on the PASCAL VOC 2007 dataset [54], Lapuschkin et al. [117] found that this model actually focuses on a source tag present in about one-fifth of the horse figures. This is of course not desired, because the source tags are not the goal of classification, and these spurious correlations between some features and the prediction target probably do not generalize in a product environment. The reliance on such spurious correlations is typically referred to as the Clever Hans phenomenon in psychology [173, 117]. The acquired spurious correlations are a type of model bug caused by the contamination in the learning pipeline [2]. Adebayo et al. [2] categorize model bugs into the following types based on their source:

1. Data contamination bugs are caused by problems in the training data such as a spurious correlation signal.

2. Model contamination bugs are caused by problems in the model parameters.

3. Test-time contamination bugs are caused by problems in test examples, for example, domain shift at test time.

The ultimate goal of diagnosing model bugs is to correct the bugs. Diagnosing and correcting model bugs — model debugging — is especially important as automated systems are being deployed in high-stakes domains [2, 13, 84, 144]. In this work, we restrict our attention to data contamination bugs that originate from the learning pipeline. In this case, debugging defective models requires improving their learning pipeline. Therefore, we ponder the question of whether explanations could be effectively used as debugging tools to improve defective models during the learning stage. Assuming explanations can faithfully reveal the bugs in the model, debugging amounts to adapting the model to generate explanations that align with user intuitions or knowledge. In the standard supervised learning setting, humans guide models to learn by providing annotated labels

on each example. This information is however very restricted and does not contain much complexity. In some cases, ambiguity may even arise. Take again the aforementioned example, the source tags exclusive to the horse images in the PASCAL VOC 2007 dataset make the labels for horse images and non-horse images ambiguous. Take another example, if a classifier is trained to distinguish cats and dogs, but the training examples contain only black cats and white dogs, then a single label for each image is obviously ambiguous for this task. A direct consequence is that the models may exhibit the Clever Hans phenomenon or learn entangled representations. However, explanations may encode more information and be used as the training signal for imposing additional constraints. In this thesis, we present various approaches to guide a variety of deep models using feedback on the explanations provided by users, in addition to the labels, with the aim of learning models that align with user explanations. We build our solutions using the explanatory interactive learning framework [237] where the user and the machine can interact via the explanations inside a training loop.

1.3. Outline and Summary of Contributions

This thesis is driven by the following three high-level scientific questions:

(Q1) How to improve the interpretability of deep networks on the global level?

(Q2) Can deep probabilistic models be explained?

(Q3) Can explanations be used to diagnose and correct model bugs?

This thesis is organized into three parts. In the first part, we prepare the background knowledge on explainable AI. Then in the second part, we present some deep models that will be employed later. These deep models include both neural networks and probabilistic models. In the third part, we show how to interpret and explain deep models. Ultimately, we provide solutions to interactively revise deep models to be right for the right reasons or latent factors. The respective chapters can contain verbatim quotes from the corresponding publications. A more detailed contribution for each chapter is given as follows.

Chapter 2 first gives an overview of the mainstream post-hoc explanation methods for deep models, including both local approaches and global approaches. Based on the introduced explanation methods, we present some approaches to revise the models using user feedback on explanations. Then we have some general discussion about explainable AI.

Chapter 3 presents deep neural networks by their model representation and learning routines. Then some common regularization techniques are reviewed, some of which

can potentially induce more interpretable models as a side effect. In the end, a neural generative model is presented, which will be used in part III.

Chapter 4 introduces sum-product networks (SPNs), which are another type of deep models we want to interpret in part III. This chapter covers the model representation, inference and learning of SPNs, followed by a scalable variant of SPNs. In the end, some perspectives on SPNs are given, which can be used to interpret SPN architectures by drawing connections to classical probabilistic models.

Chapter 5 introduces conditional sum-product networks (CSPNs). This is a conditional counterpart for sum-product networks (SPNs) to represent conditional probability distributions. Inheriting the advantages of SPNs, CSPNs achieve a good balance between expressiveness and tractability. Besides, they maintain — at least on a high level — an interpretable domain structure. More specifically, CSPNs can be used as building blocks to impose a rich structure on high-dimensional joint distributions. This chapter is based on the following publications:

> Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. "Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures". In: *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*. 2020

> Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. "Conditional Sum-Product Networks: Modular Probabilistic Circuits via Gate Functions". In: *International Journal of Approximate Reasoning* 140 (2022), pp. 298–313

Chapter 6 demonstrates the effectiveness of CSPNs in building modular structures for joint probabilistic models. The modular structures give high-level interpretability to deep models, which helps us with domain understanding and integrating domain assumptions. This work is related to the scientific question **(Q1)**. This chapter is based on the same publications as the previous chapter.

Chapter 7 presents a novel approach for explaining individual inferences of SPNs by generating counterfactual examples, which is essentially a generative task. Compared to the counterfactual literature that mainly focuses on neural networks, the big advantage of studying SPNs is that the underlying density distributions are naturally useful for generative tasks. Empirical study will show the effectiveness and efficiency of this approach. This work is related to the scientific question **(Q2)**. This chapter is based on the following work:

> Xiaoting Shao and Kristian Kersting. "Gradient-Based Counterfactual Explanations Using Tractable Probabilistic Models". In: *arXiv preprint arXiv:2205.07774* (2022)

Chapter 8 presents our work on interactively learning from user feedback on explanations. Built on the work of Ross et al. [198], we teach the models to effectively learn the right features by using user feedback on influence functions. This way, the models do not only learn the right labels, but also learn the right features that are responsible for the labels. We show with empirical evidences that this approach is very effective at counteracting the Clever Hans effect, and improving adversarial robustness. This work is related to the scientific question **(Q3)**. This chapter is based on the following publications:

> Xiaoting Shao, Arseny Skryagin, P Schramowski, W Stammer, and Kristian Kersting. "Right for Better Reasons: Training Differentiable Models by Constraining their Influence Function". In: *Proceedings of 35th AAAI Conference on Artificial Intelligence (AAAI)*. 2021
>
> Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. "Making Deep Neural Networks Right for the Right Scientific Reasons by Interacting With Their Explanations". In: *Nature Machine Intelligence* 2.8 (2020), pp. 476–486

Chapter 9 presents our study on the Clever Hans effect for deep generative models, specifically VAEs. We show that confounded datasets are harmful for VAEs by causing them to learn entangled representations. To improve the latent representations, user explanations are given as a small set of examples back to the model inside the training loop. Empirical evidences show the effectiveness of this approach on learning the right latent factors. This work is related to the scientific question **(Q3)**. This chapter is based on the following work:

> Xiaoting Shao, Karl Stelzner, and Kristian Kersting. "Right for the Right Latent Factors: Debiasing Generative Models via Disentanglement". In: *arXiv preprint arXiv:2202.00391* (2022)

Chapter 10 concludes this thesis, and points out some interesting directions to explore in future work.

Apart from the publications mentioned above, there are additionally the following publications that are relevant but not expanded in this thesis:

> Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. "Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning". In:

Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI). ed. by Ryan P. Adams and Vibhav Gogate. Vol. 115. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 334–344

Tjitze Rienstra, Matthias Thimm, Kristian Kersting, and Xiaoting Shao. "Independence and D-separation in Abstract Argumentation". In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 17. 1. 2020, pp. 713–722

Xiaoting Shao, Zhongjie Yu, Arseny Skryagin, Tjitze Rienstra, Matthias Thimm, and Kristian Kersting. "Modelling Multivariate Ranking Functions With Min-Sum Networks". In: *Proceedings of the 14th International Conference on Scalable Uncertainty Management*. Springer. 2020, pp. 281–288

Andrea Galassi, Kristian Kersting, Marco Lippi, Xiaoting Shao, and Paolo Torroni. "Neural-Symbolic Argumentation Mining: An Argument in Favor of Deep Learning and Reasoning". In: *Frontiers in big Data* 2 (2020), p. 52

2. Explainable AI

The topic about explanations can be found in vast and valuable bodies of research in philosophy, psychology, and cognitive science [147]. In the field of psychology, explanations are the currency in which we exchange beliefs, as noted by Lombrozo [133]. Miller argues that explanation involves cognitive process and social process: The cognitive process in explanation is the process of determining an explanation for a given event by identifying the causes of the event and selecting a subset of these causes as the explanation. The social process in explanation is the process of transferring knowledge between explainer and explainee, with the purpose of providing the explainee with enough information to understand the causes of the event [147].

In the field of AI, there is a surging demand for explanations for intelligent autonomous systems. For instance, taking effect as a law across the European Union (EU) in 2018, GDPR requires that individuals affected by algorithmic decisions have a right to ask for an explanation of an algorithmic decision that was made about them [72]. What form such an explanation may take and how such an explanation could be proven correct remain to be researched [127]. Fortunately, the surging demand has given rise to increasing research attention within the field of *Explainable Artificial Intelligence* [78], abbreviated XAI.

AI, especially ML, has gained a lot of success and attention in the last decades and revolutionized our way of life, from super-human performance in visual pattern recognition to astounding achievements in natural language processing and intelligent agents. However, current ML systems are also very fragile.

First, ML models are susceptible to adversarial perturbations [71, 23, 18, 252]. One study points out that adversarial examples are features instead of bugs [96], which implies the fact that machines learn different relations in the data as we humans do. This adversarial behavior of AI systems seems counter-intuitive to us because humans are inclined to humanize machines and think they make decisions the same way as we do. Therefore machine explanations can give us an objective understanding of the machinery of AI systems.

Second, ML systems may reinforce social stereotypes or exhibit bias towards a certain group of people. Statistical translation tools such as Google Translate can exhibit gender

biases and a strong tendency towards male defaults: By translating professional-related sentences such as “He/She is an engineer” from gender-neutral languages into English, the male defaults are found to be exaggerated in fields associated with gender stereotypes, such as STEM (Science, Technology, Engineering and Mathematics) occupations [176]. An automated commercial facial recognition tool sold by major tech companies shows substantial disparities in its performance on faces with different skin-tone and different gender. Specifically, they perform overall best for individuals with a lighter skin tone and males, and worst for females with dark skin [19].

The brittleness of ML models can cause severe consequences, e.g. when deployed in the clinical domain. In a famous example, a ML model predicted a lower risk of pneumonia for patients with asthma and a higher risk otherwise [24]. This counter-intuitive behavior is explained by the fact that the model learns to pick up the correlations between patients with asthma and their fatality rate [24]. The fatality rate of asthma patients is lower because asthma patients tend to seek medical help earlier than other patients but obviously their inherent risk is higher. Consequently, this model would make systematically wrong predictions which could potentially bring real harm to those risk patients [24]. Another example is a class of commercial risk-prediction tools for identifying patients who will derive the greatest benefit from the “high-risk care management” programs. The tools, employed to approximately 200 million people in the United States each year, are found to exhibit significant racial bias: Black patients are much sicker than White patients at a certain predicted risk level, as evidenced by signs of uncontrolled illnesses [161].

Trust is the cornerstone of major theories of interpersonal relationships in psychology [223, 89]. Similarly, trustworthiness is also a desirable property for machines to have. However, it is hard for machines to gain trust or social acceptance if there are significant consequences for incorrect results, or the problem has not been sufficiently studied and validated in real-world applications [48]. Due to the aforementioned counter-intuitive behaviour and brittleness of ML systems, potential fairness and safety issues that come along with it, the desire for improvements naturally arises. Explanations can help in validating ML systems, which in turn guide the systems to improve and boost human trust. Explainable AI is one of the sub-fields that augment the current ML systems, which is of vital importance for combating these issues. Dating back to the early 1990s, a great number of publications on explainable AI initially emerged [90, 57, 199]. This chapter, however, deals primarily with the state-of-the-art research and not with the historical background.

In the remainder of this chapter, we will focus on recent work on explaining black-box ML models and give an overview of a number of state-of-the-art techniques in Section 2.1. Subsequently, a collection of research work on using explanations to improve ML models will be reviewed in Section 2.2 based on the explanation methods introduced in Section

2.1.

In the following, we assume N input-label pairs $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{i=N}$ in the training set where $\mathbf{x}^{(i)} \in \mathbb{R}^D$ and $\mathbf{y} = f(\mathbf{x})$. Our goal is to explain a learner $\hat{f}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ parameterized by $\hat{\boldsymbol{\theta}}$ which approximates the target function $f(\cdot)$ and yields $\hat{\mathbf{y}}$ as its prediction for $\mathbf{x}$, i.e. $\hat{\mathbf{y}} = \hat{f}(\mathbf{x}; \hat{\boldsymbol{\theta}})$.

2.1. Explainable Methods

In this section, we discuss post-hoc techniques for explaining machine learning models. Due to the quickly expanding volume of research in XAI, we do not enumerate all the methods. Instead, we focus on the mainstream explainable methods in deep neural architectures. As mentioned previously, explainable methods can be categorized as local approaches and global approaches. Local approaches aim to explain the model in a local context, for example explaining the decision on a particular instance [188, 10, 81, 206, 221, 122]. Global approaches are designed to study the aggregated behaviour that goes beyond single instances. In particular, they aim to reveal insights and improve overall understanding of a set of examples, an entire class of interest or the entire model [104, 58, 7].

We first review a variety of local approaches in Subsection 2.1.1, which concludes with a unifying framework. Then we go through a few global approaches in Subsection 2.1.2.

2.1.1. Local Approaches

Perturbation-based: Local Interpretable Model-agnostic Explanations (LIME) [188] is a modular and extensible method that interprets the decision for a sample $\mathbf{x}^{(i)}$ based on a linear model $\hat{g}(\cdot)$, which approximates the decision boundary of the original model $\hat{f}(\cdot)$ around a local vicinity. In order to induce the linear model $\hat{g}(\cdot)$, the following objective function is minimized

$$\xi = \underset{\hat{g} \in \mathcal{G}}{\arg\min}\, L(\hat{f}, \hat{g}, \pi_{\mathbf{x}}) + \Omega(\hat{g}), \tag{2.1}$$

where $L(\hat{f}, \hat{g}, \pi_{\mathbf{x}})$ denotes the approximation loss over a set of local samples weighted by kernel $\pi_{\mathbf{x}}$, and $\Omega(\hat{g})$ regularizes the complexity of $\hat{g}$.

The main advantages of LIME are threefold. Firstly, it yields interpretable explanations to a wider audience using the weights of linear models [188]. Secondly, it promotes local fidelity, i.e. the explanations correspond to the model's underlying behaviour in the vicinity of the selected instance [188]. This is a desirable property that was formally defined as local accuracy by Lundberg et al. [137]. Thirdly, it is agnostic to the underlying

models because the model's predicted outputs for a set of samples are sufficient for the computation of LIME, regardless of the internal mechanism of the predictive models [188]. The model agnosticism in turn makes it flexible to use in a wide variety of problems. However, the reliance on sampling process makes the method computationally heavy to use. Moreover, LIME explanations are not stable when applied to complex models, meaning that the explanations can vary significantly for neighbouring inputs [5]. In addition, it is hard to know how far one can extrapolate the local model for the explanations [151].

Perturbation-based: Influence Functions [111] are used to understand a model's prediction through the lens of its training data by attributing a given prediction to training points that are most responsible for it. According to Cook et al. [34], the influence of upweighting an instance $\mathbf{z}$ on the parameters $\hat{\boldsymbol{\theta}}$ by a small amount can be efficiently approximated by

$$\mathcal{I}_{\text{up, params}}(\mathbf{z}) \coloneqq -\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}}), \tag{2.2}$$

where $\mathbf{z} = (\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ and $L(\mathbf{z}, \hat{\boldsymbol{\theta}})$ is the loss.

Koh et al. [111] derived a finer-grained version of Equation 2.2, which is given by

$$\mathcal{I}_{\text{pert, loss}}(\mathbf{z}, \mathbf{z}_{\text{test}}) \coloneqq -\nabla_{\boldsymbol{\theta}} L(\mathbf{z}_{\text{test}}, \hat{\boldsymbol{\theta}})^{\mathrm{T}} \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}}). \tag{2.3}$$

The statistic in Equation 2.3 approximates the influence of perturbing the instance $\mathbf{z}$ on a specific test instance $\mathbf{z}_{\text{test}} = (\mathbf{x}_{\text{test}}^{(i)}, \mathbf{y}_{\text{test}}^{(i)})$. In order to efficiently calculate this statistic, especially inverting the Hessian matrix $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1}$, it is reformulated as $-\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}_{\text{test}}, \hat{\boldsymbol{\theta}}) \cdot \nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}})$ where $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}_{\text{test}}, \hat{\boldsymbol{\theta}})$ is the implicit Hessian-vector products trick to prevent explicitly computing $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1}$. Koh et al. [111] have demonstrated several use cases of influence functions. Firstly, it can be used to interpret the model's underlying behavior by identifying the most helpful or harmful training instances. Secondly, it can reveal insights when the training distribution does not match the test distribution, i.e. domain mismatch. Thirdly, by spotlighting the most influential training point, it allows humans to inspect and correct mislabeled training instances.

Gradient-based: Input Gradient [10] explains the prediction for a particular instance in a similar scheme as LIME by approximating the local decision boundary in the vicinity of the instance. It computes the gradient of the model's output with respect to its inputs, i.e.

$$\eta_{\hat{f}}(\mathbf{x}) \coloneqq \bigtriangledown_{\mathbf{x}} \hat{\mathbf{y}}, \tag{2.4}$$

where $\eta_{\hat{f}}(\mathbf{x}) \in \mathbb{R}^{N \times D}$ by construction. For each instance $\mathbf{x}^{(i)}$, the input gradient yields a vector of D dimensions which indicates the direction of a local perturbation that induces the steepest transition in the output. For the purpose of explanation, the features associated

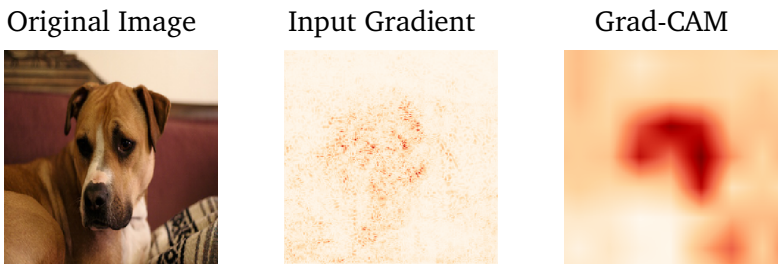


Figure 2.1.: Saliency maps on an image from PASCAL VOC 2007. The explained classifier is the VGG-16. The input gradient is noisier than the Grad-CAM.

with the bigger absolute values in the input gradient vector are considered more salient for a particular input $\mathbf{x}^{(i)}$. Input gradients align with sample-based methods such as LIME but they are much faster to compute [198]. In addition, input gradients also guarantee local fidelity and they are generally applicable for differentiable models [198]. However, this approach can yield quite noisy explanations, that is, it does not always capture salient features when used to generate explanations because the gradient becomes increasingly smaller when the instance moves further away from the decision boundary [198, 219, 9, 152, 56].

Gradient-based: Gradient-weighted Class Activation Mapping (Grad-CAM) [206] explains decisions of the output layer of Convolutional Neural Networks (CNNs) for a particular instance using the gradient w.r.t. the feature maps of the last convolutional layer. The importance weights of feature map c for target class k are formally formulated as

$$\alpha_c^k := \frac{1}{Z} \sum_i \sum_j \frac{\partial y^k}{\partial A_{ij}^c}, \tag{2.5}$$

where Z denotes the number of pixels in the feature map, y^k denotes the output for class k and $\boldsymbol{A}^c$ denotes feature map activations of a convolutional layer. i and j denote width and height dimensions respectively. Grad-CAM $L^c_{\text{Grad-CAM}}$ is in turn defined as the weighted combination of forward activation filtered by Rectified Linear Unit (ReLU) to account only for a positive effect on the class of interest, i.e.

$$L^c_{\text{Grad-CAM}} = \text{ReLU}(\sum_c \alpha_c^k \boldsymbol{A}^c). \tag{2.6}$$

Grad-CAM is developed based on the following two facts about CNN architectures:

1. Feature representations learnt in deeper layers in a CNN capture higher-level abstractions of the visual features [206, 12, 139].

2. Convolutional operations excel at capturing spatial information [206].

These two facts imply that the last convolutional layer is supposed to have the optimal balance between high-level semantics and detailed spatial information [12, 139].

Akin to input gradients, Grad-CAM is also based on gradients and operates on a local level. The key difference is that Grad-CAM is designed for CNNs and input gradients can be used for any differentiable networks. That means, Grad-CAM is tailored to explain image classification tasks. In addition, Grad-CAM explains an intermediate layer of deep networks while input gradients operate on the input layer. The saliency maps on the input layer are generally noisier than those on an intermediate layer. See Figure 2.1 for an example.

A Unified Approach: SHapley Additive exPlanations (SHAP) [137] provides a unified framework for interpreting model predictions that are approximated by various existing interpretation methods. This unification is based on a class of additive feature attribution methods which refer to an explanation model that is a linear function

$$g(\mathbf{z}') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i', \tag{2.7}$$

where $\mathbf{z}'$ is a vector of binary values, i.e. $\mathbf{z}' \in \{0,1\}^M$, M denotes the number of input features and $\phi_i \in \mathbb{R}$ denotes the attribution of feature i. The function in Equation 2.7 describes a class of methods that aggregate feature attributions to approximate the original function $\hat{f}(\mathbf{x})$, i.e. $g(\mathbf{z}') \approx \hat{f}(\mathbf{x})$. This class of methods includes LIME [188], DeepLIFT [219], layer-wise relevance propagation [9] and classic Shapley value estimation [125, 232, 41]. Lundberg et al. show that there is a unique additive feature importance measure in this class of methods [137].

SHAP provides a new perspective of viewing explanation methods as another simple and interpretable model, termed as the explanation model, approximating the original model [137]. In addition, the unique additive feature importance measure given by SHAP also adhere to three desirable properties — local accuracy, missingness and consistency — that were previously only tied to classical Shapley value estimation methods [137]. Local accuracy suggests that the approximated explanation model is faithful to the output of the original model [137]. Missingness restricts that absent features in the original input contribute nothing to the explanation model [137]. Consistency implies that if a model changes so that some simplified input's contribution increases or stays unchanged regardless of the other inputs, that input's attribution should not decrease [137].

Figure 2.2.: Counterfactual explanations w.r.t. the digit class 8. The explained classifier is a SPN.

Counterfactual Explanations are a novel type of explanation for automated decisions, made popular by Wachter et al. [247]. Without loss of generality, we consider a binary classifier $\hat{f}$ and a reference example $\mathbf{x}$ whose class prediction is y. A counterfactual is an alternative example $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$ that is predicted as a different class y'. Counterfactual examples have a natural connection with adversarial examples in that both of them aim to perturb the class prediction by perturbing the input. However, they are created with different purposes and have different properties: Adversarial examples aim to mislead the classifiers to yield wrong predictions, and adversarial perturbations should be so small that they are barely perceptible by the human eye. Counterfactual examples reflect counterfactual thinking in psychology, which is associated with mental representations of alternatives to past events, actions, or states [193, 21, 53]. Wachter et al. describe counterfactuals as "close possible worlds" that arrive at a different prediction y' [247]. That means, the counterfactual perturbation $\boldsymbol{\delta}$ should be small and the induced counterfactual example $\mathbf{x}'$ should be possible w.r.t. the domain.

Wachter et al. frame the task of generating counterfactual examples as the following optimization process:

$$\mathbf{x}' = \arg\min_{\mathbf{x}'} \max_{\lambda} \lambda \, \mathrm{yloss}(\hat{f}(\mathbf{x}'), y') + d(\mathbf{x}, \mathbf{x}')$$

Here, $\mathrm{yloss}(\hat{f}(\mathbf{x}'), y')$ is implemented as l_2-loss, i.e. $(\hat{f}(\mathbf{x}') - y')^2$. $d(\cdot, \cdot)$ is a distance function that measures how far the counterfactual $\mathbf{x}'$ and the reference example $\mathbf{x}$ are from each other. l_1 norm or Manhattan distance are suggested for the distance function, however, the choice should be refined depending on the specific task [247].

Following Wachter et al. [247], Mothilal et al. [156] augmented the loss with a diversity

constraint to encourage diverse solutions:

$$\begin{aligned}\mathbf{x}'^{(1)}, \ldots, \mathbf{x}'^{(k)} = \underset{\mathbf{x}'^{(1)}, \ldots, \mathbf{x}'^{(k)}}{\arg\min} & \frac{1}{k} \sum_{i=1}^{k} \text{yloss}(\hat{f}(\mathbf{x}'^{(i)}), y') \\ & + \frac{\lambda_1}{k} \sum_{i=1}^{k} d(\mathbf{x}, \mathbf{x}'^{(i)}) \\ & - \lambda_2 \, \text{diversity}(\mathbf{x}'^{(1)}, \ldots, \mathbf{x}'^{(k)}).\end{aligned}$$

Here, $\text{yloss}(\hat{f}(\mathbf{x}'^{(i)}), y')$ is implemented as a hinge-loss function $\max(0, 1 - z \cdot \text{logit}(\hat{f}(\mathbf{x}'^{(i)})))$, where z is -1 when $y' = 0$ and z is 1 when $y' = 1$. The diversity metric is built on determinantal point processes (DPP) [114], which is the determinant of the kernel matrix given the counterfactuals:

$$\text{diversity} = \det(\boldsymbol{K}),$$

where $K_{ij} = \dfrac{1}{1 + d(\mathbf{x}'^{(i)}, \mathbf{x}'^{(j)})}$. However, both of the above counterfactual approaches do not have explicit knowledge of the underlying density or data manifold, which may lead to unrealistic counterfactual examples.

Fortunately, the issue of density constraint has already been considered in a few recently proposed methods, most of which learn an auxiliary density model to impose additional density constraints on the optimization process. As a common choice for the density approximator, Variational Autoencoders (VAEs) [108] and their variants [110, 97] are used. For instance, Dhurandhar et al. [46] proposed Contrastive Explanations Method (CEM) to find contrastive perturbations, in particular, additions, that should be necessarily absent to justify the classification. CEM approaches this task by solving the following optimization problem:

$$\mathbf{x}' = \underset{\boldsymbol{\delta} \in \mathcal{X} \setminus \mathbf{x}}{\arg\min} \, c \cdot f_k^{\text{neg}}(\mathbf{x}, \boldsymbol{\delta}) + \beta \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\delta}\|_2^2 + \gamma \|\mathbf{x} + \boldsymbol{\delta} - \text{AE}(\mathbf{x} + \boldsymbol{\delta})\|_2^2.$$

f_k^{neg} is a loss function defined as

$$f_k^{\text{neg}} = \max\{[\text{Pred}(\mathbf{x} + \boldsymbol{\delta})]_t - \max[\text{Pred}(\mathbf{x} + \boldsymbol{\delta})]_i, -k\},$$

where $t = \arg\max_i [\text{Pred}(\mathbf{x} + \boldsymbol{\delta})]_i$ and $[\text{Pred}(\mathbf{x} + \boldsymbol{\delta})]_i$ is the i-th class prediction score of $\mathbf{x} + \boldsymbol{\delta}$. This loss function encourages the perturbed example $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$ to induce a different class prediction than the original prediction $y = \arg\max_i [\text{Pred}(\mathbf{x})]_i$. See Figure 2.2 for an example.

2.1.2. Global Approaches

In contrast to local explanation methods, global approaches are designed on the model-level or class-level to explain higher-level or aggregated behavior.

Testing with Concept Activation Vectors (TCAVs) [104] interprets the collective behaviour of a model for a particular class using human-friendly concepts. This is realised by quantifying the alignment of the model's sensitivity with high-level concepts which are represented as *Concept Activation Vectors* (CAVs).

Conceived and experimented in the domain of computer vision, the method of TCAVs implicitly relies on the fact that deeper representations in a CNN capture higher-level abstractions of the visual features [12, 139]. The higher-level abstraction captured in deep layers is presented as CAV, which is defined by the normal vector of the decision boundary of a binary linear classifier $\mathbf{v}_C^l$. This linear classifier is induced by training on the deep representations to distinguish the presence of a concept. In other words, CAVs are defined by

$$\begin{aligned} S_{C,k,l}(\mathbf{x}) &= \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(\mathbf{x}) + \epsilon \mathbf{v}_C^l) - h_{l,k}(f_l(\mathbf{x}))}{\epsilon} \\ &= \nabla h_{l,k}(f_l(\mathbf{x})) \cdot \mathbf{v}_C^l \end{aligned}$$

where $f_l(\mathbf{x})$ denotes the activations for input $\mathbf{x}$ at activation layer l and $h_{l,k}(\mathbf{x})$ denotes the logit for $\mathbf{x}$ for class k. $\nabla h_{l,k}(f_l(\mathbf{x}))$ can be viewed as input gradients in the latent space of layer l, which measures the sensitivity of the prediction w.r.t. the feature abstractions at layer l. $\mathbf{v}_C^l$ aligns with the most sensitive direction for concept C which gives feature importance in layer l. The dot product essentially measures similarity between feature importance for prediction $h_{l,k}(f_l(\mathbf{x}))$ and feature importance for concept C in layer l.

The TCAV score is in turn defined as

$$\text{TCAV}_{\text{Q},C,k,l} = \frac{|\{\mathbf{x} \in \mathbf{x}^k : S_{C,k,l}(\mathbf{x}) > 0\}|}{|\mathbf{x}^k|}, \tag{2.8}$$

which computes the proportion of inputs of class k that align with concept C among all the inputs of class k, $\mathbf{x}^k$, as concept attribution. A variety of experiments demonstrated that TCAV is a helpful tool for generating human-friendly explanations.

Partial Dependence Plot (PDP) [58] visualizes the partial dependence of the model's output on a subset of the input features. The partial dependence function of a machine learning model $\hat{f}$ for the selected features $\mathbf{x}_S$ is defined as

$$\hat{f}_S(\mathbf{x}_S) = \mathbb{E}_{\mathbf{X}_C}[\hat{f}(\mathbf{x}_S, \mathbf{X}_C)] = \int \hat{f}(\mathbf{x}_S, \mathbf{X}_C) d\mathbb{P}(\mathbf{X}_C), \tag{2.9}$$

where $\mathbf{X}_C$ are the rest features used in the model $\hat{f}$. A partial dependence plot helps humans to understand the relationship between the target and a feature by marginalizing the model's output over the distribution of the features in set C [151].

Greenwell et al. [76] proposed a feature importance score based on partial dependence. In particular, the importance of numerical features is defined as

$$I(\mathbf{x}_S) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(\hat{f}_S(\mathbf{x}_S^{(k)}) - \frac{1}{K}\sum_{k=1}^{K}\hat{f}_S(\mathbf{x}_S^{(k)}))^2}, \tag{2.10}$$

where $\mathbf{x}_S^{(k)}$ are the K unique values of $\mathbf{X}_S$. The importance of categorical features is defined as

$$I(\mathbf{x}_S) = (\max_k(\hat{f}_S(\mathbf{x}_S^{(k)})) - \min_k(\hat{f}_S(\mathbf{x}_S^{(k)})))/4. \tag{2.11}$$

This feature importance score should be interpreted with caution because it captures only the main effect of the feature and ignores possible feature interactions [151].

Accumulated Local Effects (ALE) Plot [7] visualizes the *average* influence of a feature on the model output. ALE plots average the changes in the predictions:

$$\hat{f}_{S,ALE}(\mathbf{x}_S) = \int_{\mathbf{z}_{0,S}}^{\mathbf{x}_S} \mathbb{E}[\hat{f}^S(\mathbf{X}_S, \mathbf{X}_C)|\mathbf{X}_S = \mathbf{z}_S]d\mathbf{z}_S - \text{constant}, \tag{2.12}$$

where $\hat{f}^S(\mathbf{X}_S, \mathbf{X}_C) = \dfrac{\partial \hat{f}(\mathbf{X}_S, \mathbf{X}_C)}{\partial \mathbf{X}_S}$ represents the local effect of $\mathbf{X}_S$ on $\hat{f}(\cdot)$ [7].

In contrast to PD plots, ALE plots have the following advantages: They are unbiased and faster to compute [151]. However, ALE effects may differ from the coefficients specified in a linear regression model when features interact and are correlated [151].

2.2. Learning from Explanations

In supervised learning, models are trained using a set of input and label pairs. However, labels alone may not contain sufficient information in some cases, which can cause the task to be ambiguous and the problem to be underdetermined. That means, there are infinitely many solutions to the task. Sometimes, it is relatively easy to identify an underdetermined task. For example, in a dataset that labels white male faces as one category and dark female faces as another, ambiguity arises and one can identify this ambiguity by inspecting the data. Nevertheless, in a high-dimensional domain, there may be a wide variety of fundamentally different solutions to a task that can't be easily identified, and models

like deep neural networks are well known to be prone to learn a shortcut solution in the data. Many solutions may lead the model to a low training error, but not all these solutions generalize equally well outside of the training set. In fact, the shortcut solutions oftentimes turn out to be spurious correlations that generalize poorly [117, 203, 212].

To counteract the aforementioned problem, many studies have been done to learn from explanations. Learning from explanations means, that not only do models give machine explanations to users, but users also help the model by giving human explanations as additional supervision for training the model. Learning models that align with proper user explanations can result in better generalization [198] and may substantially reduce the sample complexity [22].

Allowing models to automatically learn from explanations can be implemented as using explanations to regularize the model, which in turn can be implemented as penalizing unwanted explanations. This is especially useful for deep neural networks which maintain tremendous predictive capacity and often need more inductive bias than the training data can provide. To penalize unwanted explanations, it is common to augment the classification loss with an additional term that accommodates explanatory feedback. Then the parameters can be learnt using standard gradient-based techniques. Given the ground-truth target $\mathbf{y}$ and explanation $\mathbf{e}$, the predicted target $\hat{\mathbf{y}}$ and machine explanation $\hat{\mathbf{e}}$, the loss generally takes the form

$$L(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{e}, \hat{\mathbf{e}}, \hat{\boldsymbol{\theta}}) = \underbrace{L_{\text{Task}}(\mathbf{y}, \hat{\mathbf{y}})}_{\text{predictive loss}} + \lambda_1 \underbrace{L_{\text{Expl}}(\mathbf{e}, \hat{\mathbf{e}})}_{\text{explanatory loss}} + \lambda_2 \underbrace{L_{\text{Reg}}(\hat{\boldsymbol{\theta}})}_{\text{regularization}}, \tag{2.13}$$

where L_{Task} is the task-specific loss to ensure predictive accuracy. L_{Expl} is the explanatory loss to penalize machine explanations that do not align with human explanations. Optionally, an additional regularization L_{Reg} may also be used. λ_1 and λ_2 control the balance between each loss term.

In the remaining of this section, a variety of approaches for implementing the explanatory loss $L_{\text{Expl}}(\mathbf{e}, \hat{\mathbf{e}})$ will be presented.

2.2.1. Passive Learning

In passive learning, both input and supervision are available prior to the training phase, and no interaction takes place during training. This subsection presents a set of popular approaches for learning from explanations in passive settings.

Right for Right Reasons (RRR) [198] is proposed by Ross et al. to learn models with

right reasons using binary supervision on input gradients. The loss function is as follows.

$$
\begin{aligned}
L_{\text{RRR}} = & \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K} -y_{nk}\log(\hat{y}_{nk})}_{\text{predictive loss}} \\
& + \underbrace{\lambda_1 \sum_{n=1}^{N}\sum_{d=1}^{D}(A_{nd}\frac{\partial}{\partial x_{nd}}\sum_{k=1}^{K}\log(\hat{y}_{nk}))^2}_{\text{explanatory loss}} \\
& + \lambda_2 \underbrace{\sum_{i}\theta_i^2}_{\text{regularization}} \quad ,
\end{aligned}
\tag{2.14}
$$

where $\mathbf{A} \in \{0,1\}^{N\times D}$ is a binary annotation matrix that encodes user feedback by using 1 to indicate irrelevant features and 0 to indicate absence of feedback. That means, large input gradients $\nabla_{\mathbf{x}}\hat{\mathbf{y}}$ should be penalized for the features annotated as 1 by users. By optimizing the parameters to reduce the predictive loss, user feedback can be propagated back to the model, which constrains the model to align with user feedback. The explanatory loss augments the predictive loss that uses the popular cross-entropy to ensure predictive accuracy and the L2 regularization on the parameters $\boldsymbol{\theta}$. λ_1 and λ_2 control the strength of the explanatory loss and the regularization. According to empirical evidence, λ_1 and λ_2 should be chosen such that the predictive loss and the explanatory loss are on the same magnitude [198].

Empirical studies demonstrate that RRR can learn generalizable decision rules even when ambiguities are present. Moreover, a range of qualitatively different solutions can be discovered with find-another-explanation method when user feedback is not present.

Human Importance-aware Network Tuning (HINT) [207] leverages focused user feedback to improve visual grounding by optimizing the alignment between human attention maps and gradient-based network importances.

Human importance map $\mathbf{A}^d \in \mathbb{R}^{h\times w}$ indicates high support for an output d at location (i,j) with a high value $A^d_{i,j}$. The normalized importance inside and outside a proposal region r for decision d are given respectively by

$$
E_i^d(r) = \frac{1}{a_r}\sum_{(i,j)\in r} A_{ij}^d \quad \text{and} \quad E_o^d(r) = \frac{1}{hw - a_r}\sum_{(i,j)\notin r} A_{ij}^d \tag{2.15}
$$

for a given proposal region r with area a_r. The overall importance score for proposal k for

decision d is in turn defined as

$$s_k^d = \frac{E_i^d(k)}{E_i^d(k) + E_o^d(k)}. \tag{2.16}$$

Network importance is defined as the importance of spatial regions of the input for making a specific prediction. This is computed by extending Grad-CAM [206, 205] to compute importance over salient image regions and their features. In particular, the importance of a proposal r for the ground-truth decision is computed based on the gradient of one-hot encoded score for the ground-truth output o_{gt} w.r.t. the embedding $\mathbf{P}$:

$$\alpha_{gt}^r = \sum_{i=1}^{|\mathbf{P}|} \frac{\partial o_{gt}}{\partial P_i}. \tag{2.17}$$

In order to enforce human importance and network importance to align with each other, a ranking loss is used to search all possible pairs of proposals and penalize misranked pairs, i.e. the pairs where the ranking based on network importance does not match the ranking based on human importance. The loss is given by

$$L = \sum_{(r',r)\in\mathcal{S}} |\alpha_-^{r'} - \alpha_+^r|, \tag{2.18}$$

based on the set of all the misranked pairs r and r'. $+$ indicates that the proposal r is more important than r' according to human importance, and $-$ indicates otherwise.

Finally, the ranking loss in Equation 2.18 is combined with the original task loss to yield a complete objective for training:

$$L_{\text{HINT}} = \sum_{(r',r)\in\mathcal{S}} |\alpha_-^{r'} - \alpha_+^r| + \lambda L_{\text{Task}}, \tag{2.19}$$

where L_{Task} is the cross-entropy loss for Visual Question Answering (VQA) and the negative log-likelihood for image captioning. λ controls the balance between explanatory loss and predictive loss.

Empirical studies demonstrate that HINT is effective at improving visual grounding in vision and language tasks, which in turn improves the generalization ability out of the training distribution and boosts trust from the users.

Contextual Decomposition Explanation Penalization (CDEP) [190] is proposed by Rieger et al. to improve deep learning models by incorporating domain knowledge using a particular explanation technique — Contextual Decomposition (CD). CDEP works in a

similar spirit as RRR and HINT — directly penalizing importances that are not relevant according to user feedback. Yet the use of contextual decomposition for explanations allows CDEP to account not only for feature importances but also for importances of feature interactions [224, 157].

Originally conceived for Long Short-Term Memory (LSTM) [157], contextual decomposition was later adapted to arbitrary DNNs [224]. This algorithm decomposes logits $g(\mathbf{x})$ into a sum of two parts. One part captures the importance score of the feature group $\{\mathbf{x}_j\}_{j\in S}$, denoted as $\beta(\mathbf{x})$. The other part is the contributions to $g(\mathbf{x})$ not included in $\beta(\mathbf{x})$ — $\gamma(\mathbf{x})$ [190]. CD score is then given by applying decompositions $g_i^{CD}(\mathbf{x})$ iteratively for each layer $g_i(\mathbf{x})$ in a given DNN:

$$\begin{aligned} g^{CD}(x) &= g_L^{CD}(g_{L-1}^{CD}(\dots(g_2^{CD}(g_1^{CD}(\mathbf{x}))))) \\ &= (\beta(\mathbf{x}), \gamma(\mathbf{x})) \\ &= g(\mathbf{x}). \end{aligned} \tag{2.20}$$

Applying SoftMax on $g^{CD}(\mathbf{x})$ to normalize the output yields a vector $\beta(\mathbf{x}_j)$ for any subset of features $S \subseteq \{1, \dots, D\}$. CDEP loss is in turn given by

$$L_{\text{CDEP}} = \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K} -y_{nk}\log\hat{y}_{nk}}_{\text{predictive loss}} + \lambda \underbrace{\sum_{n=1}^{N}\sum_{s\in S}\|\beta(\mathbf{x}_{ns}) - \mathbf{e}_{ns}\|_1}_{\text{explanatory loss}}, \tag{2.21}$$

where $\mathbf{e}_{ns}$ is the ground-truth explanation for a subset of features S.

The main advantage of CDEP compared to RRR and HINT is that it allows for penalizing complex features and feature interactions. Empirical studies demonstrate CDEP learns solutions that align better with prior knowledge, which in turn improves predictive accuracy. The efficacy of CDEP is also shown in combating bias on several datasets.

2.2.2. Explanatory Interactive Machine Learning (XIL)

The approaches in the previous subsection all view explanations as a simple heatmap. However, they do not consider the fact that explanation is a conversation and conveying explanation to someone is a social process [87, 237, 147]. As such, the importance of interaction for explanation deserves research attention.

As a step towards extending explanations in an interactive fashion, Teso et al. propose the framework of Explanatory Interactive Machine Learning (XIL), which establishes the link between interacting, explaining and building trust [237]. In particular, they posit that both interaction and explanations are essential for building trust in machine learners.

In conventional interactive learning approaches, such as active [208], coactive [218], and active imitation learning [101], interaction usually takes place only based on labels. That is, the model can present decisions or query labels from humans to facilitate its own learning process. As a step forward, explanatory interactive learning does not only communicate with humans by labels, but also by explanations.

According to Teso et al. [237], interactions in XIL can be categorized into three scenarios:

1. Right for the right reasons: The prediction and the explanation are both correct, so no interaction from humans is required.
2. Wrong for the wrong reasons: The prediction is wrong, therefore the explanation must be wrong as well. In this case, humans are supposed to provide feedback.
3. Right for the wrong reasons: The prediction is correct but the explanation is wrong. In this case, humans are required to give their explanations on this prediction.

These three scenarios are summarized into the framework *CAIPI*, see Algorithm 1 for the high-level algorithm. It assumes black box access to the learner model $\hat{f}(\cdot)$, its explanation model `Explain(·)`, as well as the strategy to transform human explanations to counterexamples `ToCounterExamples(·)`. CAIPI includes a simulated human annotator in the training loop that repeatedly performs the following operations in each iteration until the model is good enough or the iteration budget is used up: The learner selects an instance $\mathbf{x}$ to query the user by giving $\mathbf{x}$, its own prediction $\hat{\mathbf{y}}$, and its own explanations $\hat{\mathbf{e}}$. As feedback, the user gives the label correction $\mathbf{y}$ and the explanation correction $\mathcal{C}$. $\mathcal{C}$ is further transformed into counterexamples $\{(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)\}_{i=1}^{c}$ to update the labeled training data $\mathcal{L}$, which in turn is used to update the current model.

In the original work, Teso et al. [237] instantiate the explanation model using LIME [188] and automatically generate a set of augmented data, known as counterexamples, to implicitly encode user explanations that can be consumed by the learner model. Specifically, counterexamples are generated based on the query instance $\mathbf{x}$ by either randomizing its j-th component, changing its j-th component to an alternative value or the value of the j-th component observed in other training samples of the same class, for every $j \in \mathcal{C}$ [237]. The counterexamples are labeled identically to the prediction $\hat{\mathbf{y}}$. Intuitively, counterexamples collectively express that predictions are invariant to perturbations of the j-th component, which alludes to the fact that j-th component is of nearly no importance to the prediction.

Empirical evidence confirmed that explanatory interactive learning can indeed boost performance of the learner and promote user's trust into the model.

Data augmentation provides great generality for propagating explanatory feedback from the user to the model. However, it is worth noting that data augmentation is easiest for

Algorithm 1: CAIPI($\mathcal{L}, \mathcal{U}, T$) [237]

Input : A set of labeled examples $\mathcal{L}$, a set of unlabeled instances $\mathcal{U}$, and iteration budge T.

Output : A function $\hat{f}$.

$\hat{f} \leftarrow$ `Fit(`$\mathcal{L}$`)`
repeat
  $\mathbf{x} \leftarrow$ `SelectQuery(`$\hat{f}, \mathcal{U}$`)`
  $\hat{\mathbf{y}} \leftarrow \hat{f}(\mathbf{x})$
  $\hat{\mathbf{e}} \leftarrow$ `Explain(`$\hat{f}, \mathbf{x}, \hat{\mathbf{y}}$`)`
  Present $\mathbf{x}, \hat{\mathbf{y}}$ and $\hat{\mathbf{e}}$ to the user
  Obtain $\mathbf{y}$ and explanation correction $\mathcal{C}$
  $\{(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)\}_{i=1}^{c} \leftarrow$ `ToCounterExamples(`$\mathcal{C}$`)`
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{x}, \mathbf{y})\} \cap \{(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)\}_{i=1}^{c}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus (\{\mathbf{x}\} \cup \{\bar{\mathbf{x}}_i\}_{i=1}^{c})$
  $\hat{f} \leftarrow$`Fit(`$\mathcal{L}$`)`
until *budget T is exhausted or $\hat{f}$ is good enough*;
return $\hat{f}$

classification tasks and difficult for a density estimation task unless the density estimation problem is solved already [69]. Apart from data augmentation, CAIPI is also compatible with other more restrictive approaches, for instance, using contrastive examples [254] or penalizing gradient-based explanations for differentiable models [198, 203, 212]. In Chapter 8 another variant of explanatory feedback will be discussed in detail.

On the whole, CAIPI is instantiated and studied in [237] based on feature-attribution explanations. That means, both the learner model and the user explain by attributing importance to each feature. However, this form of explanation has limitations facing explanations arising from a higher level of abstraction such as latent factors. In Chapter 9 CAIPI will be further extended to scenarios based on explanations of latent and abstract features such as shape and color.

Neuro-Symbolic Explanatory Interactive Machine Learning (NeSy XIL) [228] is a neuro-symbolic form of learning from explanations. In particular, this approach computes machine explanations and interacts with user feedback at the concept level to correct Clever Hans behavior that cannot be expressed by a saliency map on the features.

NeSy XIL consists of two modules: The concept embedding module $h(\mathbf{x}) = \hat{\mathbf{z}}$ transforms the input to a symbolic representation, and the reasoning module $g(\hat{\mathbf{z}}) = \hat{\mathbf{y}}$ yields the final

predictions given the symbolic representation.

The concept learner is instantiated by Slot Attention [132] for parsing the latent representation into a set of slots, i.e. task-dependent output vectors, to yield an object-based representation of a input image. The attention maps associate each slot with the original input features and form visual explanations on the level of objects. The dot-product attention for a sample $\mathbf{x}$ is then given by

$$\mathbf{B} \coloneqq \sigma(\frac{1}{\sqrt{D'}} k(\mathbf{F}) \cdot q(\mathbf{S})^T) \in \mathbb{R}^{P \times K}, \tag{2.22}$$

where σ denotes the softmax function, $k(\mathbf{F}) \in \mathbb{R}^{P \times D'}$ denotes a linear transformation of the feature maps $\mathbf{F}$ for an image $\mathbf{x}$, $q(\mathbf{S})^T \in \mathbb{R}^{K \times D'}$ denotes a linear transformation of the slot encoding $\mathbf{S}$. P and K are respectively the dimensions of the feature map and the slots. D' is the dimension which the key and query functions map to.

The symbolic explanation $\hat{\mathbf{e}}^h$ is in turn defined based on the attention map $\mathbf{B}$ as

$$\hat{\mathbf{e}}^h \coloneqq \sum_{k=1}^{K} \begin{cases} \mathbf{B}_k & \text{if } \max(\hat{\mathbf{e}}_k^g) \geq t \\ \mathbf{0} \in \mathbb{R}^P & \text{otherwise} \end{cases} \tag{2.23}$$

for a threshold t as hyperparameter.

Given an explanation function $E(m(\cdot), \mathbf{o}, \mathbf{s})$ for a specific module $m(\cdot)$, the module's input $\mathbf{s}$ and the module's output $\mathbf{o}$, $\hat{\mathbf{e}}^g \coloneqq E^g(g(\cdot), \hat{\mathbf{y}}, \mathbf{z})$ represents the explanation of the reasoning module for the predicted output $\hat{\mathbf{y}}$, and $\hat{\mathbf{e}}^h \coloneqq E^h(h(\cdot), \hat{\mathbf{e}}^g, \mathbf{x})$ represents the explanation of the concept module given the explanation of the reasoning module $\hat{\mathbf{e}}^g$, whereby $\hat{\mathbf{e}}^h$ gives a visualization of a learned concept explanation.

The reasoning module is instantiated with the Set Transformer [121], which yields the concept embedding as the output. To explain the concept embedding, Integrated Gradients [234] are used to estimate the importance of each feature of a particular sample. Specifically, Integrated Gradient for feature j of the sample $\mathbf{z}$ is defined as

$$IG_j(\mathbf{z}) \coloneqq (z_j - \tilde{z}_j) \times \int_{\alpha=0}^{1} \frac{\partial g(\tilde{\mathbf{z}} + \alpha \times (\mathbf{z} - \tilde{\mathbf{z}}))}{\partial z_j} d\alpha, \tag{2.24}$$

whereby $\tilde{z}_j$ is the baseline input for feature j and $\tilde{\mathbf{z}} \in [0, 1]^{N \times D}$ is the input to the Set Transformer. Applying a threshold at zero and feeding $\tilde{\mathbf{z}} = 0$ as input yields the importance score

$$\hat{\mathbf{e}}^g \coloneqq \sum_{j=1}^{D} \min(IG_j(\hat{\mathbf{z}}), \mathbf{0}) \tag{2.25}$$

Finally, the loss function arrives at

$$L_{\text{NeSy-XIL}} = \underbrace{L_{\text{Task}}}_{\text{predictive loss}} + \underbrace{\lambda \sum_{n=1}^{N} r(\mathbf{A}_n^v, \hat{\mathbf{e}}_n^h) + (1-\lambda) \sum_{n=1}^{N} r(\mathbf{A}_n^s, \hat{\mathbf{e}}_n^g)}_{\text{explanatory loss}} \tag{2.26}$$

where $\mathbf{A}_n^v$ and $\mathbf{A}_n^s$ are respectively binary masks for the visual feedback and the semantic feedback. $r(\cdot, \cdot)$ is the regularization function, and λ controls the balance between the predictive loss and the explanatory loss.

Empirical evidence on several datasets confirmed that NeSy XIL allows for identifying Clever Hans behavior on the concept level and counteracting Clever Hans behavior using user feedback.

2.3. Caveats of Explainable AI

Amid the rapid development of explainable AI that aims to explain black-box models with post-hoc methods, some researchers raise concerns about this group of methodology [199, 107]. One of the major concerns is that post-hoc explanations may not be faithful to the original model, meaning that the explanations are simply wrong [148].

As it is hard to assess viable explanations directly, one can use invariance under certain transformations to rule out the adequacy of certain explanations [1, 107]. Running examinations on a number of explanation methods to date, numerous of them are found to deviate substantially from the original model. On one hand, explanation methods may be undesirably sensitive to certain perturbations. For instance, Kindermans et al. [107] introduce input invariance as a prerequisite for reliable attribution, which requires the saliency method to reflect the sensitivity of the original model to input transformations. In particular, for models that are invariant to a constant shift in input by construction, their explanations are supposed to be invariant to a constant shift in input as well. Unfortunately, some saliency methods, such as Deep-Taylor Decomposition [152] and Integrated Gradients [234], fail to satisfy this property [107]. On the other hand, explanation methods may lack sensitivity to the model and the data generating process [1]. Adebayo et al. [1] observe that some saliency methods, such as Guided Backpropagation [226] and Guided GradCAM [206], are surprisingly invariant under model randomization and label randomization, therefore they are incapable of capturing any pattern in the decision-making process of the original model.

More worrisome than apparent infidelity, explanations are subject to purposeful and visually imperceptible manipulations as well [47]. Dombrowski et al. [47] propose to

manipulate input by optimizing the loss function

$$L = \|\hat{\mathbf{e}}_{adv} - \mathbf{e}_{tar}\|^2 + \gamma\|\hat{f}(\mathbf{x}_{adv}) - \hat{f}(\mathbf{x})\|^2 \tag{2.27}$$

w.r.t. $\mathbf{x}_{adv}$ such that the manipulated image $\mathbf{x}_{adv} = \mathbf{x} + \delta\mathbf{x}$ yields a similar prediction as $\mathbf{x}$, i.e. $\hat{f}(\mathbf{x}_{adv}) \approx \hat{f}(\mathbf{x})$, while its explanations $\hat{\mathbf{e}}_{adv}$ close to the target $\mathbf{e}_{tar}$. Empirical evidence on six different explanation methods and four network architectures show that this method is very effective at manipulating images for an arbitrary explanation map. Although theoretical considerations have been given to this phenomenon and methods have been proposed to improve the robustness of gradient-based explanation methods, rigorous proof still lacks for neural networks that are deeper than one layer [47]. Consequently, the robustness of the explanation methods is still not guaranteed and unfit for high-stakes prediction applications [199].

Due to the aforementioned problems, it is important to properly evaluate the quality of explanations before employing them in real applications. For evaluating predictive quality there are a wide variety of principled metrics such as Mean Squared Error (MSE), precision and recall, F1 score etc. However, there is little consensus on the definition of what constitutes a viable explanation and how to evaluate an explanation for benchmarking [48].

Doshi-Velez et al. [48] propose a taxonomy of approaches for evaluating the quality of an explanation. *Application-grounded evaluation* requires experimenting with real humans on real tasks, specifically in the context of its end task. This type of evaluation compares machine explanations with human explanations for task-specific goals, which is most costly to conduct and requires human experts for the explanations [48]. *Human-grounded evaluation* requires experimenting with real humans while using simplified tasks that capture the essence of the target task. For simplified tasks, explanations of laypeople instead of domain experts suffice as the baseline. This type of evaluation does not consider the end goal of a particular task but a general comparison between machine explanations and human explanations [48]. *Functionally-grounded evaluation* further reduces the cost of the previous two types of evaluation and experiments with no human and only proxy tasks [48].

Functionally-grounded evaluation is most commonly used in the research of explanation methods. The core problem lies in the design strategy of the proxy task [57]. As an example, one may measure the improvement of the predictive power of interpretable models, such as decision trees [57], based on explanations. Another widely used proxy task is to remove the important features according to the explanations and measure the drop in predictive performance [201]. However, this approach violates the key assumption in machine learning that the training set and the test set should be identically and

independently distributed, i.e. i.i.d. assumption. The consequence is that it is unclear whether to attribute the performance drop to absent important features or the distribution shift [38, 56, 91]. Therefore, Hooker et al. [91] propose RemOve And Retrain (ROAR) to retrain the model after removing the supposedly informative features and measure the performance drop of the retrained model to determine the efficacy of the explanations. Extensive empirical evaluation implies that Gradients [10], Integrated Gradients [234] and Guided BackProp [226] perform no better than random assignment of importance [91]. The results partially correspond to the findings made by Kindermans et al. [107] that Integrated Gradients fail to mirror the sensitivity of the original model to certain input transformations. The evaluation of Guided BackProp reinforces the assessment of Adebayo et al. [1] that Guided BackProp fails to respond to the change of both the model and the input. Nevertheless, it is worth noting that ROAR may unfairly underestimate the explanation quality. Think of a classification task being solved on a dataset with a confounding factor, assume the target model learnt the confounder for a seemingly good performance and the explanations correctly highlight the confounders as important features. Removing these features and retraining the model would very likely not lead to a performance drop because the model is capable of learning another solution in the data. Yet, the explanations still maintain high quality.

This concludes Chapter 2. We started by discussing explainable methods for post-hoc explanations, then we proceeded with the techniques using explanations as additional signals for learning the predictive model. In the end, we discussed open research questions in explainable AI.

Part II.

Deep Models

Deep models in machine learning have demonstrated impressive representational and predictive power in many complex problems. They are featured as graphs composed of multiple layers of computation units, hence the name "deep models". Due to the fact that humans cannot understand their underlying decision logic by inspecting the architecture and parameters, they are also called black box models. In this part, we put the spotlight on these deep black box models.

A dominant family of deep models are deep neural networks which have been used in a wide range of successful applications since 2012. They are computational models of multiple processing layers, designed to learn representations of data with multiple levels of abstraction [119]. By learning from a large quantity of labeled data $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, DNNs excel in approximating smooth functions $\mathbf{y} = f(\mathbf{x})$ that map an input to an output. This is known as the predictive or supervised learning approach.

Sometimes, we are more interested in finding patterns from data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ instead of predicting specific outcomes. This problem can be tackled by a descriptive or unsupervised learning approach, which in turn can be formalized as density estimation using probabilistic models. As opposed to DNNs, probabilistic models provide a rich language to encode complex representations of the world and in theory allow for a wider range of inference tasks that may require full modelling of the input distribution $P(\mathbf{X})$, for example, structured output prediction, density estimation under the data-generating distribution, missing value imputation and sampling [69]. Probabilistic models not only lay a cornerstone for advanced and automated reasoning but more importantly, they compute sound and consistent probabilities in a principled way, hence they allow for making uncertain statements or reasoning with uncertainty [69]. Among probabilistic models, Probabilistic Circuits (PCs) are becoming increasingly popular because they maintain a good balance on both tractability and expressivity. Therefore, we discuss PCs in this part for probabilistic modelling. Similar to DNNs, PCs are essentially computation graphs composed of multiple processing layers.

This part is organized as follows. We start with reviewing deep neural networks in Chapter 3, where we cover model representation, learning and some common regularization techniques. Then, we review Sum-Product Networks (SPNs) in Chapter 4 from three aspects — representation, inference, and learning, which are critical components in constructing an intelligent system [112]. Additionally, a deep learning approach for SPNs is introduced, which is followed by different perspectives on SPNs. Finally, we present our work on Conditional Sum-Product Networks (CSPNs) as a conditional counterpart for SPNs in Chapter 5. CSPNs are designed for conditional density estimation. Apart from pure conditional density estimation, CSPNs also have great use as modular building blocks in joint probability distributions to impose structure on the high level for better global interpretability.

3. Deep Neural Networks

In this chapter, we briefly review a popular type of deep models — deep neural networks. We start with the layered representation of classical DNNs in Section 3.1, which is also the computation graph for inference routines. Then we review how deep neural networks are learnt from data in Section 3.2, followed by common regularization techniques used to improve learning in Section 3.3. Finally, we present a neural network-based generative model in Section 3.4 — variational autoencoders — which will be employed later in Chapter 6 and 9. Note that deep neural networks are a very broad and fast-growing field, and we only bring up the essential building blocks here that will be relevant for this thesis.

3.1. Model Representation

Deep neural networks are computation graphs composed of multiple layers of different simple functions $f^{(i)}(\cdot)$ representing a more complex function

$$f(\mathbf{x}) = f^{(n)}(f^{(n-1)}(\cdots(f^{(1)}(\mathbf{x}))),$$

where $f^{(1)}(\mathbf{x})$ is the input layer, $f^{(n)}(\mathbf{x})$ is the output layer, and the rest are hidden layers. They are also known as *deep feedforward networks* or *feedforward neural networks*. They are called "deep" models due to the chained structure, which can be graphically represented as directed acyclic graphs (DAG) (see Figure 3.1). Within each layer, there are multiple processing units, which are called neurons because they play an analogous role as biological neurons. In particular, they take inputs from the neurons of the previous layer and compute activation values. The choice of the functions $f^{(i)}(\cdot)$ used to compute these representations is also loosely guided by neuroscientific observations about the functions that biological neurons compute [69]. The number of hidden layers and the number of neurons within each layer is a design choice.

To ensure that a neural network can represent more than linear functions, each layer should compute a nonlinear function $f^{(i)}(\cdot)$. This is because the composition of linear functions still yields linear functions. Most neural networks introduce nonlinearity in each

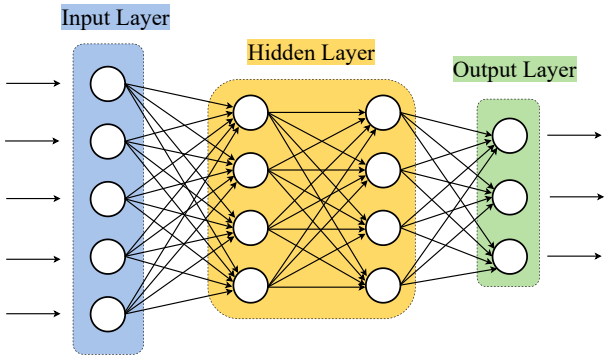


Figure 3.1.: The architecture of feedforward networks.

hidden layer using an affine transformation controlled by learnable parameters, followed by a fixed nonlinear function called an activation function [69]. That is,

$$\mathbf{h} = g(\mathbf{W}^\top \mathbf{x} + \mathbf{b}),$$

where $\mathbf{W}$ is the weight vector of a linear transformation and $\mathbf{b}$ is the bias vector. The activation function $g(\cdot)$ applied element-wise to the output of a linear transformation is the source of nonlinearity. A common activation function for modern DNNs is Rectified Linear Unit (ReLU) [99, 159, 68], i.e. $g(z) = \max\{0, z\}$.

The universal approximation theorem [92, 37] states that a feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ with an arbitrarily small amount of error, given enough hidden units. Following the universal approximation theorem, a feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly [69]. Montúfar et al. [154] investigated the advantage of depth for neural networks with piecewise linear activations and showed that the layer-wise composition of the functions re-uses low-level computations exponentially often as the number of layers increases.

Feedforward networks are the most general-purpose neural networks. More specialized neural networks emerged and keep emerging to solve particular tasks. The most established example is probably *Convolutional Neural Networks* (CNNs) [118] that are dominant for vision tasks. They are simply neural networks that use convolution, a specialized kind of linear operation, in place of general matrix multiplication in at least one of their layers [69]. For a two-dimensional image I and a two-dimensional kernel K, convolution in

mathematics is

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n).$$

Convolution commonly used in CNNs is slightly different:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n).$$

Convolution makes use of three key concepts that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations [69]. Interactions are sparse because only a small area of pixels that correspond to the kernel are processed at each operation. A kernel with the same parameters is reused across multiple functions, hence the name “parameter sharing". Convolution is equivariant to translation. That means, translating input prior to a convolution is the same as translating the output of the convolution [69].

There are many different architectures for CNNs, such as AlexNet [113], GoogLeNet [235], VGGNet [222] etc. Another specialized family of neural networks are Recurrent Neural Networks for sequence data, for example Long Short-Term Memory [88], Gated Recurrent Unit [32], and Neural Turing Machines [75]. These are, however, less relevant for this thesis and we do not expand on them here.

3.2. Learning

Once the number of hidden layers and the number of neurons within each layer are chosen, the structure is fixed. Learning the network amounts to finding a set of parameters $\hat{\boldsymbol{\theta}}$ to approximate the smooth function $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$, given a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$. For evaluation, one needs a test set which is a held-out dataset with data previously unseen in the training set $\mathcal{D}$. The ultimate goal of learning is generalization, i.e. the ability to obtain high predictive accuracy on the test set. The basic assumption about the training and the test set is that they are samples of the same underlying distribution, and they are independently and identically distributed (i.i.d.).

A standard way to learn neural networks from a dataset is by descending a pre-defined cost function $L(\cdot)$ w.r.t. the model parameters $\boldsymbol{\theta}$, i.e.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\mathcal{D}}} L(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}),$$

where $\hat{p}_{\mathcal{D}}$ is the empirical distribution and

$$\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\mathcal{D}}} L(\hat{f}(\mathbf{x};\boldsymbol{\theta}),\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} L(\hat{f}(\mathbf{x}^{(i)};\boldsymbol{\theta}),\mathbf{y}^{(i)}).$$

A commonly used cost function is negative log-likelihood, which induces Maximum Likelihood Estimation (MLE) for the parameters. That means, the parameters of the neural networks are learnt by

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\mathcal{D}}} \log p_{\text{model}}(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}),$$

where $p_{\text{model}}(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ is the model distribution. Typically, this cost function is nonconvex for neural networks, therefore it is usually iteratively reduced using gradient descent [25]. Due to the nonconvexity, global convergence is not guaranteed. In practice, we simply optimize the loss to reach a low value and settle with that result.

Gradient descent is based on the idea that the gradient $f'(\mathbf{x})$ specifies how to scale a small change in input to obtain a corresponding change in output: $f(\mathbf{x}+\epsilon) \approx f(\mathbf{x}) + \epsilon f'(\mathbf{x})$, for a small ϵ [69]. This algorithm specifically depends on the fact that $f(\mathbf{x})$ can be reduced by a gradient step $\mathbf{x} - \epsilon\text{sign}(f'(\mathbf{x}))$. Stochastic gradient descent (SGD) may accelerate pure gradient descent by using an approximate gradient of randomly selected mini batches, which should be drawn i.i.d. from the data-generating distribution for an unbiased estimation. SGD is given in Algorithm 2.

A model is called *underfitting* if it is not able to obtain sufficiently low cost on the training set, which implies the model may not have enough capacity to represent the underlying patterns of the given data. On the contrary, a model is called *overfitting* if the cost is sufficiently low on the training set but high on the test set, which implies the model has high capacity but memorizes the training set. In other words, the model generalizes poorly. To counteract overfitting and reduce generalization error, a bunch of regularization strategies are designed. This will be further explained in the following section.

3.3. Regularization

When a model has enough capacity, it can learn more than one solution on a given dataset. These solutions may obtain similar training costs, but differ significantly in their generalization ability. Regularization is a way to give a learning algorithm a preference for one solution over another in its hypothesis space of solutions that the learning algorithm is able to choose from [69]. The preference for a specific solution can correspond to

Algorithm 2: Stochastic gradient descent (SGD) update [69]

Input : Learning rate schedule $\epsilon_1, \epsilon_2, \ldots$.
Output : Initial parameter $\boldsymbol{\theta}$.

$k \leftarrow 1$
while *stopping criterion not met* **do**
  Sample a minibatch of N examples from the training set $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$
  Compute gradient estimate: $\hat{\mathbf{g}} \leftarrow \frac{1}{N} \sum_{i=1}^{N} L(\hat{f}(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\mathbf{g}}$
  $k \leftarrow k + 1$
end

prior knowledge about the domain. As a side effect, regularization may induce a more interpretable model by learning a sensible solution.

Regularization can be realized by explicitly constraining the machine learning model or by adding a penalty term in the cost function as a soft constraint on the parameters [69]. Constrained optimization is an example of imposing explicit constraint. Another example is parameter tying, which is a key design behind CNNs. Sometimes it is easier to formulate our preference for certain solutions as soft constraints, i.e.,

$$\tilde{L}(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) = L(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) + \lambda \Omega(\boldsymbol{\theta}),$$

where λ controls the trade-off between the standard cost and the penalty $\Omega(\boldsymbol{\theta})$. The advantage of soft constraints is that the strength of the regularization effect can be flexibly adjusted, where larger λ induces a stronger regularization effect. In case of noisy or imprecise penalty, the optimization can be less influenced by tuning down λ.

A simple and commonly used penalty term is the L^2 parameter norm penalty

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2 = \sum_i \theta_i^2.$$

Thus the cost function with the L^2 penalty becomes

$$\tilde{L}(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) = L(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) + \lambda \|\boldsymbol{\theta}\|_2^2,$$

the gradient update of which is

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon\big(\nabla_{\boldsymbol{\theta}} L(\hat{f}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) + 2\lambda \boldsymbol{\theta}\big).$$

This gradient update has the additional effect of linearly reducing the magnitude of the parameters $\boldsymbol{\theta}$ by a factor of $2\epsilon\lambda$ [69]. Therefore, L^2 penalty is used when small size is desired for the model parameters.

Another technique to penalize the magnitude of the parameters is L^1 regularization, i.e.

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_i |\theta_i|.$$

Thus the cost function becomes

$$\tilde{L}(\hat{f}(\mathbf{x};\boldsymbol{\theta}), \mathbf{y}) = L(\hat{f}(\mathbf{x};\boldsymbol{\theta}), \mathbf{y}) + \lambda\|\boldsymbol{\theta}\|_1,$$

the gradient update of which is

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon\big(\nabla_{\boldsymbol{\theta}} L(\hat{f}(\mathbf{x};\boldsymbol{\theta}), \mathbf{y}) + \lambda \text{sign}(\boldsymbol{\theta})\big).$$

This gradient update also shrinks the parameters, but the shrinkage does not scale linearly with each parameter as L^2 regularization does [69]. More importantly, L^1 regularization does not only shrink the parameters, but also encourage the parameters to be zero. That means, L^1 regularization promotes sparsity whereas L^2 regularization does not.

In addition to the most basic and generic L^1 and L^2 penalty, designing more advanced and specific penalties for encoding fine-grained preferences over certain solutions is also an active research area. In Chapter 8 and 9, we will present some techniques for softly constraining DNNs based on explanatory feedback from the user.

Data augmentation provides another way for regularization. The most successful application of this technique has so far been seen for classification, especially object recognition. In particular, some transformations on the training images can often greatly improve generalization, for example, translating the training images a few pixels in each direction, rotating or scaling the images [69]. Data augmentation will be used in Chapter 9 as explanatory user feedback in combination with a penalty constraint to regularize latent representations for learning the right latent factors.

3.4. Variational Auto-Encoder

Variational Autoencoders (VAEs) [108] are generative models with latent variables. Assume observed variables $\mathbf{X}$, latent variables $\mathbf{Z}$ and parameters ψ of a generative model, a VAE represents a probability distribution over observed variables and latent variables $p_\psi(\mathbf{X}, \mathbf{Z})$.

Consider the posterior inference task $p_\psi(\mathbf{Z}|\mathbf{X})$, which can be answered given the generative model via the chain rule:

$$p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{Z}, \mathbf{X})}{p(\mathbf{X})}.$$

However, the marginal likelihood $p(\mathbf{X})$ is in general intractable because it requires marginalizing out the latent variables $\mathbf{Z}$, i.e. $p(\mathbf{X}) = \int p(\mathbf{Z}, \mathbf{X}) \mathrm{d}\mathbf{Z}$. Therefore, the posterior $p_\psi(\mathbf{Z}|\mathbf{X})$ is intractable as well. Variational Bayesian approach is one way to solve this problem by optimizing an approximation $q_\omega(\mathbf{Z}|\mathbf{X})$ of the intractable posterior. The Kullback-Leibler (KL) divergence of the approximate posterior from the true posterior is

$$\mathbb{D}_{\mathrm{KL}}[q_\omega(\mathbf{Z}|\mathbf{X})||p_\psi(\mathbf{Z}|\mathbf{X})] = \mathbb{E}_q[\log q_\omega(\mathbf{Z}|\mathbf{X})] - \mathbb{E}_q[\log p_\psi(\mathbf{Z}, \mathbf{X})] + \log p_\psi(\mathbf{X}). \tag{3.1}$$

The optimal approximate posterior is given by minimizing Equation 3.1. However, directly minimizing Equation 3.1 is intractable due to $\log p_\psi(\mathbf{X})$. Note $\mathbb{E}_q[\log p_\psi(\mathbf{Z}, \mathbf{X})] - \mathbb{E}_q[\log q_\omega(\mathbf{Z}|\mathbf{X})]$ as the evidence lower bound $\mathrm{ELBO}(\omega)$, which is the variational lower bound of $\log p_\psi(\mathbf{X})$, then

$$\log p_\psi(\mathbf{X}) = \mathrm{ELBO}(\omega) + \mathbb{D}_{\mathrm{KL}}[q_\omega(\mathbf{Z}|\mathbf{X})||p_\psi(\mathbf{Z}|\mathbf{X})]. \tag{3.2}$$

Since KL divergence is greater or equal than zero by definition, maximizing the ELBO is equivalent to minimizing KL divergence. Therefore, the ELBO is used as optimization objective in variational Bayesian inference to approximate posterior inference. $\mathrm{ELBO}(\omega)$ can also be written as

$$-\mathbb{D}_{\mathrm{KL}}[q_\omega(\mathbf{Z}|\mathbf{X})||p_\psi(\mathbf{Z})] + \mathbb{E}_q[\log p_\psi(\mathbf{X}|\mathbf{Z})].$$

Taking the perspective of auto-encoders, $p_\psi(\mathbf{X}|\mathbf{Z})$ is captured by the decoder and $\mathbb{E}_q[\log p_\psi(\mathbf{X}|\mathbf{Z})]$ can be interpreted as negative reconstruction error. $q_\omega(\mathbf{Z}|\mathbf{X})$ is captured by the encoder and $\mathbb{D}_{\mathrm{KL}}[q_\omega(\mathbf{Z}|\mathbf{X})||p_\psi(\mathbf{Z})]$ can be interpreted as a regularizer to enforce the approximate posterior $q_\omega(\mathbf{Z}|\mathbf{X})$ to be close to the prior distribution $p_\psi(\mathbf{Z})$. The prior distribution of the latent factors $p_\psi(\mathbf{Z})$ is often assumed to be isotropic Gaussian, i.e. $p_\psi(\mathbf{Z})$ is fully factorial, and hence the encoder network performs mean-field inference. The overall VAE model is a generative model $p_\psi(\mathbf{Z}, \mathbf{X})$ over the observed variables and the latent variables, which is given by $p_\psi(\mathbf{X}|\mathbf{Z})p_\psi(\mathbf{Z})$. The directed graphical model behind VAEs is given in Figure 3.2.

Learning VAEs involves approximating the intractable posterior $p_\psi(\mathbf{Z}|\mathbf{X})$ with $q_\omega(\mathbf{Z}|\mathbf{X})$. Kingma et al. [108] propose to rewrite an expectation w.r.t. $q_\omega(\mathbf{Z}|\mathbf{X})$ using the reparameterization trick such that the Monte Carlo estimate of the expectation is differentiable

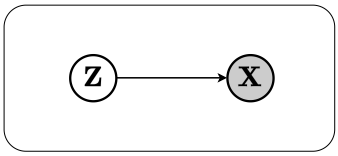


Figure 3.2.: The directed graphical model behind VAEs.

w.r.t. ω, which in turn yields a differentiable estimator of the variational lower bound. The essential idea of the reparameterization trick is to make sampling deterministically depend on parameters of the distribution. In other words, no stochasticity is involved in the distributional parameters. Take univariate Gaussian $z \sim \mathcal{N}(\mu, \sigma^2)$ as an example, a valid reparameterization is $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ is an auxiliary variable. As a consequence, $\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}[f(\mu + \sigma\epsilon)]$. With the reparameterization trick, a practical estimator for VAEs can be derived to optimize both ψ and ω jointly. In other words, the optimization can simultaneously fit the generative model and perform amortized posterior inference. The reparameterization trick will also be used for differentiable sampling from Conditional Sum-Product Networks (see Algorithm 7) in Chapter 5.

This concludes Chapter 3. We reviewed the basic components of DNNs and regularization, which is a common strategy to reduce generalization errors. In the end, we touched upon a deep generative model based on DNNs.

4. Sum-Product Networks

In this chapter, we review a deep probabilistic model — Sum-Product Networks (SPNs). We start by introducing network polynomial in Section 4.1, which lays the foundation for SPNs. Then, we give the structure representation of SPNs in Section 4.2, followed by their inference routines in Section 4.3 and learning algorithms in Section 4.4. Then we introduce a deep learning approach for SPNs in Section 4.5. In the end, we give some perspectives on SPNs in Section 4.6, which can be used for interpreting the network structure.

4.1. Network Polynomial

The notion of network polynomial was initially proposed by Darwiche [39] to parameterize probability tables. In particular, a pair of evidence indicators λ_{x_i} and $\lambda_{\bar{x}_i}$ are respectively introduced for each binary random variable X_i and its negation $\bar{X}_i$. Each indicator variable λ_{x_i} and its counterpart $\lambda_{\bar{x}_i}$ are respectively multiplied with the network parameters related to their corresponding variables X_i and $\bar{X}_i$.

Following the notation of Darwiche [39], we let $\mathbf{F} = \{X\} \cup \mathbf{U}$ be the family of variable X and let $\mathbf{f} = x\mathbf{u}$ be a corresponding observation. The conditional probability $p(x|\mathbf{u})$ is represented by $\theta_{\mathbf{f}}$ or $\theta_{x\mathbf{u}}$. Let $\mathbf{x} \sim \mathbf{f}$ denote that observations $\mathbf{x}$ and $\mathbf{f}$ are consistent. Then a Bayesian network $\mathcal{N}$ has a polynomial representation $\mathcal{F}$ as follows.

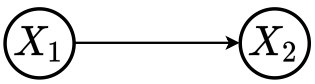
Definition 1 ([39]). *Let $\mathcal{N}$ be a Bayesian network with variables $\mathbf{X} = X_1, \ldots, X_n$ and families $\mathbf{F}_1, \ldots, \mathbf{F}_n$. Then,*

$$\mathcal{F}(\lambda_{x_i}, \theta_{\mathbf{f}_i}) = \sum_{\mathbf{x}} \prod_{\mathbf{f}_i \sim \mathbf{x}} \theta_{\mathbf{f}_i} \prod_{x_i \sim \mathbf{x}} \lambda_{x_i}$$

is called the canonical polynomial of network $\mathcal{N}$.

Take the Bayesian network $X_1 \rightarrow X_2$ as an example [39], its probability tables are given in Figure 4.1. Its canonical polynomial is:

$$\mathcal{F} = \theta_{x_1}\theta_{x_1x_2}\lambda_{x_1}\lambda_{x_2} + \theta_{x_1}\theta_{x_1\bar{x}_2}\lambda_{x_1}\lambda_{\bar{x}_2} + \theta_{\bar{x}_1}\theta_{\bar{x}_1x_2}\lambda_{\bar{x}_1}\lambda_{x_2} + \theta_{\bar{x}_1}\theta_{\bar{x}_1\bar{x}_2}\lambda_{\bar{x}_1}\lambda_{\bar{x}_2}. \tag{4.1}$$



X_1	$p(X_1)$
0	$\theta_{\bar{x}_1}\ \lambda_{\bar{x}_1}$
1	$\theta_{x_1}\ \lambda_{x_1}$

X_1	X_2	$p(X_2\|X_1)$
0	0	$\theta_{\bar{x}_1\bar{x}_2}\ \lambda_{\bar{x}_2}$
0	1	$\theta_{\bar{x}_1 x_2}\ \lambda_{x_2}$
1	0	$\theta_{x_1\bar{x}_2}\ \lambda_{\bar{x}_2}$
1	1	$\theta_{x_1 x_2}\ \lambda_{x_2}$

Figure 4.1.: A simple Bayesian Network and its probability table, adapted from [39]. λ are evidence indicators.

This polynomial is depicted graphically in Figure 4.2. Note that the canonical polynomial $\mathcal{F}$ of a Bayesian network is a multilinear function over its network variables $\mathbf{X}$ [39]. Besides, the size of $\mathcal{F}$ is always exponential in the number of $\mathbf{X}$ [39], however, Darwiche also showed how to factor $\mathcal{F}$ using the technique of variable elimination. The factored counterpart for the polynomial in Equation 4.1 is

$$\mathcal{F} = \theta_{x_1}\lambda_{x_1}(\theta_{x_1 x_2}\lambda_{x_2} + \theta_{x_1\bar{x}_2}\lambda_{\bar{x}_2}) + \theta_{\bar{x}_1}\lambda_{\bar{x}_1}(\theta_{\bar{x}_1 x_2}\lambda_{x_2} + \theta_{\bar{x}_1\bar{x}_2}\lambda_{\bar{x}_2}). \tag{4.2}$$

This polynomial is depicted graphically in Figure 4.3. We refer the reader to the original paper for more details on the technique of compiling a *factored* polynomial representation of a Bayesian network.

To do probabilistic inference using the network polynomials, the unnormalized probability of an evidence $\mathbf{x}$ is induced by setting λ_{x_i} to 1 and $\lambda_{\bar{x}_i}$ to 0 if x_i is consistent with the evidence, i.e. $x_i \sim \mathbf{x}$. To marginalize an variable X_i, we set both λ_{x_i} and $\lambda_{\bar{x}_i}$ to 1 and evaluate its network polynomial. The indicator variables in effect select which network parameters to add together for inference [39].

The notion of network polynomial lays a cornerstone of sum-product networks, which will be introduced in the following section.

4.2. From Network Polynomial to SPNs

The network polynomial introduced in the previous section can be compiled from Probabilistic Graphical Models (PGMs) to represent unnormalized probabilistic inference. The partition function is the value of the network polynomial when all evidence indicators are

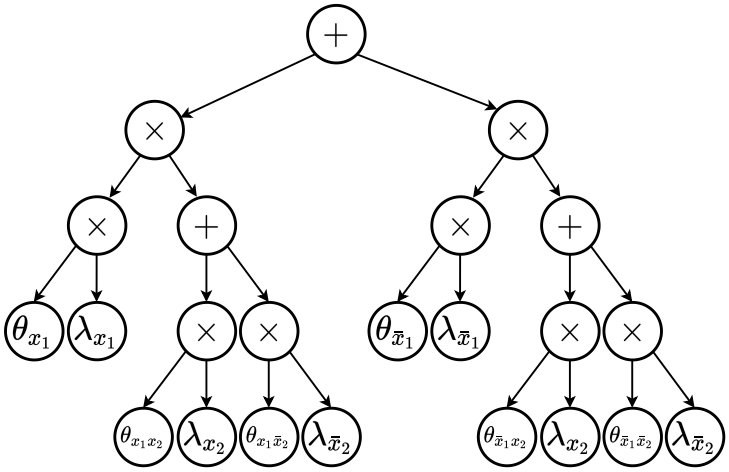


Figure 4.2.: The canonical polynomial for the BN in Figure 4.1.

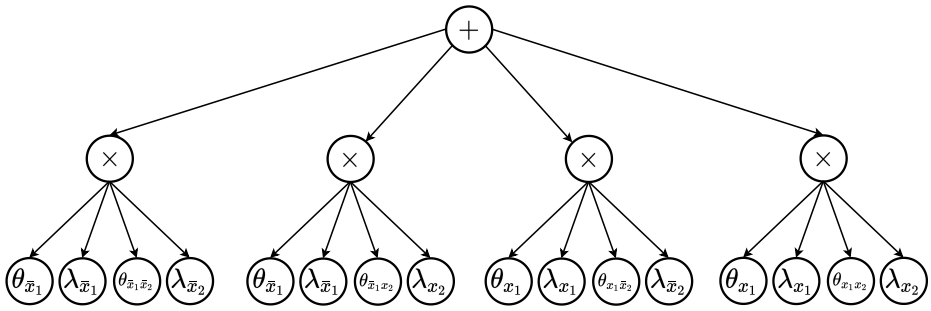


Figure 4.3.: The factored polynomial for the BN in Figure 4.1, given by [39].

set to 1 [174]. However, the complexity of the partition function induces an exponential blowup in the worst case, i.e., the network polynomial has a size exponential in the number of network variables [174, 112]. This leads to difficulties in inference, and in turn in learning. Driven by the question of finding general conditions under which the partition function is tractable, Poon et al. [174] proposed Sum-Product Networks (SPNs) to potentially represent and evaluate network polynomials in polynomial space and time. This model belongs to the Probabilistic Circuits (PCs) family, along with arithmetic circuits [39], AND/OR-graphs [140], Cutset Networks (CNets) [180], and Probabilistic Sentential Decision Diagrams (PSDDs) [109]. The formal definition of SPNs is given by definition 2.

Definition 2 ([174])**.** *A Sum-Product Network S over variables $X_1, \ldots, X_d$ is a rooted*

directed acyclic graph whose leaves are univariate random variables $X_1, \ldots, X_d$ and whose internal nodes are sums and products. Each edge (i, j) emanating from a sum node n_i has a non-negative weight W_{ij}. The value of a product node is the product of the values of its children. The value of a sum node is $\sum_{c_j \in Ch(n_i)} W_{ij} V_j$ for each child of n_i $Ch(n_i)$, c_j, and its value V_j. The value of a SPN S is the value of its root which represents a function of the univariate variables $S(X_1, \ldots, X_d)$. The scope of an arbitrary node $sc(n_i)$ in a SPN is a subset of the univariate variables which its value is propagated from, i.e.

$$sc(n) = \begin{cases} X_i, & \text{if } n_i \text{ is a univariate distribution over } X_i \\ \cup_{c_j \in Ch(n_i)} sc(c_j), & \text{otherwise.} \end{cases}$$

The scope of a SPN is the set of variables that appear in S.

In contrast to PGMs, where nodes represent random variables, nodes in SPNs represent computations instead. With edges denoting data flow, the whole SPN structure then builds a computational graph representing the network polynomial [39], which in turn yields a multilinear function of the univariate variables. As noted previously, the fundamental difference between neural networks and PGMs is whether the graphical units are to be considered as computational nodes or random variables [12]. In this sense, SPNs are more like neural networks in which hidden neurons can only compute two types of arithmetic operations: weighted sums and products, and input neurons are Probability Density Functions (PDFs) [244]. However, SPNs also have a probabilistic interpretation for their graphical structures [170]. This will be further elaborated in Section 4.6.

$S(\mathbf{X})$ for all $\mathbf{X} \in \mathcal{X}$ define an unnormalized probability distribution over $\mathcal{X}$. The unnormalized probability of evidence x under the distribution $S(\mathbf{X})$ is given by $\Phi_S(\mathbf{x}) = \sum_{\mathbf{X} \in \mathbf{x}} S(\mathbf{X})$. The partition function of this distribution is $Z_S = \sum_{\mathbf{X} \in \mathcal{X}} S(\mathbf{X})$.

Definition 3 ([174]). *A Sum-Product Network is* valid *iff $S(\mathbf{x}) = \Phi_S(\mathbf{x})$ for all evidence $\mathbf{x}$.*

With additional structural constraints, a wide range of tractable inference routines can be guaranteed. These constraints are built upon the notion of completeness and consistency [174, 171].

Definition 4 ([174]). *A Sum-Product Network is* complete *iff all children of the same sum node n_i have the same scope, i.e. $\forall c_1, c_2 \in Ch(n_i) : sc(c_1) = sc(c_2)$.*

Definition 5 ([174]). *A Sum-Product Network is* consistent *iff no variable appears negated in one child of a product node and non-negated in another.*

Theorem 1 ([174]). *A Sum-Product Network is* valid *if it is complete and consistent.*

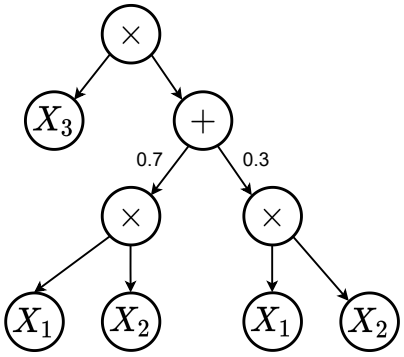


Figure 4.4.: A valid SPN example encoding $P(X_1, X_2, X_3)$.

Definition 6 ([174])**.** *A Sum-Product Network is* decomposable *iff no variable appears in more than one child of a product node, i.e.* $\forall c_1, c_2 \in Ch(n_i), c_1 \neq c_2 : sc(c_1) \cap sc(c_2) = \emptyset$.

Decomposability is a more restricted constraint than consistency. This makes SPNs more general than representations that require decomposability. Completeness and decomposability together guarantee the inference to be efficient. Specifically, inference of joint configurations, marginals and conditionals can be computed in time linear in its size, i.e. tractable. Due to this reason, usually only valid SPNs are considered. Figure 4.4 gives an example of a valid SPN. A SPN is deterministic if every sum node has at most one non-zero child for every instance [39, 243]. Determinism and consistency together are sufficient to guarantee efficient Maximum A Posteriori (MAP) inference.

4.3. Inference

In general, a valid SPN can tractably compute two types of inference: inference of joint configurations and marginals. Conditional queries can also be answered in polynomial time using the marginals, based on the chain rule. MAP inference is generally NP-hard for SPNs [170]. A particular type of SPNs, i.e. selective SPNs [169], allows for tractable MAP inference. A SPN is selective when each sum node has at most one child with positive output, for each possible input and each possible parametrization [169]. This is also known as deterministic SPN in [40, 135] because the sum nodes induce deterministic functions of the input.

To make inference of a joint configuration $\mathbf{x}$, $P(\mathbf{x})$, the leaf nodes are first evaluated by assigning the evidence to the univariate distributions of the corresponding scope. Then the values obtained at the leaf nodes are propagated in an upward pass to the root node. That

means, each node takes inputs from its children and computes the output correspondingly. Upon a sum node, the children's values are weighted by a set of parameters associated with that node and then added together, i.e. $\sum_{j \in Ch(i)} W_{ij} V_j$. Upon a product node, the children's values are simply multiplied together, i.e. $\prod_{j \in Ch(i)} V_j$. Finally, the probability of $\mathbf{x}$, $P(\mathbf{x})$, is induced by the root value. Take the SPN in Figure 4.4 as example, $P(\mathbf{x})$ evaluates to

$$P(x_1, x_2, x_3) = P(x_3) \times (0.7 \times (P(x_1) \times P(x_2))) + 0.3 \times (P(x_1) \times P(x_2))).$$

This requires one upward pass of the network, hence scales linearly in the network size.

Marginal queries of a subset variables $\mathbf{V}_1 \subseteq \mathbf{X}$, $P(\mathbf{V}_1)$, compute the probability of the target variables by summing or integrating out all the rest variables $\mathbf{V}_2$, i.e. $P(\mathbf{V}_1) = \sum_{\mathbf{v}_2} P(\mathbf{V}_1, \mathbf{V}_2 = \mathbf{v}_2)$. To compute marginal inference in SPNs, the leaf nodes within the scope $\mathbf{V}_2$ are first set to 1.0 to induce marginalized leaf distributions, and all the rest leaf nodes are evaluated by their configurations. Then the leaf values are propagated in an upward pass to the root node, inducing the final evaluation of the marginal queries. Using the SPN in Figure 4.4 again, $P(x_2, x_3)$, for example, evaluates to

$$P(x_2, x_3) = 1 \times (0.7 \times (P(x_1) \times P(x_2))) + 0.3 \times (P(x_1) \times P(x_2))).$$

This requires also one time upward pass evaluation, and hence scales linearly in the network size as well.

After computing a marginal inference $P(\mathbf{V}_2)$, conditional queries can be computed by $P(\mathbf{V}_1|\mathbf{V}_2) = \dfrac{P(\mathbf{V}_1, \mathbf{V}_2)}{P(\mathbf{V}_2)}$. This requires two times upward pass evaluation, and hence remains tractable.

To induce the MAP state of a subset of variables $\mathbf{V}_1$ is to find the configuration that maximizes the posterior probability $\arg\max_{\mathbf{V}_1} P(\mathbf{V}_1|\mathbf{V}_2)$ whereby $\mathbf{V}_1 \cap \mathbf{V}_2 = \emptyset$. However, MAP inference for latent variable models is intractable [33]. Fortunately, SPNs that satisfy selectivity and decomposability proved to support tractable MAP again [40]. A SPN is selective if all its sum nodes are selective, which in turn means that only one child of a sum node is non zero for any input. To this end, a SPN is turned into a max-product network by replacing sums by maximizations, which output the maximum weighted value among its children in the upward pass. Then a downward pass is followed starting from the root node whereby max nodes select the maximum-valued child component and product nodes select all children. $\arg\max_{\mathbf{V}_1} P(\mathbf{V}_1|\mathbf{V}_2)$ is then induced by the MAP state of each selected leaf distribution.

Tractable inference routines not only make SPNs appealing but also benefit the gradient-based learning algorithm.

4.4. Learning

Learning a SPN involves learning the structure and the parameters. The parameters consist of the weights for all the sum nodes and the distributional parameters in the leaf nodes.

A popular estimator for parameter learning of SPNs is Maximum Likelihood Estimation (MLE). The goal of MLE is to find parameters $\boldsymbol{\theta}$ that maximize the likelihood of a given dataset D, i.e.

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(D|\boldsymbol{\theta}), \tag{4.3}$$

which in turn can be optimized by the well-known Expectation-Maximization (EM) algorithm [44] or Gradient Descent (GD) [174, 62]. EM is designed to estimate the parameters of probabilistic models under missing data by iteratively carrying out two steps: The E-step (expectation) estimates the expected value to impute the missing data, and the M-step (maximization) uses this completed data to adapt the parameters. These two steps are repeated until convergence. GD updates randomly initialized parameters to incrementally increase the likelihood by

$$\hat{\boldsymbol{\theta}}^{(s+1)} = \hat{\boldsymbol{\theta}}^{(s)} + \gamma \frac{\partial P(D|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{4.4}$$

for a learning rate γ. This is used for parameter learning for SPNs by Poon et al. [174] and Gens et al. [62]

There are multiple algorithms for learning SPN structures. Unlike parameter learning, current structure learning algorithms for SPNs are based on heuristics and less principled. Gens et al. [63] proposed a scheme for learning the structure of SPNs from data. This algorithm recursively splits a SPN into a product or a sum of SPNs until a set of univariate random variables are induced. A product node is induced when it is possible to split the variables into mutually independent subsets, otherwise a sum node is induced by clustering the training instances into similar subsets. Then the algorithm recuses on the resulting split subsets for the same procedure until each subset contains only one random variable. The parameters are learnt automatically along with the structure. In particular, the weight of a sum node is the fraction of instances in the corresponding subset, and the distributional parameter of a leaf node is estimated by the corresponding subset it obtained at the innermost recursion by means of MLE. This scheme is summarized in Algorithm 3.

LearnSPN can be instantiated by various subroutines that partition the variables and instances into subsets. For instance, the clustering subroutine can be completed using the EM algorithm [44]. The splitting subroutine is instantiated in [63] by G-test of pairwise

Algorithm 3: LearnSPN(T, V) [63]

Input : Set of instances T and set of variables V.
Output : A SPN representing a distribution over V learned from T.

if $|V| = 1$ **then**
  return univariate distribution estimated from the variable's values in T
else
  partition V into approximately independent subsets V_j
  if *success* **then**
    return $\prod_j$ LearnSPN(T, V_j)
  else
    partition T into subsets of similar instances T_i
    return $\sum_i \frac{|T_i|}{T} \cdot$ LearnSPN(T_i, V)
  end
end

independence. Specifically, a graph with a node for each variable is initialized with no edge, then an edge is added to each pair of variables that are dependent according to G-test. In the end, the disconnected subgraphs indicate the independent groups of variables.

LearnSPN has been demonstrated by comprehensive empirical evidence to be comparable to graphical models in likelihood [63]. However, most structure learning algorithms for SPNs, including LearnSPN, are based on intuition and lack an explicit principle [241]. Therefore Trapp et al. [241] propose Bayesian learning of SPNs by phrasing structure learning as Bayesian inference in a latent variable model. This approach decomposes structure learning into searching over computational graphs and nested learning of the SPN's scope function using Gibbs sampling. Empirical studies demonstrate that this approach works well and possesses additional benefits, such as prevention against overfitting and robust learning under missing data.

Originally proposed on Bernoulli and Gaussian domains, SPNs have multiple adaptions and extensions. Molina et al. [149] extend SPNs to Poisson domain and Molina et al. [150] extend SPNs to hybrid domains. Peharz et al. [172] propose to employ SPNs for deep learning by demonstrating a random SPN structure combined with simple training mechanisms like soft EM or Adam work surprisingly well.

4.5. Random and Tensorized SPNs

As mentioned in the previous section, valid SPNs need to satisfy completeness and consistency to guarantee tractable inference routines. These structural constraints pose an additional complication to the learning algorithms. Can SPNs be learnt in a simple deep learning way, i.e. by generating unspecialized random structures and applying gradient-based optimization? Peharz et al. gave an affirmative answer: SPNs can not only be learnt in this deep learning approach, but they often even perform on par with state-of-the-art SPN structure learners and deep neural networks on a diverse range of generative and discriminative scenarios [172].

In order to substitute structure learning with a random structure, Peharz et al. [172] introduced a simple and scalable method to construct *Random and Tensorized Sum-Product Networks* (RAT-SPNs) based on the notion of a region graph [45, 168] as an abstract representation of the network structure. A *region* $\mathbf{R}$ is defined as any non-empty subset of a set of RVs $\mathbf{X}$. Given any region $\mathbf{R}$, a K-partition $\mathcal{P}$ of $\mathbf{R}$ is a collection of K non-overlapping sub-regions $R_1, \ldots, R_K$, whose union is again R. A *region graph* $\mathcal{R}$ over $\mathbf{X}$ is a connected DAG whose nodes are regions and partitions. This DAG has several structural constraints: 1) There is exactly one root region, i.e. region without parents. 2) All leaves are regions. 3) All children of regions are partitions and all children of partitions are regions. 4) If $\mathcal{P}$ is a child of $\mathbf{R}$, then $\cup_{\mathbf{R}' \in \mathcal{P}} \mathbf{R}' = \mathbf{R}$. 5) If $\mathbf{R}$ is a child of $\mathcal{P}$, then $\mathbf{R} \in \mathcal{P}$.

Region graphs can be randomly created as follows: Starting from the root, split each region into sub-regions of equal size recursively until depth D. Then repeat this recursive splitting process R times. The resulted random region graph can then be used to construct a SPN: Create I input distributions for each leaf region, S sum nodes for the internal regions and C sum nodes for the root region. The product nodes are created by taking all cross-products of child regions of a partition, which are then connected as children to the sums from their parent region [172].

Given a random SPN structure $\mathcal{S}$ over $\mathbf{X}$, its parameters $\boldsymbol{\omega}$, i.e. sum-weights and parameters of the input distributions, are set via an optimization process. The objective for optimization is the log-likelihood in generative setting. Assume i.i.d. samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ from the distribution $P^*(\mathbf{X})$ that we want to approximate, then

$$\boldsymbol{\omega} = \arg\max \frac{1}{N} \sum_{n=1}^{N} \log \mathcal{S}(\mathbf{x}_n).$$

To optimize for this objective function, the EM algorithm derived for SPNs [170] from the standard EM algorithm [44] can be used. Alternatively, gradient-based optimization techniques can also be used [172].

RAT-SPNs, as standard SPNs, are primarily used for density estimation. However, when this density is represented as a mixture of class-conditional density $\mathcal{S}(\mathbf{X}) = \sum_{Y=y} \mathcal{S}(\mathbf{X}|Y)P(Y)$, it can also be used as a generative classifier for discriminative tasks by applying Bayes' rule:

$$\begin{aligned}\mathcal{S}(Y|\mathbf{X}) &= \frac{\mathcal{S}(\mathbf{X}|Y)P(Y)}{\mathcal{S}(\mathbf{X})} \\ &= \frac{\mathcal{S}(\mathbf{X}|Y)P(Y)}{\sum_{Y=y} \mathcal{S}(\mathbf{X}|Y)P(Y)}.\end{aligned}$$

In this case, each mixture component $\mathcal{S}(\mathbf{X}|Y = y)$ is represented by a SPN or RAT-SPN. See an illustration of this special structure for SPNs in Figure 4.5[1].

To optimize the parameters for discriminative tasks, assume i.i.d. samples $\mathcal{X} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_N, y_N)\}$. The objective function is the *cross-entropy* instead of the log-likelihood, i.e.

$$\boldsymbol{\omega} = \arg\min \frac{1}{N} \sum_{n=1}^{N} \log \frac{\mathcal{S}(\mathbf{x}_n|y_n)}{\sum_y \mathcal{S}(\mathbf{x}_n|y)}.$$

Akin to deep neural networks, RAT-SPNs take a random structure and apply simple parameter estimation techniques. The empirical study proved the capacity of RAT-SPNs as density estimators in generative tasks. In particular, they perform on par with the most prominent SPN structure learner and deep neural networks such as VAEs [108] and Masked Autoencoders (MADE) [64]. For discriminative tasks, RAT-SPNs can also compete with neural networks [172].

4.6. The Connections between SPNs and Other Models

On the architecture level, SPNs can be interpreted as both a probabilistic model and a deep feedforward neural network. Initially used to depict the computation of factored polynomials, SPNs were built on the black box inference machine for classical BNs [39]. As computation graphs, SPNs can naturally be interpreted as peculiar Multi-Layer Perceptrons (MLPs) with sparse and local weight connections where hidden neurons compute weighted sums and products and input neurons are PDFs [244]. Similar to MLPs, SPNs can also be employed for representation learning. In particular, the hidden layers of SPNs can be viewed as extracting a hierarchy of feature representations at different levels of abstractions

[1] Node shade indicates node category.

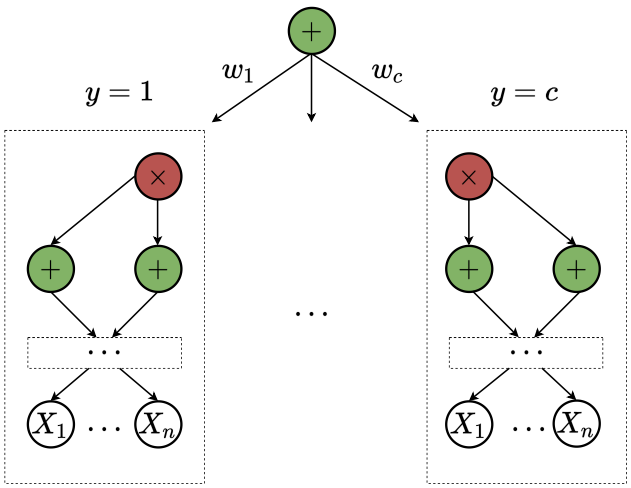


Figure 4.5.: The SPN structure used to estimate a distribution as a mixture of class-conditional densities.

from samples. The difference to MLPs is, by definition, each inner node n directly encodes a valid probability distribution $\mathcal{S}_n(\mathbf{x}_{|sc(n)})$ over the input space indicated by its scope $sc(n)$, resulting in a probabilistic part-based feature representation [244]. Such a representation can be individually visualized by input that maximizes its activation, i.e.

$$\mathbf{x}^*_{|sc(n)} = \arg\max \mathcal{S}_n(\mathbf{x}_{|sc(n)}; \boldsymbol{\theta}),$$

which can in turn be solved by running MPE inference over the scope $sc(n)$ [244]. Vergari et al. [244] further conjectured and empirically proved that the nodes with longer scope length extract representations of a higher abstraction level.

Vergari et al. [244] compared several criteria for extracting embeddings from SPNs by aggregating individual representations of (a) the same node type, (b) comparable scope lengths and (c) the same scope. By training a linear classifier on top of the extracted embeddings to predict unseen RVs, the quality of embeddings is assessed and compared against each other. Scope aggregation proves to be a sensible first option, as those embeddings are both compact enough and have sufficient informative power [244].

Apart from the connection of SPNs to deep learning concerning the network topology, another connection can be established via the way we are learning them. This can be seen through RAT-SPNs [172] introduced in the previous section: SPNs can be learnt akin to DNNs by scaling up a random but valid structure and applying simple gradient-based techniques for parameter estimation. This deep learning view of SPNs eludes sophisticated

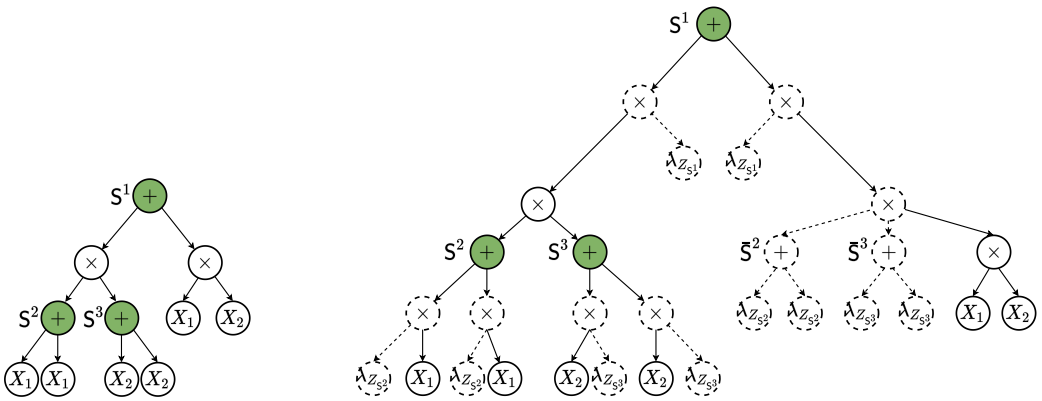


Figure 4.6.: An example of SPN augmentation.

structure learning for SPNs and brings additional advantages to deep learning such as calibrated anomaly detection, treatment of missing features, and most importantly, the power of tractable probabilistic inference [172].

Beyond pure computational representation, SPNs have also probabilistic semantics in their own right [174], unlike their close relatives arithmetic circuits [39] and AND/OR graphs [43]. Above all, each node in SPNs is a valid probabilistic distribution over its scope. Product nodes compute the product of probability values of their children, hence they can be naturally interpreted as factorizing their child distributions, which means the child distributions are independent [170, 171]. Sum nodes are more tricky to interpret. Consider a special SPN over X with a single sum node with K leaf nodes as children, and let $\boldsymbol{w}$ be the weight and $\boldsymbol{\theta}$ be the distributional parameter. The distribution induced by the sum node

$$P(X) = \sum_{k=1}^{K} w_k P(X; \theta_k)$$

can be interpreted as a probabilistic mixture model over X. Mixture models $P(X)$ can in turn be interpreted as marginal distribution of the distribution $P(X, Z)$ over X where Z is a latent variable (LV) associated with the sum node and $P(Z = k) = w_k$ and $P(X|Z = k) = P(X; \theta_k)$. For SPNs with arbitrarily many sum nodes, Peharz et al. [170] formally established the LV interpretation by using an augmentation of SPNs as a theoretical tool. The augmentation process is depicted in Algorithm 4. A concrete example of SPN augmentation is given by Figure 4.6. The notations relevant for the algorithm are the following: The set of all sum nodes and product nodes are respectively

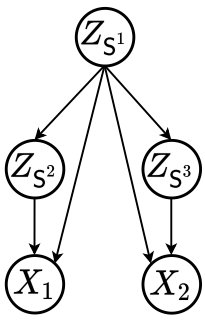


Figure 4.7.: An Bayesian Network representing the dependency of the SPN in Figure 4.6.

denoted as $\mathbf{S}(\mathcal{S})$ and $\mathbf{P}(\mathcal{S})$. For each sum node S, the conditioning sums $\mathbf{S}^c(\mathsf{S})$ is defined as $\{\mathsf{S}^c \in anc_{\mathsf{S}}(\mathsf{S}) \setminus \{\mathsf{S}\} \mid \exists C \in ch(\mathsf{S}^c) : \mathsf{S} \notin desc(\mathsf{C})\}$, whereby $anc_{\mathsf{S}}(N) \coloneqq anc(\mathsf{N}) \cap \mathbf{S}(\mathcal{S})$ and $desc_{\mathsf{S}}(N) \coloneqq desc(\mathsf{N}) \cap \mathbf{S}(\mathcal{S})$.

Based on the augmented SPNs, Peharz et al. [170] further established probabilistic interpretation of the sum weights. For an arbitrary sum node S associated with the latent variable Z_{S}, let $\mathbf{Z}_p$ be the parents, i.e. all the LVs above S, and let $\mathbf{Z}_n$ be all the non-descendants variables that are neither above nor below S. $\mathcal{Z}$ and $\bar{\mathcal{Z}}$ denote a partition of the values of $\mathbf{Z}_p$, i.e. $\mathcal{Z} \cup \bar{\mathcal{Z}} = val(\mathbf{Z}_p), \mathcal{Z} \cap \bar{\mathcal{Z}} = \emptyset$. $\bar{w}_k$ denotes the twin-weight of w_k. Then the weights of S can be interpreted as follows:

$$\mathcal{S}(Z_{\mathsf{S}} = k|\mathbf{Z}_n, \mathbf{Z}_p = \mathbf{z}) = \mathcal{S}(Z_{\mathsf{S}} = k|\mathbf{Z}_p = \mathbf{z}) = \begin{cases} w_k, & \text{if } \mathbf{z} \in \mathcal{Z} \\ \bar{w}_k, & \text{if } \mathbf{z} \in \bar{\mathcal{Z}}. \end{cases}$$

That means, given the path from the root node up to S, the sum weights of S can be viewed as conditional probability tables of choosing the k-th child node and the choice is independent of its non-descendants. This interpretation in turn allows a BN representation by connecting $\mathbf{Z}_p$ as parents of Z_{S} and all leaf variables within the scope $sc(\mathsf{S})$ as children of Z_{S} for each sum node [170]. Following the SPN in Figure 4.6, its BN representation is given in figure 4.7.

The connection of augmented SPNs with BN representation described above gives a useful tool for interpreting SPNs in the context of probabilistic graphical models [170]. Interestingly, this connection aligns with the relationship between BNs and SPNs established by Zhao et al. [255].

This concludes Chapter 4. We started with network polynomial which is the key idea behind SPNs. Then we reviewed SPNs by their representation, inference algorithm and learning scheme. As computation graphs, SPNs are to be contrasted with probabilistic

Algorithm 4: AugmentSPN($\mathcal{S}$) [170]

Input : A SPN $\mathcal{S}$.
Output : An augmented SPN $\mathcal{S}'$.

$\mathcal{S}' \leftarrow \mathcal{S}$
$\forall \mathsf{S} \in \mathbf{S}(\mathcal{S}'), \forall k \in \{1, \ldots, K_S\}$: let $w_{\mathsf{S},k} = w_{\mathsf{S},\mathsf{C}^k_{\bar{S}}}$, $\bar{w}_{\mathsf{S},k} = \bar{w}_{\mathsf{S},\mathsf{C}^k_S}$
for $\mathsf{S} \in \mathbf{S}(\mathcal{S}')$ **do**
    for $k = 1, \ldots, K_{\mathsf{S}}$ **do**
        Introduce a new product node $\mathsf{P}^k_{\mathsf{S}}$ in $\mathbf{S}(\mathcal{S}')$
        Disconnect $\mathsf{C}^k_{\mathsf{S}}$ from S
        Connect $\mathsf{C}^k_{\mathsf{S}}$ as child of $\mathsf{P}^k_{\mathsf{S}}$
        Connect $\mathsf{P}^k_{\mathsf{S}}$ as child of S with weight $w_{\mathsf{S},k}$
    end
end
for $\mathsf{S} \in \mathbf{S}(\mathcal{S}')$ **do**
    for $k \in \{1, \ldots, K_{\mathsf{S}}\}$ **do**
        Connect new indicator variable $\lambda_{Z_{\mathsf{S}}=k}$ as child of $\mathsf{P}^k_{\mathsf{S}}$
    end
    if $\mathbf{S}^c(\mathsf{S}) \neq \emptyset$ **then**
        Introduce a twin sum node $\bar{\mathsf{S}}$ in $\mathcal{S}'$
        $\forall k \in \{1, \ldots, K_{\mathsf{S}}\}$: connect $\lambda_{Z_{\mathsf{S}}=k}$ as child of $\bar{\mathsf{S}}$ and let $w_{\bar{\mathsf{S}},\lambda_{Z_{\mathsf{S}}=k}} = \bar{w}_{\mathsf{S},k}$
        for $\mathsf{S}^c \in \mathbf{S}^c(\mathsf{S})$ **do**
            for $k \in \{k \mid \mathsf{S} \notin desc(\mathsf{P}^k_{\mathsf{S}^c})\}$ **do**
                Connect $\bar{\mathsf{S}}$ as child of $\mathsf{P}^k_{\mathsf{S}^c}$
            end
        end
    end
end
return $\mathcal{S}'$

graphical models where model representation and inference algorithm are decoupled. Then we reviewed a "deep learning" approach for SPNs. In the end, we discussed different perspectives of SPNs, which allows a SPN structure to be interpreted by its BN representation.

5. Conditional Sum-Product Networks

In the previous chapter, we have reviewed SPNs for joint probabilistic distributions. They are gaining momentum in probabilistic modelling because they achieve a good balance between expressivity and tractability, which is to be contrasted with classical probabilistic models. Although a joint distribution in theory allows for flexible probabilistic inferences such as conditional inference and marginal inference, they are not optimized for such tasks. As Choi et al. [31] have shown, the joint density of a SPN is a polynomial, actually a multi-linear function. In turn, any conditional query $P(\mathbf{Y}|\mathbf{X}) = P(\mathbf{Y}, \mathbf{X})/P(\mathbf{X})$ to the SPN is a quotient of multi-linear functions. Consequently, some functions may only be (approximately) fitted using a large number of e.g. Gaussian leaf distributions, which is wasteful if only a specific conditional distribution is concerned. In this chapter, we introduce a conditional counterpart for SPNs called *Conditional Sum-Product Networks* (CSPNs), which are used to model conditional probabilistic distributions with a fixed set of conditioning variables.

This chapter is based on Shao et al. 2020 and organized as follows. First, we briefly review related work on conditional probabilistic modeling in Section 5.1. Then we set our focus on the model representation of CSPNs in Section 5.2 with a formal definition analogous to SPNs, which is followed by a heuristic learning scheme in Section 5.3. Afterwards, we evaluate CSPNs and demonstrate their effectiveness in conditional probabilistic modelling based on empirical evidence in Section 5.4.

5.1. Conditional Probabilistic Models

Principled conditional probabilistic modelling has been an active area of research. Models for conditional probability distributions aim at learning $P(\mathbf{Y}|\mathbf{X})$ over disjoint sets of output variables $\mathbf{Y}$ and input variables $\mathbf{X}$, where $\mathbf{X}$ is assumed to be always fully observed. This is closely associated with supervised predictive tasks such as classification and regression. Except for the use in predictive tasks, conditional probabilistic models serve as basic building blocks for many machine learning models as well. A few common examples are Bayesian Networks [94, 167], Gaussian Processes [184], Conditional Random Fields

[116], Dependency Networks [82], Hidden Markov Models [178] and Clustering [155]. The representational and predictive capacity of these models are then tied to how well they capture the underlying conditional probabilistic distributions. Therefore, having more powerful conditional probabilistic models can have a broad impact on many applications.

A naive way of modelling $P(\mathbf{Y}|\mathbf{X})$ is to model $P(Y_i|\mathbf{X})$ for each $Y_i \in \mathbf{Y}$ independently and multiply these univariate predictors together $\prod_i P(Y_i|\mathbf{X})$. This approach is built on the mean-field assumption that each output variable Y_i is independent of the others given $\mathbf{X}$, which is often a too strong assumption. More expressive alternatives include GPs [184] and CRFs [116]. They are particularly popular representatives for Structured Output Prediction (SOP), both for regression and classification settings. However, their inference has a serious problem when scaling to high-dimensional data or large datasets. One approach to scaling GPs to larger datasets is based on deep mixtures of GPs, proposed by Trapp et al. [240], which can be viewed as a combination of GPs and SPNs. However, they are limited to continuous domains. Other alternatives within the family of tractable probabilistic models are Logistic Circuits (LCs) [123], discriminative learning of SPNs [62], Discriminative Arithmetic Circuits (DACs) [195] and Sum-Product-Quotient Networks (SPQNs) [215]. Recently proposed as a discriminative model, LCs showed classification accuracy competitive to neural nets on a range of benchmarks. However, both LCs and discriminative learning of SPNs are limited to single output prediction. DACs are learned via a compilation of CRFs, thus requiring sophisticated and potentially slow structure learning routines. Extending SPNs by introducing quotient nodes, SPQNs represent a conditional distribution $P(\mathbf{Y}|\mathbf{X})$ as the ratio $\frac{P(\mathbf{Y},\mathbf{X})}{P(\mathbf{X})}$ where the two terms are modeled by two SPNs. However, the expressiveness of SPQNs is limited by the capacity of SPNs and their representation is not very compact due to the additional modelling of $P(\mathbf{X})$. Rahman et al. [179] proposed Conditional Cutset Networks (CCNs), a strict sub-class of SPNs, but they did neither employ conditional independence tests for learning the structure nor establish the link to neural nets.

Another line of research focuses on parameterizing conditional probabilistic distributions by neural nets. Popular examples are VAEs (reviewed in Section 3.4) [108], hierarchical variational models [182] and normalizing flows [186]. However, VAEs and hierarchical variational models involve sampling, hence inducing significant computational costs and highly intractable models. Normalizing flows allow for tractable inference, but they are limited to continuous distributions and exhibit significant computational overhead for computing the determinant of the Jacobian.

To improve upon the aforementioned limitations of conditional probabilistic models, we propose CSPNs, which will be presented in the next section. In summary, we make the following contributions:

1. We introduce CSPNs for representing conditional probabilistic distributions which allow for a set of tractable inference tasks.

2. We present a structure learning algorithm for CSPNs based on Randomized conditional Correlation Test (RCoT) which allows learning structures from heterogeneous data sources.

3. We connect CSPNs with DNNs, and demonstrate that the improved ability of the resulting neural CSPNs to model dependencies results in increased multilabel image classification performance.

5.2. Representation

CSPNs are deisnged to represent multivariate conditional probabilistic distributions $P(\mathbf{Y}|\mathbf{X})$ by extending SPNs. Specifically, SPNs over the output variables $\mathbf{Y}$ are parameterized by external functions $f(\mathbf{X})$ whereby $f(\cdot)$ can take a flexible form and $\mathbf{X} \cap \mathbf{Y} = \emptyset$. Analogous to SPNs, a formal definition of CSPNs is given as follows.

Definition 7. *A Conditional Sum-Product Network (CSPN) $\mathcal{S}$ over the output variables $Y_1, \ldots, Y_m$ conditioned on the input variables $X_1, \ldots, X_n$ is a rooted directed acyclic graph consisting of three types of nodes: leaf nodes, product nodes and gating nodes. Leaf nodes are univariate conditional random variables parameterized by $X_1, \ldots, X_n$. Gating nodes and product nodes are internal nodes. Each edge (i, j) emanating from a gating node i has a non-negative weight W_{ij} parameterized by $X_1, \ldots, X_n$. The value of a product node is the product of the values of its children. The value of a gating node i is the sum of the values of its children V_j weighted by W_{ij}, i.e. $\sum_{j \in Ch(i)} W_{ij} V_j$. The value of the CSPN $\mathcal{S}$ is the value of its root which represents a function of the output variables conditioned on the input variables $\mathcal{S}(Y_1, \ldots, Y_m | X_1, \ldots, X_n)$. The scope of an arbitrary node in the CSPN is a subset of the output variables which its value is propagated from. The scope of the CSPN is the set of output variables that appear in $\mathcal{S}$.*

Akin to unconditional SPNs, nodes in CSPNs have semantics in their own right: Leaf nodes represent univariate conditional distributions $P(Y_i|\mathbf{X})$ for each $Y_i \in \mathbf{Y}$. A gating node i, as a generalization of a sum node, is closely related to gate in Mixture of Experts (MoEs) [216] and can be viewed as a conditional mixture distribution $\sum_{j \in Ch(i)} f_j(\mathbf{X}) P_j(\mathbf{Y}|\mathbf{X})$. A product node h represents a factorial distribution of the form $\prod_{k \in Ch(h)} P_k(Y_k|\mathbf{X})$, assuming their child components are conditionally independent of each other given $\mathbf{X}$. The whole structure is then recursively composed of local conditional probabilistic distributions from the bottom up to the root node.

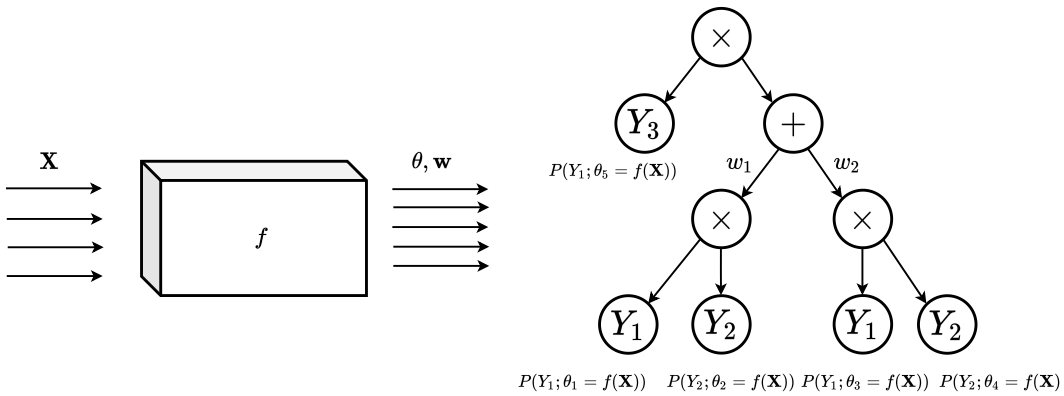


Figure 5.1.: A valid CSPN example encoding $P(Y_1, Y_2, Y_3|\mathbf{X})$.

In analogy to SPNs, a wide range of tractable inference routines in CSPNs assume specific structural constraints. These constraints are built upon the notion of completeness and consistency.

Definition 8. *A CSPN is* complete *iff all children of the same gating node have the same scope of the output variables.*

Completeness ensures that the gating nodes $\sum_{j \in Ch(i)} f_j(\mathbf{X}) P_j(\mathbf{Y}|\mathbf{X})$ compute proper mixture models.

Definition 9. *A CSPN is* consistent *iff no output variable in one child of a product node overlaps with another child.*

Definition 10. *A CSPN is* valid *if it is complete and consistent.*

A valid CSPN computes the conditional probability in time linear in its network size, which in turn makes them easier to learn. Due to this reason, we consider only valid CSPNs in our work. See Figure 5.1 for a valid CSPN example. Essentially, $f(\cdot)$ takes $\mathbf{X}$ as input and produces parameters for the SPN structure, i.e. weights of the gating nodes and distributional parameters on the leaf nodes. Gating weights are constrained to sum to unity in order to ensure normalized mixtures. When $f(\cdot)$ is neural nets, we call the induced network *neural-CSPNs*.

Assuming the input variables are always completely observed, CSPNs permit complex inference on the output variables given the input observations. In general, a valid CSPN can tractably and exactly compute two types of inference: conditional queries of a joint configuration of the output variables and marginals of conditional queries. By converting

gating nodes to max nodes, CSPNs also allow for tractable approximate MAP (maximum a-posteriori) inference on the output variables. To run any of these inference tasks, the first step is to evaluate the observations $\mathbf{x}$ by $f(\cdot)$ to induce CSPN parameters. Then the corresponding inference algorithms for SPNs (recall from Section 4.3) are followed.

5.3. Learning

Learning CPSNs involves learning a parameterizing function $f(\cdot)$ and a CSPN structure. We present two approaches: LearnCSPN for learning the structure based on heuristics by extending LearnSPN (see Algorithm 3), which automatically sets the parameters as well, using local data partitions. Another approach for learning the parameters, is by optimizing the conditional likelihood in an end-to-end fashion, where the structure is either randomly generated or learnt using LearnCSPN.

Akin to LearnSPN, LearnCSPN is a greedy algorithm that builds the structure in a top-down fashion by recursively partitioning the data matrix until the termination condition is met. The partitioning occurs for each local data matrix either by the output features $\mathbf{Y}$ or by instances. At each step, it introduces either a leaf node, a product node or a gating node. A product node is created while splitting the local data by output features into independent variables, which are in turn found by means of a statistical test. If no such partitioning is found, a gating node is then created instead by partitioning the local data into different clusters of instances. The algorithm recurses for each slice of the partitioned data. The recursion terminates by creating a leaf node when the local data slice has only one output variable left. Let us now review LearnCSPN more in detail.

Learning Leaf Nodes. Leaf nodes encode univariate conditional distributions by regressing the distributional parameters of Y_i from features $\mathbf{X}$. Take Gaussian outputs as an example, each leaf represents $\mathcal{N}(Y_i; \mu = f(\mathbf{X}), \sigma = 1)$. In order to allow for tractable inference, we require conditional models at the leaf nodes to be normalized. Learning leaf nodes amounts to estimating $f(\cdot)$ from the local data slice in regression problems. In theory, the parametric form of $f(\cdot)$ can be flexibly chosen, for example, Generalized Linear Models (GLMs) [143] or DNNs etc, as long as their outputs are constrained to be compatible with the domain of the distributional parameter. Ideally, a differentiable form is preferred so that leaf nodes can be learnt jointly with the rest of the network in an end-to-end fashion.

Learning Product Nodes. Product nodes are induced to factorize the labels $\mathbf{Y}$ into conditionally-independent subsets, which are in turn found via pair-wise Conditional Independence (CI) tests. By definition, CI test is to determine whether Y_i is independent of Y_j given $\mathbf{X} = \mathbf{x}$, for any value of $\mathbf{x}$ [42]. Conditional independence, denoted as $Y_i \perp\!\!\!\perp Y_j | \mathbf{X}$,

has various characterizations. In terms of density functions, conditional independence can equivalently be characterized as

$$P(Y_i, Y_j|\mathbf{X}) = P(Y_i|\mathbf{X})P(Y_j|\mathbf{X}),$$

which will be used in the following statistical test.

Among the well-established CI testing methods, parametric methods are based on distributional assumptions, which make them less generic. Here, we aim for a non-parametric CI test for the sake of flexibility. In particular, we adopt RCoT [231] among a variety of non-parametric approaches because it is intended for large scale settings and it scales linearly with sample size. Following Fukumizu et al., Fukumizu et al. [59, 60], RCoT specifies conditional independence using the characteristic kernels (e.g. RBFs, Laplacian) $k_{\mathcal{Y}_i}, k_{\mathcal{Y}_j}, k_{\mathcal{X}}$ for the variables $Y_i, Y_j, \mathbf{X}$ with the domains $\mathcal{Y}_i, \mathcal{Y}_j, \mathcal{X}$ whose corresponding Reproducing Kernel Hilbert Spaces (RKHS) are respectively specified by $\mathcal{H}_{\mathcal{Y}_i}, \mathcal{H}_{\mathcal{Y}_j}, \mathcal{H}_{\mathcal{X}}$. Employing the cross-covariance operator $\sum_{Y_iY_j}$ on the RKHS from $\mathcal{H}_{\mathcal{Y}_i}$ to $\mathcal{H}_{\mathcal{Y}_j}$, RCoT is defined as

$$\langle f, \sum\nolimits_{Y_iY_j} g\rangle = \mathbb{E}_{Y_iY_j}[f(Y_i)g(Y_j)] - \mathbb{E}_{Y_i}[f(Y_i)]\mathbb{E}_{Y_j}[g(Y_j)]$$

for all $f \in \mathcal{H}_{\mathcal{Y}_i}$ and $g \in \mathcal{H}_{\mathcal{Y}_j}$. The partial cross-covariance operator $\sum_{Y_iY_j\cdot\mathbf{X}}$ of (Y_i, Y_j) given $\mathbf{X}$ can then be written as

$$\sum\nolimits_{Y_iY_j\cdot\mathbf{X}} = \sum\nolimits_{Y_iY_j} - \sum\nolimits_{Y_i\mathbf{X}} \sum\nolimits_{\mathbf{XX}}^{-1} \sum\nolimits_{\mathbf{X}Y_j}$$

Under mild assumptions, it then holds: if $Y_i \perp\!\!\!\perp Y_j|\mathbf{X}$, then

$$\mathbb{E}_{\mathbf{X}}[P(Y_iY_j|\mathbf{X})] = \mathbb{E}_{\mathbf{X}}[P(Y_i|\mathbf{X})P(Y_j|\mathbf{X})]$$

and in turn $\sum_{Y_iY_j\cdot\mathbf{X}} = 0$. Indeed, there are some special cases where $Y_i \not\perp\!\!\!\perp Y_j|\mathbf{X}$ yet $\sum_{Y_iY_j\cdot\mathbf{X}} = 0$, i.e., this is not an equivalence relation. However, these cases are rarely encountered in practice. Finally, the test statistic estimator is given as $S = n\|\hat{\sum}_{Y_iY_j\cdot\mathbf{X}}\|_F^2$, and the asymptotic distribution of S under the null hypothesis is approximated by the Lindsay-Pilla-Basak method [124] that matches the first $2L$ moments to a finite mixture of Gamma distributions.

Based on the pairwise CI test, we create a graph where the nodes are the variables in $\mathbf{Y}$ and the edge between two nodes Y_i, Y_j is present if we cannot reject the null hypothesis for a specified threshold α. This algorithm is formulated in algorithm 5.

Learning Gating Nodes. Gating nodes are a generalization of sum nodes, which can be viewed as MoEs $\sum_k f_k(\mathbf{X})P_k(\mathbf{Y} \mid \mathbf{X})$ [15]. Instead of constant weights associated with

Algorithm 5: SplitLabels ($\mathcal{D}$, α)

Input : Data $\mathcal{D} = \{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) | \mathbf{y}^{(i)} \in \mathbf{Y}, \mathbf{x}^{(i)} \in \mathbf{X}\}_{i=1}^{N}$ where the label RVs are $\mathbf{Y} = \{Y_1, \ldots, Y_P\}$, α: threshold of significance

Output : A label partitioning $\{\mathcal{P}_\mathcal{D}\}$

$\mathcal{G} \leftarrow \text{Graph}(\{\})$
foreach $Y_i, Y_j \in \mathbf{Y}$ **do**
  $S_{i,j} \leftarrow n \| \hat{\sum}_{Y_i Y_j \cdot \mathbf{X}} \|_F^2$
  if $\text{LindsayPillaBasak}(S_{i,j}) > \alpha$ **then**
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{(i, j)\}$
  end
end
return $\text{ConnectedComponents}(\mathcal{G})$

sum nodes in SPNs, $f_k(\mathbf{X})$ gives dynamic weight for each child component $P_k(\mathbf{Y} \mid \mathbf{X})$ of a gating node conditioned on $\mathbf{X}$. To construct this mixture, $f_k(\mathbf{X})$ is learnt to predict the component assignment of the labels $\mathbf{Y}$, one-hot coded as $\mathbf{Z}$, conditioned on $\mathbf{X}$ under the constraint $\sum_k f_k(\mathbf{X}) = 1$ and $\forall_\mathbf{X} f_k(\mathbf{X}) >= 0$ so that the gating nodes are guaranteed to be normalized. In other words, we estimate $f_k(\mathbf{X})$ to predict $\mathbf{Z}_k = f_k(\mathbf{X})$. The member assignment is in turn induced by clustering the data into disjoint sets according to $\mathbf{X}$. The clustering scheme may be flexibly chosen here, depending on the data distribution. For example, k-Means can be used for Gaussians.

End-to-End Parameter Optimization. CSPNs contain two sets of parameters: one for the weights $\boldsymbol{W}$ of the gating nodes, and another for the parameters $\boldsymbol{\theta}$ of the leaf distributions. Although algorithm 6 automatically sets the parameters, they are only locally optimized using the data slices available at each step. To further optimize the parameters globally in a more principled way, the MLE estimator is used. Let $L(\mathbf{Y}; \mathbf{X})$ denote the likelihood of $\mathbf{Y}$ given $\mathbf{X}$ and $\boldsymbol{\omega}$ be all the learnable parameters of $f(\cdot)$, MLE induces the parameter estimation by

$$\boldsymbol{\omega} = \arg\max \log L(\mathbf{Y}; \mathbf{X}). \tag{5.1}$$

Assuming $f(\cdot)$ to be fully differentiable, $L(\mathbf{Y}; \mathbf{X})$ is hence also differentiable. Therefore, an estimation $\hat{\omega}$ can be obtained using Stochastic Gradient Descent (SGD). The gradient of the loss w.r.t. $\boldsymbol{\omega}$ is given by

$$\frac{\partial L(\mathbf{Y}; \mathbf{X})}{\partial \boldsymbol{\omega}} = \frac{\partial L(\mathbf{Y}; \mathbf{X})}{\partial f(\mathbf{X})} \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \tag{5.2}$$

Algorithm 6: LearnCSPN ($\mathcal{D}$, η, α)

Input : Data $\mathcal{D} = \{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) | \mathbf{y}^{(i)} \in \mathbf{Y}, \mathbf{x}^{(i)} \in \mathbf{X}\}_{i=1}^{N}$; η: minimum number of instances to split; α: threshold of significance.

Output : A CSPN $\mathcal{S}$ approximating $P(\mathbf{Y} \mid \mathbf{X})$ learned from $\mathcal{D}$.

if $|\mathbf{Y}| = 1$ **then**
  $\mathcal{S} \leftarrow \text{LearnConditionalLeaf}(\mathcal{D})$
else if $|\mathcal{D}| < \eta$ **then**
  $\mathcal{S} \leftarrow \prod_{j=1}^{|\mathbf{Y}|} \text{LearnCSPN}(\{(y_j^{(i)}, \mathbf{x}^{(i)})\}_{i=1}^{N}, \eta, \alpha)$
else
  $\{\mathbf{V}_i\}_{i=1}^{K} \leftarrow \text{SplitLabels}(\mathcal{D}, \alpha)$ // compare with Alg. 5
  if $K > 1$ **then**
    $\mathcal{D}_i \leftarrow \{\mathbf{v}_i^j | \mathbf{v}_i^j \sim \mathbf{V}_i\}_{j=1}^{M}$
    $\mathcal{S} \leftarrow \prod_{i=1}^{K} \text{LearnCSPN}(\mathcal{D}_i, \eta, \alpha)$
  else
    $\{\mathcal{D}_i\}_{i=1}^{K} \leftarrow \text{SplitInstances}(\mathcal{D})$ // using e.g. random splits or k-Means with an appropriate metric
    $\mathcal{S} \leftarrow \sum_{i=1}^{K} g_i(x) \cdot \text{LearnCSPN}(\mathcal{D}_i, \eta, \alpha)$, $x \in D$ where $g_i(x)$ is a gating function
  end
end
return $\mathcal{S}$

where $\frac{\partial L(\mathbf{Y}; \mathbf{X})}{\partial f(\mathbf{X})}$ differentiates through the SPN polynomial, and $\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ differentiates through the external function $f(\cdot)$.

Alternatively, MLE can also be directly applied to random structures. This is especially useful when the output dimensions are too high to learn the structure efficiently. In addition, it also makes CSPNs easily fit for building modular structures that can be learnt jointly as a full end-to-end system. This will be demonstrated in Section 6.1.

5.4. Empirical Evaluation

In this section we present experimental evidence to answer the following research questions:

(Q1) Can CSPNs perform better than regular SPNs for predictive tasks?

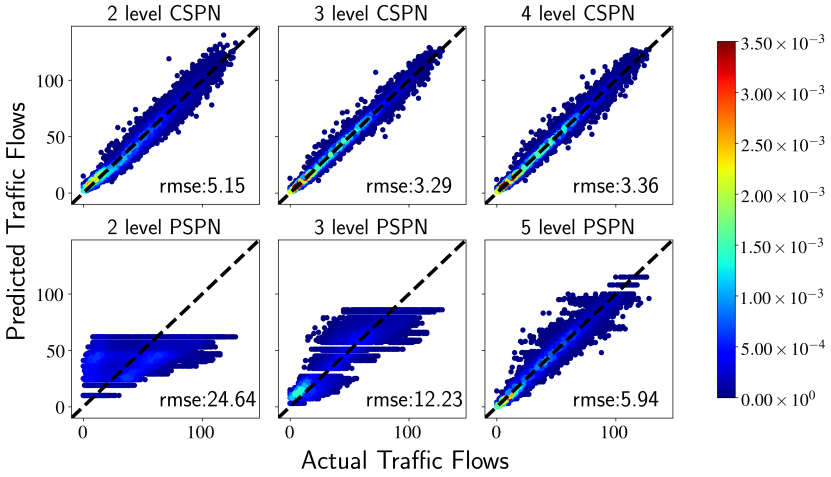


Figure 5.2.: Traffic flow predictions of Poisson CSPNs (top) versus SPNs (botton) for shallow (left) or deep models (center and right). The more examples centered around the diagonal line, the better the predictions are.

(Q2) How accurate are CSPNs for SOP?

(Q3) Do neural-CSPNs outperform other neural conditional models such as MDNs on image SOP tasks?

5.4.1. Traffic Prediction

This experiment aims to answer the research question **(Q1)**. To demonstrate the representational power of CSPNs against SPNs, we studied multivariate traffic data prediction task on the Poisson domain [149]. We considered temporal vehicular traffic flows in the German city of Cologne [95]. The data comprises 39 RVs whose values are from stationary detectors located at the 50km long Cologne orbital freeway, each one counting the number of vehicles within a fixed time interval. It contains 1440 samples, each of which is a snapshot of the traffic flow. The task of this experiment is to predict the next snapshot ($|\mathbf{Y}| = 39$) given a historical one ($|\mathbf{X}| = 39$).

We trained respectively a CSPN and a SPN of three depths. The CSPNs use GLMs with exponential link function as Poisson univariate conditional leaves. Results are summarized in Figure 5.2 by scatter plot. Each point represents the relation between a prediction and its ground truth. When perfectly predicted, a point aligns at the diagonal line. The

Table 5.1.: Average test conditional log-likelihood (CLL) of DACL, CCNs and CSPNs on 20 standard density estimation benchmarks.

		Nltcs	Msnbc	KDD	Plants	Audio	Jester	Netflix	Accidents	Retail	Pumsb.
50% evidence	DACL	-2.770	-2.918	**-0.998**	-4.655	-18.958	-24.830	-26.245	**-9.718**	**-4.825**	**-6.363**
	CCN	**-2.58**	**-2.18**	-1.19	**-4.53**	-18.67	-24.96	-26.03	-10.24	-4.88	-6.98
	CSPN	-2.795	-3.165	-1.023	-4.720	**-18.543**	**-24.543**	**-25.914**	-11.587	-5.600	-7.383
80% evidence	DACL	-1.255	-1.557	-0.386	-1.812	-7.337	-9.998	-10.482	**-3.493**	-1.687	-2.594
	CCN	**-0.99**	**-0.87**	**-0.36**	**-1.54**	-7.58	**-9.75**	**-10.22**	-3.61	-1.66	**-2.02**
	CSPN	-1.256	-1.684	-0.397	-1.683	**-7.110**	-9.830	-10.351	-4.045	**-1.654**	-2.618
		Dna	Kosarek	MSWeb	Book	EachMovie	WebKB	Reuters-52	20News	Bbc	Ad
50% evidence	DACL	-34.737	-5.053	-5.653	-16.801	-23.325	-72.072	-41.544	-76.063	-118.684	-4.893
	CCN	**-32.98**	**-4.76**	**-4.25**	-15.90	-24.85	-69.34	-36.56	-71.69	-114.52	**-3.41**
	CSPN	-38.243	-5.527	-6.686	**-10.653**	**-18.130**	**-18.542**	**-15.736**	**-35.900**	**-47.138**	-6.290
80% evidence	DACL	-12.116	-2.549	**-1.333**	-6.817	-9.403	-28.087	-17.143	-27.918	-44.811	-1.370
	CCN	-12.29	**-1.14**	-1.69	-6.48	-8.40	-25.76	-15.67	-27.72	-43.27	-1.18
	CSPN	**-11.895**	-2.397	-1.335	**-3.191**	**-4.579**	**-2.623**	**-3.878**	**-4.984**	**-2.996**	**-1.030**
50% evidence	**Wins**	DACL: 4	CCN: 7	CSPN: **9**							
80% evidence	**Wins**	DACL: 2	CCN: 8	CSPN: **10**							

more away from the diagonal, the worse a prediction is. Therefore, the more instances aligned along the diagonal, the higher the overall predictive accuracy is. Color denotes density. In horizontal comparison, one can observe that the 2-level CSPN is clearly outperformed by its 3-level and 5-level counterparts in terms of Root Mean Squared Error (RMSE), which demonstrates the benefits of deeper structure. The reason is that the 2-level CSPN constructs a naive factorization and assumes conditional independency between the response variables, but the deeper structure is more flexible in representing the complex dependency between the response variables. In vertical comparison, each CSPN outperforms its PSPN counterpart, in the 2-level case even by a wide margin. As expected, deeper CSPNs have a lower predictive error compared to shallow CSPNs. Moreover, smaller CSPNs perform equally well or even better than SPNs, empirically confirming that CSPNs are more expressive than SPNs. To conclude, CSPNs consistently yield more accurate predictions than the corresponding SPNs and, as expected, deeper CSPNs outperform shallow ones. This answers **(Q1)** affirmatively and also provides evidence for the convenience of directly modelling a conditional distribution.

5.4.2. Conditional Density Estimation

This experiment aims to answer the research question **(Q2)**. To demonstrate the effectiveness of CSPNs for conditional density estimation, we focused on conditional density estimation on 20 standard binary benchmark datasets. The number of variables of these

Table 5.2.: Comparison to mean field models and mixture density networks.

	CLL			Accuracy		
	MF	MDN	CSPN	MF	MDN	CSPN
MNIST	-0.70	-0.61	**-0.54**	74.1%	76.4%	**78.4%**
Fashion	-0.95	-0.73	**-0.70**	73.4%	73.7%	**75.5%**
CelebA	-12.1	-11.6	**-10.8**	86.6%	85.3%	**87.8%**

datasets ranges from 16 to 1556. In order to estimate conditional density, features were split into evidence and target with different proportions[1].

We compared to DACL [195] and CCNs [179] as they currently provide state-of-the-art Conditional Log-Likelihoods (CLL) on such data. We first performed structure learning on the training data (stopping learning when no more than 10% of samples are available), followed by end-to-end parameter optimization on the training and validation data [2].

Results are reported in table 5.1. The best results are given in boldface. Since we did not have access to per-instance CLLs on CCNs, a t-test was not possible, but the results still provide a tendency for comparison. We can see that for both the 80%-evidence scenario and the 50%-evidence scenario, CSPNs have the most wins. In total, CCNs have slightly fewer wins than CSPNs and more wins than DACL. Besides, we note that CSPNs are faster to learn than DACL and that, in practice, no real hyperparameter tuning was necessary to achieve these scores, while DACL ones are the result of a fine-grained grid search (see [195]). This answers **(Q2)** affirmatively and shows that CSPNs are comparable to the state-of-the-art.

5.4.3. Neural Conditional Sum-Product Networks with Random Structures

This experiment aims to answer the research question **(Q3)**. To demonstrate the efficacy of neural-CSPNs with random structures, we evaluated it on several multilabel image classification tasks and compared it against two different common ways of parameterizing conditional distributions using neural networks. Apart from the CelebA dataset, we also constructed our own multilabel versions of the MNIST and the Fashion-MNIST datasets, by adding additional labels indicating symmetry, size, etc. to the existing class labels, yielding 16 binary labels total. The first baseline model is the Mean Field (MF) approximation, whereby the output of a neural network is interpreted as logits of independent univariate

[1] We adopted the data splits of Rooshenas and Lowd [195].

[2] Note that the sophisticated structure learning in DACL directly optimizes for the CLL at each iteration.

Bernoulli distributions, assuming that the labels $\mathbf{Y}$ are conditionally independent given $\mathbf{X}$. Second, we compared to Mixture Density Network (MDN) with 10 mixture components, each itself a mean field distribution. For each of these models, including the CSPN, we used the same standard convolutional neural network architecture up to the last two layers. Those final two layers were customized to the different desired output formats: For the MF and MDN models, all the parameters were predicted using two dense layers. For the CSPN, we used a dense layer followed by a 1d-convolution, in order to obtain the increased number of SPN parameters without using drastically more neural network weights.

The resulting CLL as well as accuracies are given in table 5.2. Each model was based on the same neural network architecture, they differed in how the conditional distribution is parameterized. Accuracies were computed by obtaining MPE estimates from the models using the standard max-product approximation. On the MNIST and the Fashion dataset, estimates were counted as accurate only if all 16 labels were correct, on the CelebA dataset, we report the average accuracy across all 40 labels. The best results are marked in bold. In general, the additional representational power of CSPNs yields notable improvements upon both baselines. In terms of capacity, mean field approximation is lower than MDN, which explains the performance results. Compared to MDN, CSPNs yield a further performance increase due to improved model capacity. On CelebA, CSPN outperforms a number of sophisticated neural network architectures, despite being based on a standard CNN with only about 400k parameters [52]. The results indicate that the mean field approximation is inappropriate on the considered datasets, as allowing the inclusion of conditional dependencies resulted in a pronounced increase in both likelihood and accuracy. This provides an affirmative answer to **(Q3)**.

This concludes Chapter 5. We started by giving background on conditional probabilistic modelling for conditional distributions, then we introduced a novel conditional probabilistic model — CSPNs — allowing for a set of tractable probabilistic inferences. For learning the structure, LearnCSPN (Algorithm 6) was proposed based on heuristics. This algorithm also sets the initial parameters which can be optionally further optimized globally in an end-to-end fashion by maximizing the conditional likelihood. Empirical evidence in Section 5.4 across several datasets and domains confirmed the capacity of CSPNs in predictive tasks.

Part III.

Explaining and Debugging Deep Models

In the previous part, we presented some deep networks for various purposes including neural networks and probabilistic circuits. Neural networks are good for representing complex input-output mapping, i.e. supervised tasks, and probabilistic circuits are thought for density estimation, i.e. unsupervised tasks. It is also possible to repurpose supervised models for unsupervised tasks, and vice versa. CSPNs presented in Chapter 5 are a supervised model with probabilistic circuits, and VAEs in Section 3.4 are a unsupervised model based on neural networks. Despite the different purposes and design principles, DNNs and PCs have a commonality in that they are both computation graphs composed of multiple layers of operations. An intrinsic disadvantage of deep computation graphs is that we are not able to make sense of their decision logic by looking at their architecture and parameters, i.e. they are not interpretable by nature. To present them in understandable terms to a human, a wide range of techniques have been proposed to explain deep models in a post-hoc fashion. See Section 2.1 for an overview.

Why are interpretability and explanations desirable features for deep models? First of all, they *increase the social acceptance* of opaque ML systems for our daily uses [151]. Heider et al. show by experiments in social psychology that humans tend to assign beliefs, desires and intentions to others, including objects [83]. That means, people anthropomorphically characterise deliberative behaviour of others, which is essentially producing explanations that link to the mental states for the act of others [147]. Thus, explaining ML systems is consistent with our behaviour in a social environment.

Secondly, interpretability and explanations are also helpful for humans to *gain knowledge and insights* about the underlying domain [48, 151]. In many domains, it is cheap to collect a lot of data, which is generally hard for us to directly comprehend and make sense of. Machine learning models can automatically extract useful patterns from data. Sometimes the extracted patterns are not directly understandable by humans as well. Interpreting white box models or explaining black box models can offer insights into the extracted patterns, and in turn offer insights into the domain. In a scientific setting, such insights can help us obtain scientific understanding and make discoveries from observational or simulated data in physical, chemical, or biological systems [196, 102, 4, 233, 204, 183]. For example, Bayesian networks applied to the domain of affect recognition can provide an insight into the underlying relationships among physiological features and affective states [183].

Thirdly, Doshi-Velez et al. argue that *interpretability can be used to confirm other important desiderata* — such as fairness, privacy, reliability, robustness, causality, usability and trust — of ML systems [48]. As we know, deep models have demonstrated high representational power and scalable solutions for many complex problems by optimizing for a single metric. However, their further applications are still limited for many scenarios. The problem is that a single metric, such as classification accuracy, is an incomplete

description of most real-world tasks [48, 127]. For example, it is possible for a model to attain high predictive accuracy without actually learning the correct underlying rules [250, 117], which is also known as the Clever Hans effect [173]. This can cause severe consequences for high-risk domains such as medicine, the criminal justice system, and financial markets. Consider skin cancer detection, a popular dataset from ISIC (International Skin Imaging Collaboration) has band-aids present in approximately 50% of the non-cancerous images but not in the cancerous images. A machine learning model may obtain high predictive accuracy by learning a spurious correlation between band-aids and cancer detection [190]. As a consequence, a cancer patient may miss the opportunity to get treated because of a band-aid on the skin. Explanations can help us to identify this bug in the system. A model taking advantage of the Clever Hans effect is of course not robust, hence also not reliable, because it exploits spurious correlations that may not exist in the testing environment. Therefore, it is also hard for such a model to gain human trust even if they achieve high predictive accuracy on a limited dataset. Sometimes, the spurious correlations can be linked to a protected attribute such as gender or race. Take a widely deployed commercial risk-prediction tool in the United States as an example, which is used for identifying patients who will derive the greatest benefit from "high-risk care management" programs. This tool is found to exhibit a significant racial bias: At a given risk score, Black patients are considerably sicker than White patients [161]. The reason is that the predictive model determines the medical needs by the expected medical expenditures, which are on average lower for Black patients than White patients. Such disparities in the model's performance on faces of different races put fairness at great risk. More importantly, models, that reproduce existing patterns of bias in society, can also reinforce and propagate bias further on.

The aforementioned pathological behaviours imply a bug in the model. However, due to the opaqueness of ML systems, specifically deep models, it can be hard to identify such bugs without formalized criteria. Post-hoc explanations can help us diagnose model errors and confirm desiderata other than a single performance metric such as predictive accuracy [48]. Specifically, explanations that contradict our intuitions or prior knowledge on the underlying domain can expose a potential concern. In case we identify an unfulfilled desideratum in a deep model, how can we improve the model based on the explanations we get? In other words, how can we guide the model to produce more sensible explanations while keeping the predictive accuracy reasonably high? Of course, one can always retrain the model from scratch using a reworked dataset. However, this is both time- and resource-consuming. To efficiently tackle this task, we use explanations directly as debugging tools to fix model errors. We build our work on the framework of XIL [237] (see Section 2.2.2) to put users in the training loop and interactively take user feedback on explanations as additional training signals. This accords with the idea that the explanation process does

not stop at just selecting an explanation, an explanation is instead an interaction between explainer and explainee [147]. This way, the models can be improved in their task-specific performance and learn the rules that align with our intuitions or prior knowledge on the underlying domain by interactively regularizing the wrong rules.

This part is organized as follows. First, we show how to enhance the high-level interpretability of probabilistic circuits by imposing a domain structure in Chapter 6. Then, we present an efficient approach for explaining individual inferences in probabilistic circuits in Chapter 7. Afterwards, we move on to explain deep neural networks, whereby spurious correlations are found. Although the models are right for the predictions, they are right for the wrong reasons. To learn models that are right for better reasons, we demonstrate our explanatory interactive approach in Chapter 8. Finally, we also explain latent representations in VAEs, whereby spurious correlations in the dataset cause the model to learn the wrong latent factors. We develop another approach within the framework of explanatory interactive learning to learn the right latent factors, which is presented in Chapter 9.

6. Imposing Interpretable Structure

Probabilistic Graphical Models (PGMs) allow for natural interpretation of the underlying domain [112] due to their rich structure with probabilistic semantics represented by graphs. Specifically, one can interpret a domain by directly reading off the interactions and dependencies between features. Take a simple Bayesian network as an example: Burglary $\rightarrow$ Alarm $\leftarrow$ Earthquake. One can get the following interpretations:

1. Burglary and Earthquake are a priori independent.
2. Alarm is dependent on both Burglary and Earthquake.
3. When Alarm is observed, Burglary and Earthquake become dependent due to the explaining away effect [112]. That means, knowing that Burglary is true, then Earthquake becomes less likely, and vice versa.

However, the practical use of PGMs is restricted due to the fact that they face an inherent trade-off between expressiveness and tractable inference — their inference task faces exponential blowup in the worst case. Compiling PGMs into PCs such as arithmetic circuits or SPNs has proven an effective approach for inference [28]. PCs allow for tractable inference, i.e. the cost of exact inference is linear in the size of the network. However, they are less interpretable than their equivalent PGMs for the following reasons: First, their representations are designed for computations and not for probabilistic semantics. With slight modifications, they can potentially have probabilistic interpretations with a semantic structure [170]. As we noted for SPNs in Section 4.6, every sum node corresponds to a mixture over a subset of variables and every product node corresponds to a mixture component. SPNs can in turn be viewed as generalized directed acyclic graphs of mixture models. A mixture model has in turn a Latent Variable (LV) interpretation, which yields a syntactically well-structured model [170]. However, this usually requires expertise to interpret. In addition, SPNs are less compact than their equivalent PGMs in representation, which also makes interpretation impractical, if not impossible. See Figure 6.1 for the aforementioned BN and its corresponding SPN to compare the compactness of both representations.

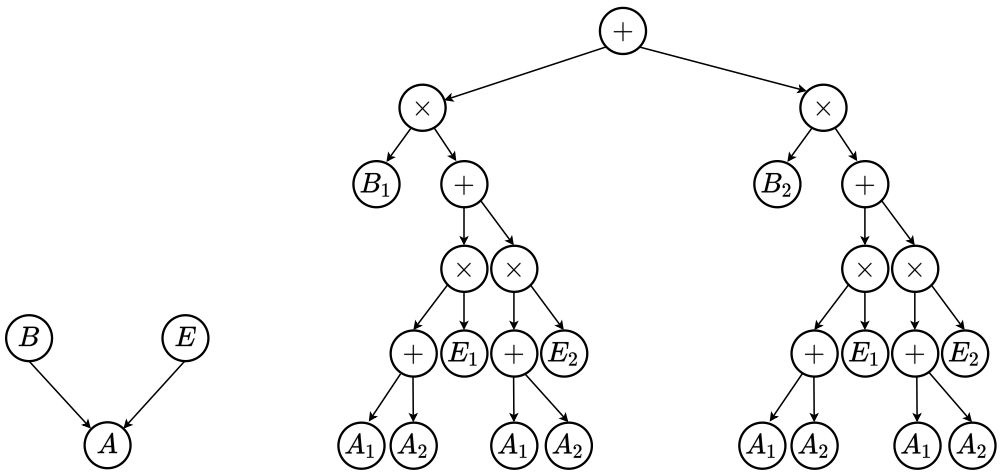


Figure 6.1.: Left: A simple Bayesian Network. B, E and A stand for Burglary, Earthquake and Alarm. Right: The corresponding Sum-Product Network. SPNs are less compact than their corresponding BNs in representation.

So how can we maintain both a high-level domain structure for interpretability and a range of tractable inference simultaneously? Driven by this goal, we propose to use CSPNs as building blocks for joint probabilistic models to impose modular structures, which can reflect assumptions or prior knowledge on the domain. Modularity can be formed using the chain rule in probability theory. In other words, we parameterize local conditional distributions of directed graphical models with CSPNs. This can bring several benefits: On one hand, CSPNs can benefit a given graphical model by allowing for efficient inference, which expands the practical use of the graphical model. On the other hand, the graphical model can help users to obtain a high-level interpretation of the underlying domain. Besides, modularity also helps machine learning practitioners to integrate background knowledge or assumptions in the form of conditional independencies in highly complex spaces. In addition, modularity allows for flexible modelling of different granularity. Take again the graphical model in Figure 6.1 as an example: One can build a module for each variable, which in effect results in the same BN parameterized by CSPNs. Alternatively, a module may be built on both the Burglary variable and the Earthquake variable when the interaction between them is not known or not interested.

In this chapter, we will demonstrate the effectiveness of CSPNs in building modular structures for two different generative models. First, we present an autoregressive model composed of CSPNs for sequential image generation in Section 6.1. Then, we present

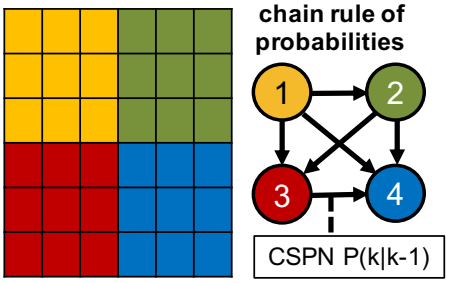


Figure 6.2.: Graphical illustration of an ABCSPN.

Figure 6.3.: Olivetti faces generated by an ABCSPN.

VAEs whose encoder and decoder are built upon CSPNs in Section 6.2.

6.1. Modular Probabilistic Modelling via Conditional Sum-Product Networks

In this section, we demonstrate generative modelling with an autoregressive model composed of CSPNs, where a flexible high-level structure is imposed based on the chain rule in probability theory.

By the chain rule of probability, a joint distribution can be factorized using a set of conditional probability distributions as

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} P(X_i | X_{<i}),$$

which forms the structure of autoregressive models. Inspired by image autoregressive models like PixelCNN [163] and PixelRNN [164], we propose *Autoregressive Block-wise*

Conditional Sum-Product Networks (ABCSPNs) to factorize image distributions $\mathbf{I}$ over n disjoint pixel blocks $\mathbf{B}_1, \ldots, \mathbf{B}_n$ as

$$P(\mathbf{I}) = \prod_{i=1}^{n} P(\mathbf{B}_i \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{C}) \cdot P(\mathbf{C})$$

where $\mathbf{C}$ is the one-hot coded image class. Each factor $P(\mathbf{B}_i \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{C})$ is captured by a CSPN that represents the distribution of a block of pixels $\mathbf{B}_i$ conditioned on all the previous blocks and on the class label. The prior distribution of the class labels $P(\mathbf{C})$ is represented by a SPN. A graphical illustration of ABCSPNs can be found in Figure 6.2.

In our experiment, we used Olivetti faces and divided the images into 64 blocks of equal size which were sequenced in raster scan order[1], i.e. row by row and from left to right. We trained a CSPN on the Gaussian domain for each block conditioned on all the blocks above and to the left of it and on the image class. The distribution of the images is then the product of all the CSPNs. In order to inspect this joint distribution, we generated some samples for visual inspection. In Figure 6.3, images of new faces (bottom row) were sampled from the ABCSPN. Each sample was sequentially generated block by block after conditioning on the mix of two original classes (shown in the top and middle row) in the Olivetti dataset, i.e. $\mathbf{C} = \text{one-hot}(C_1) + \text{one-hot}(C_2)$. This allows us to generate completely new samples that resemble prominent features from both classes. These samples show that ABCSPNs are able to learn meaningful and accurate models over the image manifold. Notably, ABCSPNs achieve this while reducing the number of independence tests among pixels required by the CSPN learning algorithm: from quadratic over all pixels in an image down to quadratic in block size.

6.2. Sum-Product Variational Auto-Encoders

In this section, we present another generative model — Variational Autoencoders (VAEs) — that uses CSPNs as modular building blocks. The graphical model behind VAEs is illustrated in Figure 6.4. It is a generative model $P(\mathbf{X}, \mathbf{Z})$ over the unobserved variables $\mathbf{Z}$ and the observed variables $\mathbf{X}$. The unobserved variables have an interpretation as a latent representation or code from which the observation $\mathbf{X}$ could have been generated [108]. A recognition model $q(\mathbf{Z}|\mathbf{X})$ approximates the intractable true posterior $p(\mathbf{Z}|\mathbf{X})$ using a CSPN, which can be interpreted as a probabilistic encoder. The likelihood function $p(\mathbf{X}|\mathbf{Z})$ is approximated using a CSPN as well, which can be interpreted as a probabilistic decoder.

As we mentioned in Section 3.4, the prior distribution of the latent factors $P(\mathbf{Z})$ in VAEs is often assumed to be isotropic Gaussian, i.e. $P(\mathbf{Z})$ is fully factorial, which is known as the

[1] Other partitions and ordering are also possible.

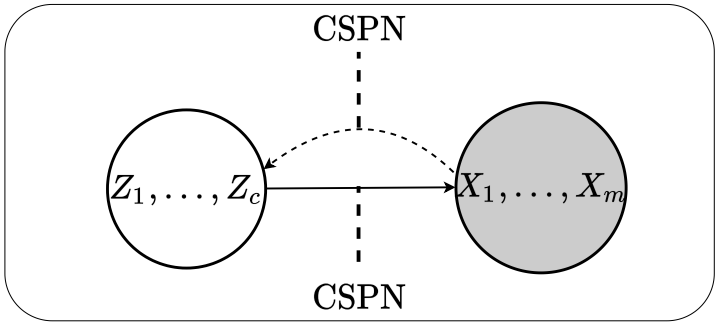


Figure 6.4.: The graphical model behind the VAEs that uses CSPNs as modular building blocks.

mean field assumption. This simplified assumption encourages VAEs to learn disentangled representations in the latent space, i.e. no dependence between the latent dimensions. As a consequence, the expressivity of mean field variational distribution for approximating the true posterior distribution is very limited, which is especially a problem when the disentanglement assumption does not hold. Exploring richer variational families would enable a tighter bound of the log partition function, and in turn bound the probability of evidence [217].

Using CSPNs as the encoder and the decoder has the advantage of allowing for more flexible and expressive variational distributions as well as the data generating distributions. We call this model *Sum-Product Variational Autoencoders* (SPVAEs). To test the representational power of SPVAEs, we do not only consider isotropic multivariate Gaussian $p(\mathbf{Z}) = \mathcal{N}(\mathbf{Z}; \mathbf{0}, \mathbf{I})$ as the prior distribution of the latent variables [108], but also more complex prior distributions where the mean field assumption is not appropriate anymore. Note that we focus only on continuous latent variables.

Optimizing the ELBO with standard gradient-based methods involves differentiating through stochasticity, i.e. a Monte Carlo estimation of the expectation $\mathbb{E}_q[\log p_\psi(\mathbf{X}|\mathbf{Z})]$ and $\mathbb{D}_{\mathrm{KL}}[q_\omega(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z})]$. To make this possible, the reparameterization trick is used to reformulate sampling as a deterministic base distribution and independent noise with a fixed distribution [108, 187, 98]. For CSPNs, stochasticity in the standard sampling process stems from sampling leaf nodes and sum nodes. Specifically, sampling leaf nodes amounts to sampling from univariate Gaussian distributions $\mathcal{N}(Z|\mu, \sigma^2)$ and the reparameterization trick reformulates Z as $Z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. Sampling from sum nodes amounts to sampling from categorical distributions. The reparameterization trick for categorical

Algorithm 7: $\text{Sample}(n_i, \tau, x)$ // differentiable sampling

Input : Node n_i parameterized via $f(\cdot)$, temperature τ, observations $\mathbf{x}$.
Output : Samples $\mathbf{z}_i$.

if n_i *is leaf node* **then**
  $\boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow f(\mathbf{x})$
  $\boldsymbol{\epsilon} \leftarrow$ i.i.d. samples from $\mathcal{N}(0, 1)$
  return $\mathbf{z}_i = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$
else if n_i *is sum node* **then**
  $\boldsymbol{\pi} \leftarrow f(\mathbf{x})$
  foreach *child* n_{ij} *of* n_i **do**
    $z_{ij} = \text{Sample}(n_{ij}, \tau, \mathbf{x})$
    $\boldsymbol{\delta} \leftarrow$ i.i.d. samples from $\text{Gumbel}(0, 1)$
    $w_{ij} = \frac{\exp((\delta_j + \log(\pi_j))/\tau)}{\sum_k \exp((\delta_k + \log(\pi_k))/\tau)}$
  end
  return $\sum_j z_{ij} w_{ij}$
else
  foreach *child* n_{ij} *of* n_i **do**
    $z_{ij} = \text{Sample}(n_{ij}, \tau, \mathbf{x})$
  end
  $\mathbf{z}_i \leftarrow$ concatenate $z_{i0}, \ldots, z_{ij}$
  return $\mathbf{z}_i$
end

distributions is proposed by Jang et al. [98] as

$$w_j = \frac{\exp((\delta_j + \log(\pi_j))/\tau)}{\sum_k \exp((\delta_k + \log(\pi_k))/\tau)}, \quad \text{for } j = 1, \ldots, k,$$

where π is parameters of the categorical distribution, δ is independent noise with Gumbel distribution, i.e. $\delta \sim \text{Gumbel}(0, 1)$, and τ is the softmax temperature. This trick refactors the softmax distribution with a deterministic function and δ. In addition, the hard selection of one category is replaced with a relaxed version of the categorical distribution by interpolating between all components based on the sampled weights w_j, so that differentiability is still guaranteed. The complete differentiable sampling routine is summarized in Algorithm 7.

To investigate the effectiveness of CSPNs in building VAEs, we trained SPVAEs using

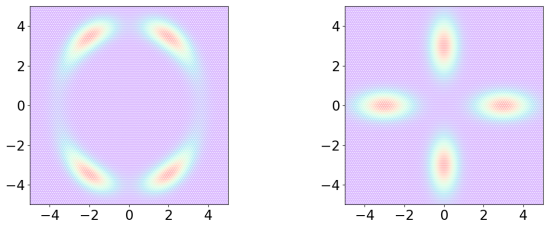


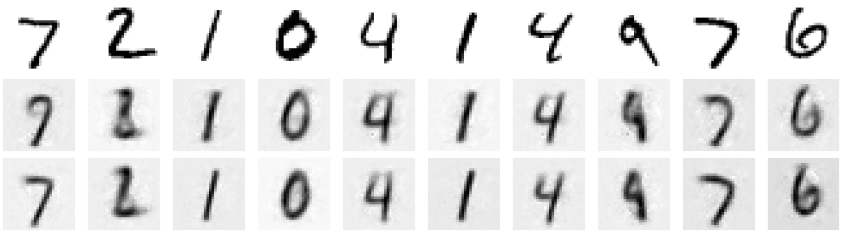
Figure 6.5.: The non-factorial prior distributions used in our experiments.

neural-CSPNs of different depths as encoders while keeping their input neural network fixed and evaluated both quantitatively and qualitatively. Quantitatively, we compared their ELBO, KL divergence and likelihood with each other (ELBO = likelihood - KL divergence). Qualitatively, we visually inspected the reconstructions of vanilla VAEs and SPVAEs. We used a standard CNN to extract parameters for SPNs, and for each dataset, we fixed this network up to the last two layers. We fixed the decoder as a naive CSPN, i.e. CSPN with only one layer[2]. We experimented on MNIST, the Fashion and the CelebA dataset. To test the benefit of rich representation of the posterior distribution, we used two different non-factorial complex priors, as shown in Figure 6.5. We trained a SPVAE on MNIST with ten latent dimensions using the isotropic multivariate Gaussian prior, and we trained another SPVAE with two latent dimensions using the non-factorial priors in Figure 6.5. A SPVAE for the CelebA dataset was trained with 128 latent variables using only the isotropic multivariate Gaussian prior.

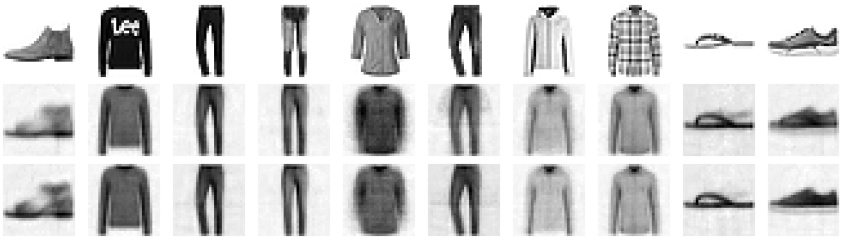
Figure 6.6a presents some randomly chosen held-out images of MNIST (top row), corresponding reconstructed images using a vanilla VAE (second row) and a SPVAE (bottom row). One can see that both models yield very plausible and attractive reconstructions. The reconstructions from the SPVAE are slightly sharper and more closely resemble the input images. The same visual effect can also be seen on the Fashion and the CelebA datasets, see Figures 6.6b and 6.6c. This gives further evidence that using CSPNs for VAEs yields very expressive generative models. Figure 6.7a[3] presents the results for SPVAE using CSPNs with respectively 1, 2, 4 and 6 layers on MNIST. The prior distribution here is the ten-dimensional isotropic Gaussian. Figure 6.7b and 6.7c present the results on the two non-factorial priors. When the latent variables are only two dimensional, the CSPN encoder can have at most 2 layers without any redundancy, so in this case we increased the capacity of the two-layered CSPN with more mixtures. Both figures show

[2]Note that a VAE with naive CSPNs as both encoder and decoder makes the mean field assumption on the posterior distribution and boils down to the conventional VAE proposed by Kingma et al. [108].

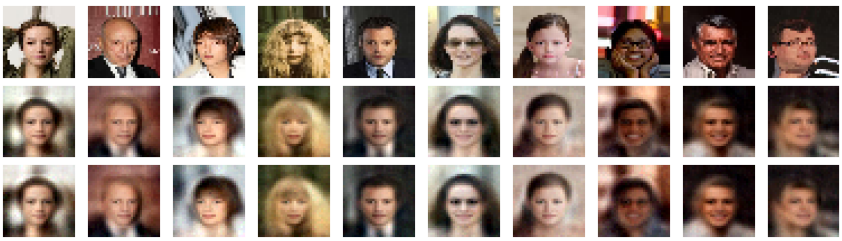
[3]Dashed lines highlight the value of a vanilla VAE.



(a) On MNIST.



(b) On Fashion.

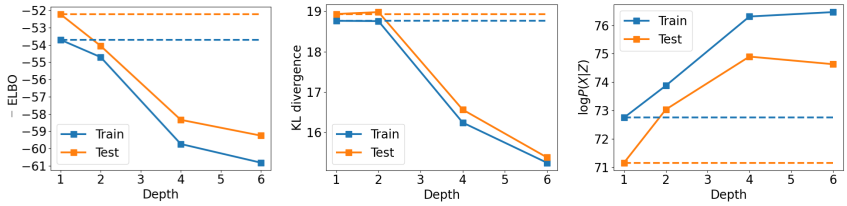


(c) On CelebA.

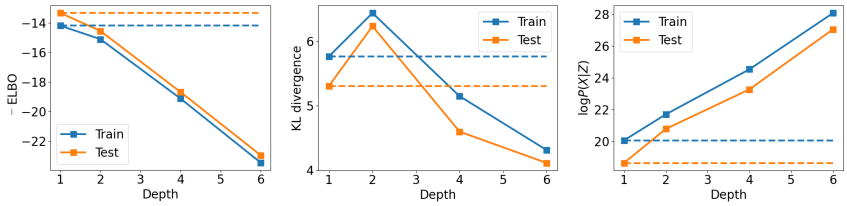
Figure 6.6.: The original test images (top row) and reconstruction images using vanilla VAEs (second row), and SPVAEs (bottom row).

that neural-CSPNs are effective at providing a tighter lower bound, specifically both KL divergence and likelihood improved. The same conclusion can also be made on the Fashion and the CelebA dataset, see Figure 6.7d and 6.7e.

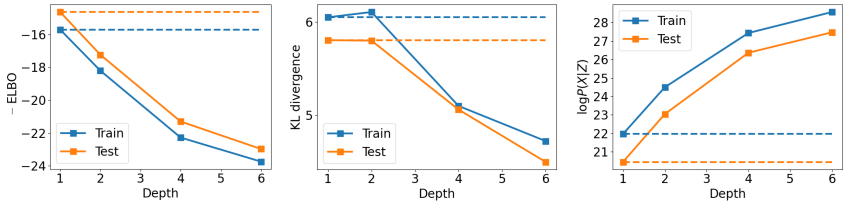
This concludes Chapter 6, which is devoted to imposing high-level structure for interpretability by using CSPNs as modular building blocks in generative models. Two deep generative models were presented here — autoregressive models chaining a number of CSPNs together, and VAEs using CSPNs for both the encoder and the decoder. Both models exhibit modular structures that are useful for interpreting the underlying domains. Empirical evidence across several datasets and domains confirmed the effectiveness of CSPNs in building generative models.



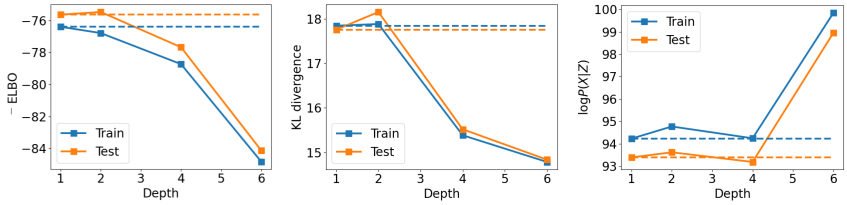
(a) Dataset: MNIST, the prior: isotropic multivariate Gaussian.



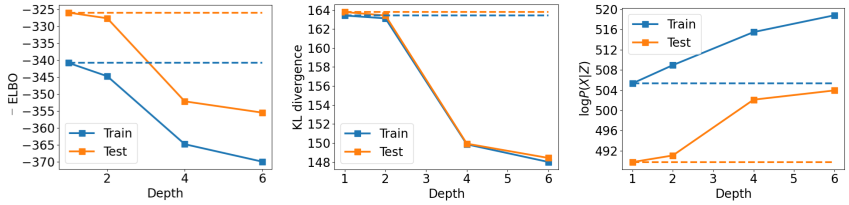
(b) On MNIST. Prior: a two-dimensional non-Gaussian distribution, see Figure 6.5 (left).



(c) On MNIST. Prior: a two-dimensional non-Gaussian distribution, see Figure 6.5 (right).



(d) On Fashion. Prior: isotropic multivariate Gaussian.



(e) On CelebA. Prior: isotropic multivariate Gaussian.

Figure 6.7.: The change of ELBO, KL divergence and likelihood when the encoder SPN increases its depth.

7. Explaining SPNs by Counterfactual Examples

In the previous chapter, we have demonstrated how to impose modular domain structure on deep probabilistic models using CSPNs as building blocks. This modularity allows for global-level interpretation of deep probabilistic models. In some cases, global interpretation is too general and we are interested in more specific or quantitative explanations for understanding an individual probabilistic inference. For example, instead of an interpretation such as "loan application is dependent on income, history of past payment, education level etc.", we want an explanation such as "this application is rejected just because the income is too low, whereas the rest factors are not critical". Compared to deep neural networks where explaining individual predictions is an active area of research, as introduced in Section 2.1.1, there are surprisingly few reports dealing with explaining individual probabilistic inferences.

Recall from Section 4.5, the deep learning approach for SPNs takes a RAT-SPN and uses GPU-based optimization for learning the parameters from the data. Since SPNs can be treated as connectionist models via RAT-SPNs, it is natural to ask whether the wide collection of explanation techniques for deep neural classifiers can be directly used or adapted for explaining probabilistic inferences in SPNs. In this section, we consider a commonly used inference type needed for prediction tasks: Most Probable Explanation (MPE) inference, i.e. $\mathbf{q}^* = \arg\max p(\mathbf{q}|\mathbf{e})$ for some evidence $\mathbf{e}$. The motivation is to interpret probabilistic inference in an analogous manner to interpreting neural classifiers, which usually takes the form of saliency maps for highlighting the most important features in vision tasks. However, these approaches are prone to yield noisy saliency maps that are not always interpretable to humans. Therefore, we consider a more human-friendly form of local explanations — counterfactuals.

Given an arbitrary data example from a domain as a reference point, a counterfactual is an contrastive example that yields a different outcome. It takes the form "If X had not occurred, Y would not have occurred". Counterfactuals are human-friendly explanations because they are contrastive to the current instance and human usually do not ask why a certain prediction was made, but why this prediction was made instead of another

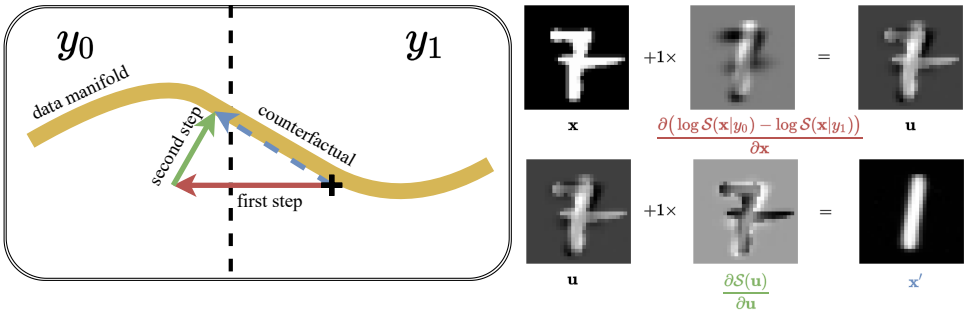


Figure 7.1.: Left: Illustration of our gradient-based approach. Arrow indicates perturbation based on a gradient step. Right: An example on MNIST.

prediction [126]. Besides, they are selective, meaning that they usually focus on a small number of feature changes [151]. Counterfactual examples are only helpful as explanations if they correspond to "possible worlds" [247], meaning that they should be likely under the given distribution. This is justified by the confirmation bias exhibited by humans: We tend to ignore information that is inconsistent with our prior beliefs [160]. This constraint is framed as different proxy metrics in the counterfactual literature [247, 156, 46, 242, 100, 166, 50]. However, the existing approaches either do not have an explicit density estimator or their density estimator is highly intractable. The direct consequence is that these approaches do not guarantee very likely counterfactuals.

RAT-SPNs, however, are not only density estimators but also allow for a range of tractable probabilistic inference routines. Therefore, they can be leveraged to pose likelihood constraints for counterfactuals. In the following, we will present a novel approach named *Sum-Product networks Interpretation by Counterfactual Examples* (SPICE), which generates counterfactual examples for RAT-SPNs using their tractable inference. Empirical study shows that this approach does not only yield very likely counterfactuals, but it is also an order-of-magnitude faster than the existing approaches. Despite being so fast, this approach is no less effective.

This chapter is organized as follows. First, we review related work in the counterfactual literature in Section 7.1. Then, we present our approach and give an intuitive understanding in Section 7.2. In the end, we show the effectiveness of our approach based on empirical evidence in Section 7.3.

7.1. Related Work

Research attention on counterfactual explanations has increased since Wachter et al. [247] presented the concept of unconditional counterfactual explanations as a novel type of explanation of automated decisions. Due to the resemblance between counterfactuals and adversarial examples, Wachter et al. frame the counterfactual generation as an optimization problem, akin to some approaches in the adversarial perturbation literature. Given a classification model, the objective function for counterfactual generation is optimized to reduce the loss for a counterfactual class w.r.t. the classifier's input. In addition, the objective function is constrained by the distance between counterfactuals and the query input to induce "close possible worlds". Standard gradient-based techniques are used to solve this optimization problem. This deep learning fashion methodology has been widely adopted in recent literature with modified objective functions.

Following Wachter et al. [247], Mothilal et al. [156] augmented the loss with a diversity constraint to encourage diverse solutions. However, these approaches do not have explicit knowledge of the underlying density or data manifold, which may lead to unrealistic counterfactuals. A bunch of methods counteract this issue by learning an auxiliary generative model to impose additional density constraints on the optimization process. As common choices for density approximators, VAEs [108] and their variants [110, 97] are used. For instance, Dhurandhar et al. [46] proposed Contrastive Explanations Method (CEM) for neural networks based on optimization. The objective function consists of a hinge-like loss function and the elastic net regularizer as well as an auxiliary VAE to evaluate the proximity to the data manifold. Poyiadzi et al. [175] proposed Feasible and Actionable Counterfactual Explanations (FACE), a graph-based algorithm to generate counterfactuals that are coherent with the underlying data distribution by constructing a graph over all the candidate targets. Besides, several domain-specific approaches are also emerging [162, 73, 26].

Related to counterfactuals, Ustun et al. [242] defined the term *recourse* as the ability of a person to change the decision of a model by altering *actionable* input variables. Joshi et al. [100] provided an algorithm called REVISE to suggest a recourse based on samples from the latent space of a VAE characterizing the data distribution. Pawelczyk et al. [166] developed a framework, named Counterfactual Conditional Heterogeneous Autoencoder (C-CHVAE), to generate faithful counterfactuals. C-CHVAE trains a VAE and returns the closest counterfactual due to a nearest neighbour style search in the latent space. Downs et al. [50] proposed another algorithmic recourse generation method, Counterfactual Recourse Using Disentangled Subspaces (CRUDS), that generates multiple recourses satisfying underlying structure of the data as well as end-user specified constraints. Based on a VAE-variant, CRUDS uses a Conditional Subspace Variational Autoencoder (CSVAE)

model [110] that is capable of extracting latent features that are relevant for prediction. Another method called Counterfactual Latent Uncertainty Explanations (CLUE) [6] is proposed for interpreting uncertainty estimates from differentiable probabilistic models using counterfactual explanations, by searching in the latent space of a Variational Autoencoder with arbitrary conditioning (VAEAC) [97].

7.2. Sum-Product networks Interpretation by Counterfactual Examples

The goal here is to automatically generate counterfactual examples as explanations for tractable probabilistic models. In particular, we are interested in explaining MPE inference in RAT-SPNs for prediction tasks. We propose a novel approach named SPICE to solve this task.

Consider a classification problem $f : \mathbb{R}^d \to \{1, \ldots, C\}$ with C labels. Assume a RAT-SPN $\mathcal{S}$ over $\mathbf{X}$ for $p(\mathbf{X})$. To use this RAT-SPN as a generative classifier, the network structure in Figure 4.5 is used. That is, C roots are used to represent class-conditional densities $\mathcal{S}_c(\mathbf{X}) \coloneqq \mathcal{S}(\mathbf{X}|Y = y_c)$. The overall density distribution is then given by

$$\mathcal{S}(\mathbf{X}) = \sum_Y \mathcal{S}(\mathbf{X}|Y = y_c)p(Y = y_c).$$

Bayes' rule is used to classify a sample $\mathbf{x}$:

$$y = \arg\max_Y \mathcal{S}(Y|\mathbf{x}) = \arg\max_Y \frac{\mathcal{S}(\mathbf{x}|Y)p(Y)}{\mathcal{S}(\mathbf{x})}.$$

Note that a RAT-SPN of this special structure has a dual use: It is both a density estimator $\mathcal{S}(\mathbf{X})$ and a classifier $\mathcal{S}(Y|\mathbf{X})$. We make use of both the generative and the discriminative aspect of the RAT-SPNs in the following.

Given a query example $\mathbf{x}$ with prediction y, the task is to generate its counterfactual example $\mathbf{x}'$ with prediction y'. Generating a counterfactual example amounts to generating an in-distribution example that is predicted as a counterfactual class. This is intrinsically a generative task. Therefore, it is beneficial to have a generative model of the data distribution. The generative model used primarily in the literature is VAEs [108]. In contrast, we use SPNs instead of VAEs. The benefits are twofold: (a) A set of probabilistic inference tasks are tractable, including data likelihood. (b) SPNs retain fully probabilistic semantics [170, 244]. Note that SPNs, when trained generatively as density estimators, can be understood as learning representations at different levels of abstraction in the

inner nodes [244]. Therefore, SPICE can also be viewed as moving in the latent space of the C root nodes by perturbing the input. As opposed to searching in the latent space of VAEs [166], no decoding from the latent space to the input space is necessary for SPICE. Besides, the latent representations are more interpretable due to probabilistic semantics, i.e. the c-th root node represents the c-th mixture component $\mathcal{S}(\mathbf{X}|Y = y_c)$. Vergari et al. [244] empirically showed that SPN embeddings can create a meaningful geometric space where same-class samples are reachable by proximity. This is correspondent to the natural clustering assumption where different values of categorical variables such as object classes are associated with separate manifolds [12]. This assumption is not only widely made for current machine learning algorithms, but also underpins many counterfactual explanation methods.

SPICE is defined as two serial perturbations: In the first step, we maximize $\log \frac{\mathcal{S}(y'|\mathbf{x})}{\mathcal{S}(y|\mathbf{x})}$ to induce the desired outcome y', which is equivalent to maximizing $\log \frac{\mathcal{S}(\mathbf{x}|y')}{\mathcal{S}(\mathbf{x}|y)}$ since

$$\begin{aligned}\log \frac{\mathcal{S}(y'|\mathbf{x})}{\mathcal{S}(y|\mathbf{x})} &= \log \Big(\frac{\mathcal{S}(\mathbf{x}|y')p(y')}{\mathcal{S}(\mathbf{x})} \frac{\mathcal{S}(\mathbf{x})}{\mathcal{S}(\mathbf{x}|y)p(y)} \Big) \\ &= \log \frac{\mathcal{S}(\mathbf{x}|y')}{\mathcal{S}(\mathbf{x}|y)},\end{aligned}$$

when assuming a uniform class prior. Towards this end, we take a gradient step w.r.t. the input $\mathbf{x}$ towards the steepest ascent of the counterfactual outcome $\log \frac{\mathcal{S}(\mathbf{x}|y')}{\mathcal{S}(\mathbf{x}|y)}$. This way, computing the first-step perturbation reduces to computing the gradient of conditional probabilistic queries, which can be done in linear time in terms of the size of the RAT-SPN. This step results in perturbed example $\mathbf{u}$ where

$$\mathbf{u} = \mathbf{x} + \frac{\partial \big(\log \mathcal{S}(\mathbf{x}|y') - \log \mathcal{S}(\mathbf{x}|y) \big)}{\partial \mathbf{x}} * \epsilon_1. \tag{7.1}$$

This linear perturbation is analogous to Fast Gradient Sign Method (FGSM) for generating adversarial perturbations [71]. More generally, counterfactual examples bear some resemblance to adversarial examples in that the perturbed sample $\mathbf{u}$ is expected to change the prediction from y to y' with a very small perturbation. In the XAI literature, the gradient of the output w.r.t. the input is known as input gradient explanation, which is a vector normal to the model's decision boundary at $\mathbf{x}$ and thus it serves as a first-order description of the model's behavior near $\mathbf{x}$ [198]. Consequently, taking a gradient step in Equation 7.1 is to cross the decision boundary with the shortest path under first-order approximation. However, it is to note that first-order approximation becomes less accurate when extrapolating too far in a non-linear space. Therefore it is advisable to try out a

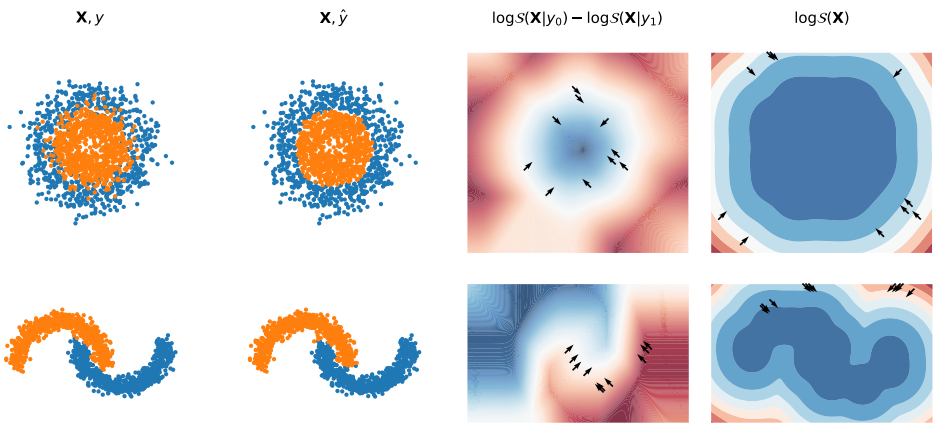


Figure 7.2.: Intuition of SPICE on 2D datasets.

small step size ϵ_1 for the perturbation first, and incrementally increase the step size if the perturbation did not cross the decision boundary. In case the manifold of the original class and the counterfactual class are adjacent in the latent space, a relatively small step size is enough to cross the decision boundary. Nevertheless, it is more tricky when the two classes are not adjacent in the latent space, and therefore a bigger step size is needed for crossing the decision boundary. The direct consequence is, that the gradient vector becomes a less accurate description of the decision boundary due to extrapolation, and it may possibly move the input to another class of manifold. A potential solution is to treat SPICE (including the second step introduced below) as one meta-step and iterate over many small SPICE-steps.

The perturbed sample $\mathbf{u}$ is intended to cross the decision boundary. Without additional constraints, this sample is very likely to deviate from the underlying data manifold. In order to generate in-distribution counterfactuals, we further maximize the density $p^*(\mathbf{u})$ of the current sample $\mathbf{u}$, which is approximated via the RAT-SPN $\mathcal{S}(\mathbf{u})$. To maximize the density $\mathcal{S}(\mathbf{u})$, we take another gradient step towards its steepest ascent, i.e.

$$\mathbf{x}' = \mathbf{u} + \frac{\partial \mathcal{S}(\mathbf{u})}{\partial \mathbf{u}} * \epsilon_2. \tag{7.2}$$

$\mathbf{x}'$ is the final output for the query example $\mathbf{x}$, which is intended to cross the decision boundary and have high likelihood at the same time.

See Figure 7.1 (left) for an illustration of SPICE. On the right is an example on MNIST. The first row corresponds to the first gradient step for perturbing the prediction irregardless

of the manifold. This perturbation removes some characteristics of the current class and adds some characteristics of the counterfactual class. The resulting sample $\mathbf{u}$ yields the desired class but obviously deviates from the training sets (It looks neither like a 1 nor 7). The second row corresponds to the second gradient step for generating an in-distribution counterfactual by pushing the intermediate sample $\mathbf{u}$ to a region with a higher density. See Algorithm 8 for a summary of SPICE.

To get a more intuitive understanding about the effect of each gradient step, we plot the two gradients in Equation 7.1 and 7.2 respectively on two commonly used 2D datasets in Figure 7.2. We trained a RAT-SPN $\mathcal{S}(\mathbf{X})$ on each dataset. From left to right, the figures are: training data for the classification task, RAT-SPN's predictions on this set, the contour lines of $\log \mathcal{S}(\mathbf{X}|y_0) - \log \mathcal{S}(\mathbf{X}|y_1)$ and the contour lines of RAT-SPN's density $\mathcal{S}(\mathbf{X})$[1]. The arrows depict gradient vectors on the underlying contour lines. The orange data points are class y_0 and the blue data points are y_1. One can see in the third column that the decision boundaries of RAT-SPNs are given by $\log \mathcal{S}(\mathbf{X}|y_0) - \log \mathcal{S}(\mathbf{X}|y_1)$. Therefore, taking the gradient of $\log \mathcal{S}(\mathbf{X}|y_0) - \log \mathcal{S}(\mathbf{X}|y_1)$ w.r.t. the input (the first step in Equation 7.1) induces a perturbation for crossing the decision boundary under first-order approximation around the query example[2]. In case the dataset is not dense in the input space, like the second dataset, the first gradient step may very likely extrapolate to a low-density region. Fortunately, the second gradient step of $\mathcal{S}(\mathbf{X})$ (Equation 7.2) serves as a first-order approximation of the local density and takes an example in low-density region to higher-density region, as illustrated in the last column[3].

7.3. Empirical Evaluation

To illustrate the advantages of SPICE, we designed experiments to evaluate it both qualitatively and quantitatively across several benchmark datasets. All the experiments were implemented in Python and Tensorflow, running on a Linux machine with two Intel Xeon processors with 56 hyper-threaded cores, 4 NVIDIA GeForce GTX 1080 under Ubuntu Linux 14.04.

Datasets: We experimented with three widely cited datasets commonly used in the counterfactual explanation literature and one real-world dataset. On the **MNIST** [120] dataset, we evaluated several contrastive pairs of classes where counterfactual pertur-

[1] Note that each figure on a row is plotted under the same scale, and the gradients in separate figures do not have one-to-one correspondence.

[2] The gradients are computed on randomly chosen test examples of class y_1 (from the blue cluster in the training set).

[3] The gradients are computed on randomly chosen *out-of-distribution* examples.

Algorithm 8: SPICE

Input : A RAT-SPN $\mathcal{S}$ with C class-conditional densities, a query example $\mathbf{x}$ with prediction y and a counterfactual class y'

Output : A counterfactual example $\mathbf{x}'$

Obtain the gradient vector normal to $\mathcal{S}$'s classification boundary at $\mathbf{x}$: $\mathbf{v}_1 = \frac{\partial\left(\log \mathcal{S}(\mathbf{x}|y') - \log \mathcal{S}(\mathbf{x}|y)\right)}{\partial \mathbf{x}}$

Perturb $\mathbf{x}$ to induce the counterfactual class y' (as in Equation 7.1): $\mathbf{u} = \mathbf{x} + \mathbf{v}_1 * \epsilon_1$

Obtain the gradient vector towards high-density region: $\mathbf{v}_2 = \frac{\partial \mathcal{S}(\mathbf{u})}{\partial \mathbf{u}}$

Perturb $\mathbf{u}$ to increase density: $\mathbf{x}' = \mathbf{u} + \mathbf{v}_2 * \epsilon_2$

return $\mathbf{x}'$

bations are intuitive to comprehend: digit 1 and 4, digit 1 and 7, digit 3 and 8, and digit 7 and 4. The **German credit dataset** [65] classifies people described by a set of attributes as good or bad credit risks. The **Adult-Income** [194] records whether a person makes over 50K a year based on census data. The **Caltech-UCSD Birds (CUB)** [249] is a real-world dataset for fine-grained bird classification. This dataset contains 200 bird species and we evaluated the counterfactuals across three contrastive pairs of bird species: Red Faced Cormorant and Crested Auklet, Myrtle Warbler and Olive sided Flycatcher, Horned Grebe and Eared Grebe. Bird species classification is a difficult problem that pushes the limits of the visual abilities for both humans and computers because some pairs of bird species are nearly visually indistinguishable and intraclass variance is very high [249]. This is arguably the most complex and high-dimensional problem studied so far in the counterfactual explanation literature. To make this problem slightly more approachable, we used feature representations extracted from the final convolutional layer of the VGG-16 [222] pretrained on ImageNet. That is, we trained a RAT-SPN as a generative classifier using the class-conditional feature maps as input.

Baselines: We considered three widely cited model-agnostic approaches in the literature that can be directly applied to our chosen datasets: Wachter et al. [247], CEM [46] and FACE [175]. Since the real-world dataset is quite high-dimensional compared to those commonly used in the literature, we discarded some baselines with scalability issue after experimenting with it, e.g. DiCE [156].

Implementation details: To evaluate SPICE with empirical evidence, we trained a RAT-SPN for each dataset. Each RAT-SPN is, as in Figure 4.5, a mixture of class-conditional densities. This RAT-SPN was used for both classification and density estimation. We used cross-validation to select hyperparameters for RAT-SPNs and for all the counterfactual

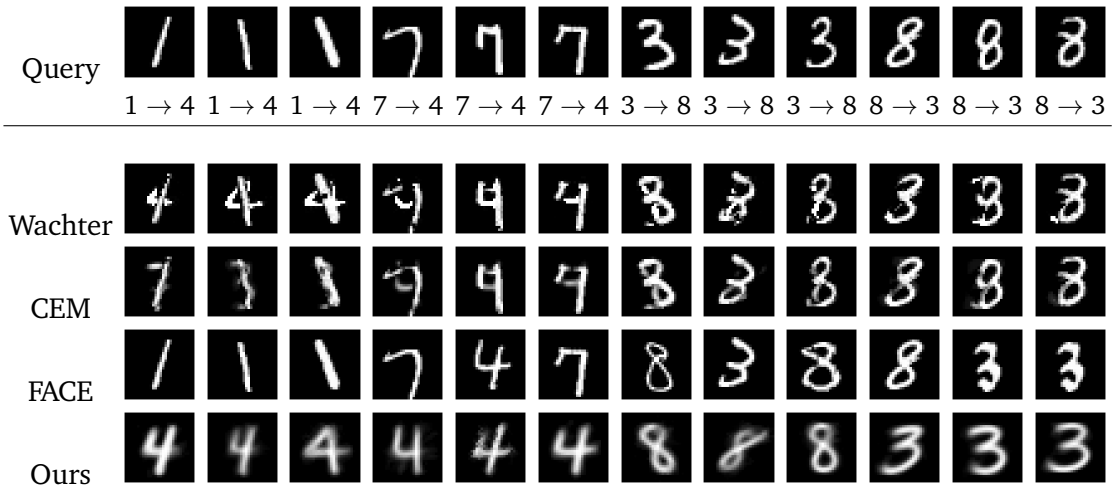


Figure 7.3.: Counterfactual examples on MNIST across several classes and methods.

methods. For the following experiments, the original SPICE method worked well enough, so the iterative process was not empirically investigated here, and this remains a topic for future investigations.

7.3.1. Visual Inspection of Counterfactual Examples

Counterfactual examples can be presented to users as contrastive explanations. The examples that correspond to prior beliefs of the users can be better received by them due to confirmation bias [160]. That means, the counterfactual examples should appear plausible and not deviate too far from the training samples. Figure 7.3 demonstrates some examples on MNIST for a variety of test cases across four methods. It is obvious to see that SPICE consistently yields the most visually appealing examples as they are smooth, clean and look very plausible. Although being smooth, our counterfactual examples still show a nice variation: For example, in the 7-th and 8-th column, the counterfactual 8 tilts to the left when the query object tilts slightly to the left, and the effect is analogous when the query object tilts to the right. In comparison, Wachter et al. [247] and CEM [46] both yield wiggly and noisy results. Although CEM includes a VAE reconstruction loss in its objective function as a proxy for constraining the examples to be in-distribution, explicit evaluation of density is still intractable. FACE [175] also yields very plausible examples because it simply returns a training instance of the counterfactual class.

Another visual example can be seen on the CUB dataset. As previously stated, we trained

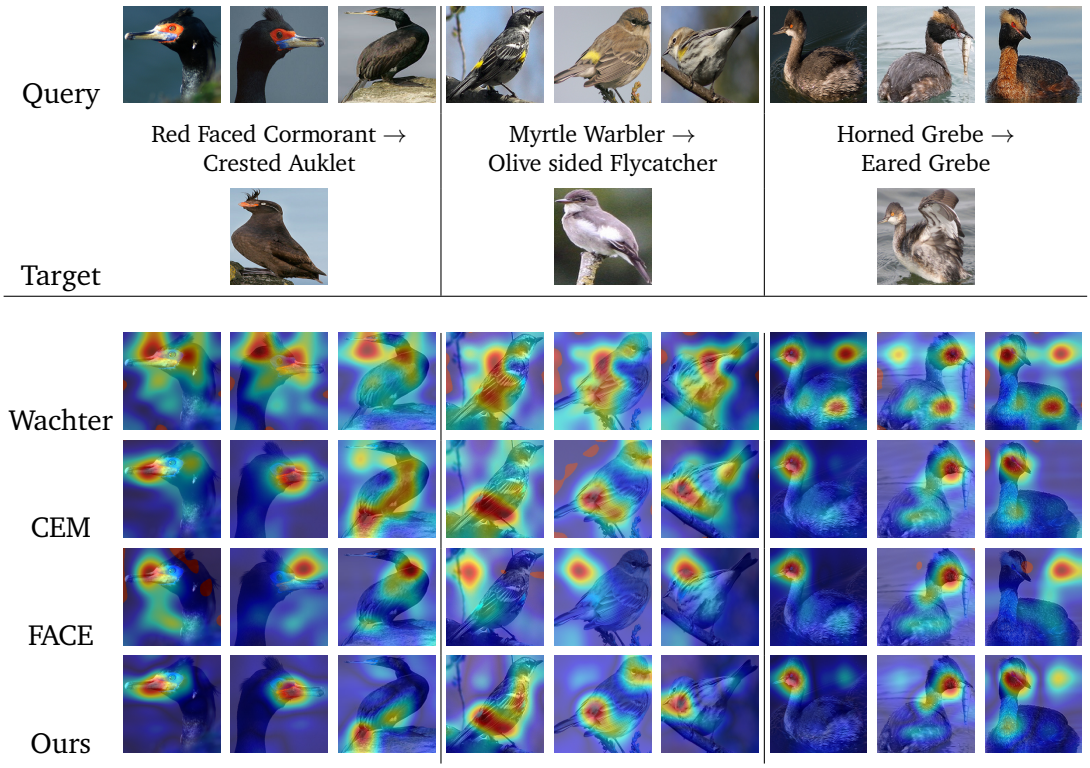


Figure 7.4.: Counterfactual examples on the CUB dataset across several classes and methods. The top row indicates the original class and the target class.

a RAT-SPN as a generative classifier using the class-conditional feature maps generated by the VGG-16. In this case, the RAT-SPN gives a density model on the feature maps and the counterfactual explanations are also computed in this transformed feature space. Since feature maps can not be trivially transformed back to raw features, we strived only for highlighting the salient counterfactual features instead of imputing them. Figure 7.4 demonstrates selected test examples on this dataset. The heatmap overlaid on the example indicates the salient counterfactual features that should be perturbed to become the counterfactual class. An example from each counterfactual class is shown in the first row. Note that this is not the particular target we strived for, but only an example for the readers to have an intuitive idea about the contrastive features. It can be seen that SPICE consistently yields plausible heatmaps: For the Red Faced Cormoran, the beak is highlighted when the image focuses on the head, and the tail is highlighted when the

image zooms out. For the Myrtle Warbler the salient features are mostly on the yellow spot of the feather. For the Horned Grebe the salient features are mostly highlighted around the head. Among the baselines, CEM and FACE are implicitly density-aware and often show consistent behavior with SPICE. However, FACE sometimes yields saliency on the background instead of on the bird. The method of Wachter et al. performs the worst and yields mostly random and unintuitive heatmaps. In conclusion, SPICE shows very competitive and intuitively plausible heatmaps that highlight the contrastive features on this complex dataset.

From visual examples on both datasets we can see that SPICE yields appealing and intuitive results that are easy to comprehend.

7.3.2. Density Evaluation of Counterfactual Examples

In contrast to wiggly examples, smooth examples often appear more plausible and realistic, and are in turn often tied with high likelihood. The goal of the following experiment is to offer a quantitative evaluation on likelihood.

It is widely agreed that out-of-distribution counterfactual examples have very little use in communicating explanations to humans. Wachter et al. [247] penalize distance from the counterfactual example to the query instance, which indirectly also prevents the counterfactual to deviate too much from the distribution. More recent approaches use VAEs as a neural density estimator to penalize out-of-distribution examples [46, 97, 110, 100, 166]. FACE [175] takes a different approach and searches through the training samples directly for a counterfactual example instead of constructing it via an optimization process.

However, these approaches are not able to constrain density explicitly and directly because density evaluation is simply intractable. The direct consequence is that the proxy constraint does not consistently yield counterfactual examples in the high-density region. To confirm this with empirical evidence, we took the advantage of RAT-SPNs on tractable inference to efficiently estimate and evaluate density.

Table 7.1 summarizes the average density evaluation on all the counterfactual examples across four datasets using four candidate approaches. On three of these datasets, SPICE yields the best density among the baselines. FACE is a strong competitor in terms of likelihood because it always returns a training instance. But its limitation is obvious: It assumes we have access to the training samples, which is oftentimes not the case, e.g. due to privacy reasons.

To have some intuition on tabular data, see Table 7.2 for randomly selected examples on the German credit dataset. The columns from A10-1 to A15-3 are all one-hot encoded

Table 7.1.: Density evaluation in log scale averaged on the test set (the higher, the better). Best result indicated using ●, runner-ups ○.

	Wachter	CEM	FACE	Ours
MNIST	-738.51	-735.13	-733.73○	-725.13●
Credit	-50.61	-43.55○	-43.83	-41.67●
Adult	-114.20	-96.99	-96.26●	-96.42○
CUB	-4593.85	-4505.00	-4501.99○	-4501.23●

categorical attributes[4]. For example, A10-1, A10-2 and A10-3 encode feature 10. Due to space constraint, only mutable features are shown. Each feature is perfectly negatively correlated with other features from the same categorical attribute due to the constraint imposed by one-hot encoding. Therefore we expect counterfactual perturbation to obey this feature correlation in order to stay close to the underlying distribution. That means, if one feature has a positive perturbation, the rest features from the same attribute should have a negative perturbation. That means, their perturbations should sum up to almost zero. Ideally each perturbation should be -1 or +1, but we worked with continuous values in practice so this is often not the case. From Table 7.2 one can see that SPICE and FACE always respect this relation — the perturbations within each attribute always sum up to zero or nearly zero, showing a negative correlation. It is no surprise that FACE always perfectly reflects this constraint because its counterfactual examples directly come from the training set, which perfectly satisfy this constraint by construction. In contrast, the method of Wachter et al. [247] and CEM often violate this constraint: Take the former method as an example, its perturbation on the first query yields 1, 1, -1 (sum up to 1) on attribute 14. These perturbations are obviously not balanced. This intuitive example is related to their likelihood evaluation (see Table 7.1): Those who respect the constraint better tend to yield higher likelihood because they are more realistic.

In conclusion, our counterfactual examples have dominant advantages on staying close to the high-density region.

[4]Attribute 10: Other debtors / guarantors — A10-1: none, A10-2: co-applicant, A103: guarantor. Attribute 14: Other installment plans — A14-1: bank, A142: stores, A14-3: none. Attribute 15: Housing — A151: rent, A15-2: own, A15-3: for free.

Table 7.2.: Counterfactual perturbations for the German credit dataset. Attributes are encoded as a one-hot numeric array. A cell color "green" denotes a negative correlation that we expect from the counterfactual perturbation. A cell color "red" denotes a wrong correlation.

	Attribute 10			Attribute 14			Attribute 15			
	A10-1	A10-2	A10-3	A14-1	A14-2	A14-3	A15-1	A15-2	A15-3	$\hat{y}$
query	0.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00	0.00	0
Wachter	0.00	1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	1.00	1
CEM	0.03	0.05	-0.09	0.08	0.08	-0.08	0.07	-0.29	0.08	1
FACE	1.00	0.00	-1.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Ours	0.90	0.05	-0.95	0.18	0.05	-0.23	0.22	-0.39	0.16	1
query	1.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0
Wachter	0.00	1.00	0.00	1.00	1.00	-1.00	0.00	0.00	1.00	1
CEM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
FACE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
Ours	-0.10	0.05	0.05	0.18	0.05	-0.23	-0.78	0.61	0.17	1
query	1.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00	0
Wachter	0.00	1.00	0.00	1.00	1.00	-1.00	1.00	-1.00	1.00	1
CEM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.21	0.00	0
FACE	0.00	0.00	0.00	0.00	0.00	0.00	1.00	-1.00	0.00	1
Ours	-0.10	0.05	0.05	0.18	0.048	-0.23	0.22	-0.39	0.17	1

7.3.3. Computation Time

The baseline methods employ a methodology that's widely represented in the literature: Defining a complex objective function with additional constraints and using optimization techniques to iteratively find a solution. This is usually too slow to yield counterfactual examples on the fly. Especially when users want to interact with machine explanations, fast computation becomes more essential. As empirical evidence, Table 7.3 shows that our method is an order-of-magnitude faster than the baseline approaches. This is no surprise due to the fact that SPICE takes only two gradient steps while the baselines can easily take up to thousands of iterations.

7.3.4. Effectiveness

Generating a counterfactual explanation amounts to finding a contrastive example that changes the class prediction. An approach is effective if it has a high success rate in perturbing the class prediction to the counterfactual class. This measurement is widely reported in the literature.

Table 7.3.: Average computation time on the test sets in seconds (the lower, the better). Best results shown in bold.

	Wachter	CEM	FACE	Ours
MNIST	181	216	281	**27**
Credit	59	97	757	**21**
Adult	52	57	1722	**10**
CUB	65	468	63	**16**

Table 7.4.: Success rates on the test sets (the higher, the better).

	Wachter	CEM	FACE	Ours
MNIST	1.00●	0.90○	0.54	0.71
Credit	1.00●	0.87	0.92	1.00●
Adult	1.00●	0.93	0.50	0.99○
CUB	1.00●	0.73	0.90	0.98○

Except for CEM, all the success rates are measured by the ratio between examples with the counterfactual prediction and all the test examples. Since CEM encourages a contrastive example belonging to any other class than the original class, measuring its success only by a specific class prediction is not fair for CEM. Therefore we measure its success rate by the ratio between examples with perturbed class prediction and all the test examples. This metric ranges between 0 and 1 where 1 is the best and 0 is the worst.

Table 7.4 gives a summary of success rates. As the results show, the approach by Wachter et al. is very effective at perturbing the class prediction, with a success rate of 1.0 across various datasets. CEM and FACE are less effective. SPICE has slightly lower success rates than the method of Wachter et al., but it is still very effective and reasonable in general. This result is not surprising: The approach by Wachter et al. has the fewest constraint in its objective function, while CEM and SPICE face a trade-off between prediction success and density constraint.

In conclusion, despite being very fast to compute, SPICE does not sacrifice the effectiveness of perturbing the class prediction.

This concludes Chapter 7, which is motivated by the desire for local explanations. We reviewed and discussed the counterfactual literature. Then we proposed our approach to improve upon the current counterfactual methods using RAT-SPNs. Finally, we gave

empirical evidence to demonstrate the effectiveness and advantages of our approach.

8. Right for the Right Features

In the previous chapter, we have presented our study on explaining individual inferences made by deep models. In this chapter, we investigate how to interactively improve deep models by observing and giving feedback on their explanations. User feedback on explanations gives an additional supervisory signal for training. This way, deep models do not only learn to make the right predictions, but also learn to align their explanations with user explanations.

This chapter is organized as follows. In Section 8.1 we give the motivation of this work and raise the research question we would like to answer. Then we present our approach to address this research question in Section 8.2. In the end, we demonstrate the effectiveness of our approach using empirical evidence in Section 8.3.

8.1. Motivation

Training deep neural networks for predictive tasks usually involves minimizing a loss function to encourage the predictions to align with the ground-truth labels. However, model parameters that result in a sufficiently low training loss may overfit to a solution that generalizes poorly to unseen data.

A wide variety of regularization techniques for deep learning seek to trade increased bias for reduced variance in order to reduce generalization error, possibly at the expense of increased training error [69]. A popular form of regularization is adding extra terms in the loss function as soft constraints on the parameter space [69], e.g. weight decay. Here, our goal is to use feedback on machine explanations to regularize the training process so that machine explanations align better with user explanations.

A typical scenario where deep models are prone to overfitting is when a confounding factor is present in the data, e.g due to bias in the data collection phase. That means, some irrelevant features correlate with the target labels serendipitously, therefore deep networks can achieve sufficiently low training error by simply learning these irrelevant features instead of the really salient features. The confounding factors often provide a shortcut solution that is easier to learn for deep neural networks. See Figure 8.1 for

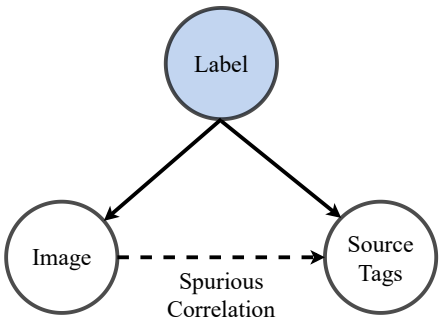


Figure 8.1.: Illustration of a confounding factor that by accident correlates with the label.

an illustration of this phenomenon. This phenomenon was first identified by behavioral researchers via a horse named Clever Hans that seems to demonstrate intelligence but in fact just responds to simple cues [173]. Later, the Clever Hans effect was also used to describe an analogous phenomenon identified in deep neural networks [117].

Our goal is to incorporate domain-specific regularization by penalizing the model from learning particular features. One may argue that this problem can be simply addressed by feature engineering prior to the training phase or by dataset augmentation. However, feature engineering usually aims to yield features that are generally useful for the underlying problem, while we strive for fine-grained control of features for particular decisions. As a regularization technique, dataset augmentation has shown its effectiveness particularly for classification problems [69]. This has also been used together with explanatory feedback from users to correct the model by randomizing the irrelevant features [237]. Yet, this approach incurs training overhead due to increased training examples. This problem becomes especially prominent and concerning when the number of features to penalize increases, in which case augmenting data via randomization may become tricky and costly. Moreover, current machine learning models usually rely on datasets with a careful analysis of generalization, dissecting the challenges in detection and classification [11].Take cow detectors as an example, the models show poor generalization ability to cows in new environments, i.e. at the beach, when trained on samples that do not capture enough variation of the environments [11]. In this case, taking pictures of cows in various environments, or manually generating such samples, is of course infeasible and costly. By designing a method to directly make use of user explanations to regularize the training process of a deep model, intensive data consumption and high sample complexity can potentially become less necessary since explanatory feedback provides a more compact representation of the samples than pure labels do.

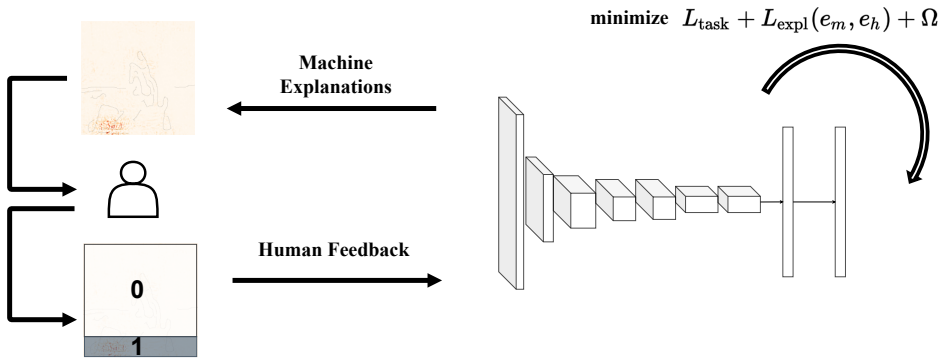


Figure 8.2.: Human explanations are given to correct machine explanations by adding a soft constraint on the parameter space. This will penalize the solutions that lead to mismatch between machine explanations and human explanations.

Subsequently, we are driven by the research question: How can we efficiently use explanations to regularize deep models so that they do not learn irrelevant features? This question will be addressed in the next section.

8.2. Right for Better Reasons

Built upon the work of Ross et al. [198], we extend RRR to Right for Better Reasons (RBR) using influence functions [111] instead of input gradients for machine explanations. Input gradients are a common explanation method that approximates the target model locally with a simple model for humans to understand how sensitive a prediction is to each feature. This type of explanation method assumes the underlying model is fixed. In contrast, influence functions explain further where the model comes from by tracing the predictions through its learning algorithm and back to the training data [111]. This robust statistic yields robust explanations that will be shown to be very effective supervisory signals for improving the model.

Specifically, we use influence functions to formulate a soft constraint on the parameters of the underlying model by penalizing influence functions for particular features according to user feedback. We augment the cross-entropy loss for classification tasks with an

Algorithm 9: RBR($\mathcal{L}, \mathcal{U}, T$)

Input : N labeled examples $\mathcal{L}$ of D dimensions, a set of unlabeled instances $\mathcal{U}$, and iteration budge T

Output : A function $\hat{f}$.

$\boldsymbol{A} \leftarrow \{0\}^{N \times D}$
$\mathcal{L} \leftarrow$ Concatenate $\mathcal{L}$ with $\boldsymbol{A}$
$\hat{f} \leftarrow$ `FitRBR(`$\mathcal{L}$`)`
repeat
  $\mathbf{x} \leftarrow$ `SelectQuery(`$\hat{f}, \mathcal{U}$`)`
  $\hat{\mathbf{y}} \leftarrow \hat{f}(\mathbf{x})$
  $\hat{\mathbf{e}} \leftarrow$ `ExplainRBR(`$\hat{f}, \mathbf{x}, \hat{\mathbf{y}}$`)`
  Present $\mathbf{x}, \hat{\mathbf{y}}$ and $\hat{\mathbf{e}}$ to the user
  Obtain $\mathbf{y}$ and explanatory feedback $\mathbf{a}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{x}, \mathbf{y}, \mathbf{a})\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\mathbf{x}\}$
  $\hat{f} \leftarrow$`FitRBR(`$\mathcal{L}$`)`
until *budget T is exhausted or $\hat{f}$ is good enough*;
return $\hat{f}$

additional penalty term for explanations, which gives the RBR loss

$$
\begin{aligned}
L_{\text{RBR}} = & \underbrace{\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} -y_{nk} \log(\hat{y}_{nk})}_{\text{right answers}} \\
& + \underbrace{\lambda_1 \sum_{n=1}^{N} \sum_{d=1}^{D} (A_{nd} e_{\text{IF}_{nd}}^{\text{T}} e_{\text{IG}_{nd}})^2}_{\text{right reasons}} \\
& + \underbrace{\lambda_2 \sum_{i} \theta_i^2}_{\text{L2 regularization}} ,
\end{aligned}
\tag{8.1}
$$

where $\boldsymbol{A} \in \{-1, 0, 1\}^{N \times D}$ encodes user feedback by annotating irrelevant features as 1s, salient features as -1s and the rest as 0s. λ_1 and λ_2 are hyperparameters that control the balance between the three terms. Empirical evidence suggests that good choices for λ_1

usually are those that bring the right answer loss and the right reason loss to be on the same order of magnitude. λ_2 controls the strength of L2 regularization. Let $\mathbf{e}_{\text{IF}}^{\text{T}}\mathbf{e}_{\text{IG}}$ be the machine explanations where $\mathbf{e}_{\text{IF}}$ and $\mathbf{e}_{\text{IG}}$ denote respectively approximated influence functions and input gradients, i.e.

$$\mathbf{e}_{\text{IF}}^{\text{T}} \coloneqq \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}}), \qquad \text{and,} \qquad \mathbf{e}_{\text{IG}} \coloneqq \bigtriangledown_{\mathbf{x}} \hat{\mathbf{y}}.$$

$\mathbf{e}_{\text{IF}}^{\text{T}}$ is an approximation of the fine-grained influence function

$$-\nabla_{\boldsymbol{\theta}} L(\mathbf{z}_{\text{test}}, \hat{\boldsymbol{\theta}})^{\text{T}} \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}})$$

leaving out $\nabla_{\boldsymbol{\theta}} L(\mathbf{z}_{\text{test}}, \hat{\boldsymbol{\theta}})$ which can not be calculated because the model does not have access to test samples $\mathbf{z}_{\text{test}}$ at training time. This gives us an efficient approximation of the effect of a training point perturbation $\mathbf{z} \rightarrow \mathbf{z} + \boldsymbol{\delta}$ on the parameters of the model, which in turn gives us the feature importance for $\mathbf{z}$. In addition, we approximate the Hessian matrix with an identity matrix to reduce the training cost and stabilize the training dynamics.

A special case of RBR uses no user feedback and sets $\boldsymbol{A}$ to a $N \times D$ all-ones matrix instead. Using this loss has the effect of regularizing all the features by their influence functions, which is a comparable approach to regularizing input gradients [197]. The goal of regularizing input gradients is to penalize the sensitivity of the KL divergence between the predictions and the labels. In other words, if any input changes slightly, the predictions will not change significantly [197]. Ross et al. [197] show that regularizing input gradients provides an effective adversarial defense for deep neural networks. Similarly, regularizing influence functions penalizes the sensitivity of parameter gradients $\nabla_{\boldsymbol{\theta}} L(\mathbf{z}, \hat{\boldsymbol{\theta}})$ to small input perturbations. That means, this form of regularization encourages the parameters to converge to solutions where slight input perturbations affect the parameter gradients only slightly, which in turn influence the optimization outcome only slightly. As Koh et al. [111] mentioned, influence functions can be used to craft adversarial training images that are visually indistinguishable and can flip a model's prediction on a separate test image. Regularizing influence functions can improve the adversarial robustness to counteract this problem.

In the general case, RBR regularizes a subset of features rather than the entire set to encourage the classifier to learn from the remaining features. We instantiate the framework of XIL (refer to Section 2.2.2 for recap) for RBR using `ExplainRBR(·)` to compute the gradient-based machine explanations $\mathbf{e}_{\text{IF}}^{\text{T}}\mathbf{e}_{\text{IG}}$ and `FitRBR(·)` to learn the classifier using the RBR loss in Equation 8.1 incorporating user feedback. Refer to Algorithm 9 for details.

The RBR loss (Equation 8.1) is built on influence functions that give the feature importance on the input level. Since influence functions can flexibly adapt to various domains

and model architectures, so can RBR. However, feature attributions on the raw input level are usually quite noisy because the raw input may contain too many fine-grained details. Therefore Schramowski et al. also proposed an alternative approach to RBR using explanations on the transformed feature level rather than the raw input level. To this end, Schramowski et al. used Grad-CAM [206] for machine explanations because it explains up to the highest-level feature maps, i.e. the last convolutional layer $\mathbf{h}$, and hence it is less subject to noise in the input. This gives the loss

$$
\begin{aligned}
L = & \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K} -c_k y_{nk} \log(\hat{y}_{nk})}_{\text{right answers}} \\
& + \underbrace{\lambda_1 \sum_{n=1}^{N}\sum_{d=1}^{D} \left(A_{nd} \frac{\partial}{\partial h_{nd}} \sum_{k=1}^{K} c_k \log(\hat{y}_{nk}) \right)^2}_{\text{right reasons}} \\
& + \underbrace{\lambda_2 \sum_{i} \theta_i^2}_{\text{L2 regularization}} \quad ,
\end{aligned}
\tag{8.2}
$$

where c_k is a rescaling weight given to class k. This is especially useful for real-world and high-dimensional data, although its use is limited to CNNs only. For more details about this approach, we refer the readers to Schramowski et al. 2020 [203].

8.3. Empirical Evaluation

In this section, we present empirical studies that aim to answer the following research questions:

(Q1) Does RBR regularization help to improve adversarial robustness using no user feedback?

(Q2) Do classifiers learn the right rules when using RBR?

(Q3) Is RBR effective in high-dimensional domains?

In the following experiments, we simulate user interactions by running only one iteration of the outer training loop.

8.3.1. Dataset

Toy-Color Dataset. We used the toy-color dataset constructed by Ross et al. [198]. This dataset contains $5 \times 5 \times 3$ RGB images of four possible colors. The labels are determined by two independent decision rules:

1. Four corner pixels have all the same color.
2. Top-middle three pixels have all different colors.

Specifically, images that satisfy both rules belong to class 1 and images that satisfy neither rule belong to class 2. That means, only pixels on the corners and the top-middle three pixels are salient features, and faithful machine explanations that attribute high importance to the rest features reveal the deficiency of the underlying model. In fact, since the labels are constructed in the way that the two decision rules are strongly correlated, only one decision rule suffices to determine the label. These two rules can then be viewed as two distinct solutions to this classification problem.

Decoy MNIST. Based on the popular MNIST dataset [120], Ross et al. [198] constructed a decoyed version by adding confounding factors to the images. In the training set, the confounders are 4×4 gray patches in randomly chosen corners whose shades are determined by the corresponding label y using the function $255 - 25y$. In the test set, the gray patches have random shades. If the confounding factors successfully fool the model, we expect to observe a high generalization error and machine explanations that highlight the corner pixels.

PASCAL VOC 2007. Published for a challenge to recognize objects from a number of visual object classes in realistic scenes, PASCAL VOC 2007 [54] consists of twenty object classes. We used a subset of it by considering only the horse class and the dog class because only images of the horse contain obvious confounding factors — the watermarks on the bottom left corner. We rescaled all the images to $224 \times 224 \times 3$ to fit for the VGG-16 classifier [222].

Plant Phenotyping. This dataset is acquired from plant physiologists for a real-world and scientific task — plant phenotyping. It consists of RGB and hyperspectral images of leaf tissue from Cercospora beticola-inoculated and healthy sugar beet plants. Each tissue sample has images over five consecutive days with only slight differences. In this way, each tissue sample is represented five times in the dataset. In order to prevent training samples from leaking to the test set, the training or validation set always includes all days of a sample. It is also worth noting that the disease severity varies strongly across samples. Some inoculated tissue samples show a clear characteristic of Cercospora Leaf Spot (CLS) in RGB images while others are indistinguishable for human inspection. We rescaled all the images to $224 \times 224 \times 3$ to fit for the VGG-16 classifier [222].

8.3.2. Experiment Protocol

Experiments were implemented in Python and Tensorflow, ran on a Linux machine with two Intel Xeon processors with 56 hyper-threaded cores, 4 NVDIA GeForce GTX 1080 under Ubuntu Linux 14.04.

8.3.3. Adversarial Robustness

This experiment is to study how effective RBR regularization is against adversarial input perturbations without any user feedback.

Let $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ be an adversarial example based on perturbation $\boldsymbol{\delta}$. For classification problems with well-separated classes, the classifier is expected to yield the same prediction to $\mathbf{x}$ and $\tilde{\mathbf{x}}$ as long as $\|\boldsymbol{\delta}\|_\infty < \epsilon$, where ϵ is small enough to be discarded by the sensor or data storage apparatus associated with our problem [71]. *Fast Gradient Signed Method* (FGSM) [71] computes a max-norm constrained adversarial perturbation by approximating the loss function L around the current value of $\boldsymbol{\theta}$ with a linear model, that is, perturbation

$$\boldsymbol{\delta} = \epsilon \text{sign}(\nabla_{\mathbf{x}} L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})) \tag{8.3}$$

increases the local linear approximation of the loss function. This method was found to cause a wide range of models to yield wrong predictions of their perturbed input.

We adapted FGSM using influence functions, i.e.

$$\boldsymbol{\delta} = \epsilon \text{sign}(\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})). \tag{8.4}$$

This gives us the perturbation on $\mathbf{x}$ that increases the local approximation of the parameter change during training, which in turn influences the prediction of $\mathbf{x}_{\text{test}}$ post-training by $-\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{x}_{\text{test}}, \mathbf{y}_{\text{test}})^{\mathrm{T}} \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})$. We relaxed the calculation here again by approximating the Hessian matrix $\boldsymbol{H}$ with an identity matrix, resulting in a linear approximation of the local model around $\boldsymbol{\theta}$.

In order to defend against this adversarial perturbation, we regularized influence functions during training by setting the feedback matrix $\boldsymbol{A}$ in the RBR loss (Equation 8.1) with all-ones. Doing so would regularize the input gradients at the same time because the machine explanations in RBR are the element-wise product of both influence functions and input gradients. The intuition is that by regularizing influence functions, we encouraged the model to learn parameters that were relatively insensitive to small training input perturbations. By regularizing input gradients, we pushed the model into regions where the model was less sensitive to small test input perturbations when the model was fixed.

This experiment was done on the toy color dataset [198] and MNIST [120]. We have two baselines, a vanilla classifier trained without any form of regularization and a classifier

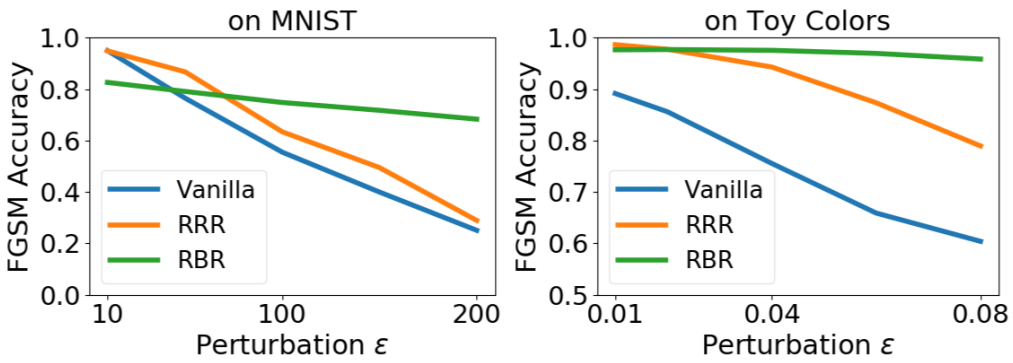


Figure 8.3.: Accuracies of the vanilla model, RRR and RBR on adversarial examples with increasing perturbations ε.

trained with RRR regularization [198] using no human feedback as well. For RBR and the baselines, we used the same model architecture, i.e. a multilayer perceptron with six hidden layers of size 1000, 500, 200, 100, 50 and 30 with softplus activations. The model was overparameterized for both tasks so that regularization can contribute positively to the training process. We used grid search to select the best hyperparameters λ_1 for both RRR and RBR approaches. Here, the hyperparameter λ_1 for RBR and RRR were respectively $1e+8$ and $1e+3$ on MNIST, $1e-9$ and $1e-4$ on the toy colors. We set $\lambda_2 = 1$ for all the experiments.

Figure 8.3 shows the accuracy of the vanilla model, RRR and RBR for a range of perturbation ϵ. One can see that the accuracy of RBR regularization drops only very slightly when variation on the input increases, while RRR regularization and the vanilla model deteriorate significantly. This is clear evidence that RBR regularization leads to a much more robust model in terms of FGSM-based adversarial perturbations on both datasets compared to the vanilla model and RRR-regularized model. This answers the first research question affirmatively.

8.3.4. RBR Learns the Right Rules

Unlike the previous experiment where no user feedback is used and all features were regularized equivalently with RBR, in this experiment, we used domain-specific feedback to regularize particular features. The goal is to investigate whether this regularization allows the model to maintain its predictive power by learning the right rules when the wrong features are penalized.

We used the toy color dataset and the decoyed MNIST for this experiment. As baselines,

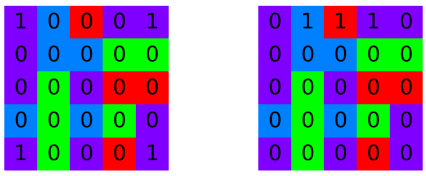


Figure 8.4.: Human feedback $\boldsymbol{A}$ for the toy color dataset to penalize four corner pixels (Left) and top-middle three pixels (Middle) respectively.

we used again RRR. On the toy color dataset, the model architecture we used was a multilayer perceptron with one hidden layer of size 30 with softplus activations and softmax output. Hyperparameter λ_1 for RBR and RRR were respectively set to 10 and 0 according to grid search results. $\lambda_2 = 1$ for both models.

As mentioned in Section 8.3.1, rule 1 (four corner pixels are all the same) or rule 2 (top-middle three pixels are all different) alone is sufficient for predicting the labels on the toy color dataset. Therefore, we regularized each rule at a time by giving feedback on the matrix $\boldsymbol{A}$. See Figure 8.4 for both feedback on one sample, the rest samples received the same feedback in this problem. The results are demonstrated on randomly chosen 25 samples in Figure 8.5. On the left, the top three middle pixels were penalized. Each sample on both the RRR model and the RBR model was explained by both input gradients and influence functions to make sure the machine explanations were not biased. On the right, the four corner pixels were penalized. The same 25 samples are plotted with their explanations. It is obvious to see that both models roughly learnt rule 1 when rule 2 was penalized, and vice versa. RBR learnt even slightly better rules than RRR, since the explanations of RRR on some samples either did not recognize the right rule or did not recognize the complete rule.

The same experiment is repeated on the decoyed MNIST [198]. We used a multilayer perceptron with two hidden layers of size 10 and 5 with softplus activations and softmax output. The feedback annotated every feature confounded with a grey patch as one. Figure 8.6 presents randomly selected 25 samples and their explanations on each model. Without any regularization, the vanilla model was heavily dependent on the features in the corner. RRR and RBR both corrected the model from this wrong behavior. RBR learnt slightly better rules than RRR because the machine explanations of RBR focused more on the digits than on the background.

This experiment demonstrates that RBR effectively regularizes the model, forcing the model to learn the right and even better rules. This answers the second research question affirmatively.

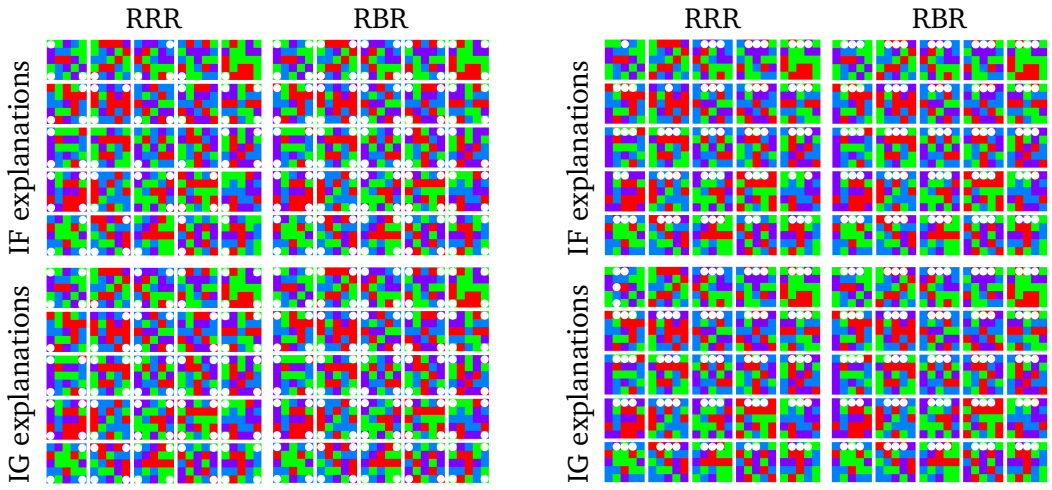


Figure 8.5.: IF and IG explanations after penalizing the top-middle three pixels (the left column) and the four corner pixels (the right column), respectively. White dots denote the salient features.

8.3.5. Effectiveness in High-Dimensional Domains

The previous experiment demonstrated the effectiveness of RBR on synthetic or toy datasets. In the following experiment, we aim to investigate how effective is RBR in a real-world setting with high-dimensional datasets using possibly noisy feedback. To this end, we used PASCAL VOC 2007 [54] and the plant phenotyping dataset. We used the VGG-16 [222] and replaced its output layer to predict two-way classifications on both datasets. ImageNet-pretrained weights were used to initialize the parameters, then we did fine-tuning for each task. Balanced accuracy score, defined as the average of recall obtained on each class, was used as a performance metric to account for class imbalance.

On PASCAL VOC 2007 we chose the category of horse because only this category contains obvious confounding factors, for the other category we chose dog. The vanilla model reached 99% training accuracy and 87% test accuracy after fine-tuning. If one does not ask for machine explanations, the vanilla model may seem quite good according to the accuracy. However, machine explanations across a broad range of samples, see Figure 8.8, imply that the model failed to distill the essence of the classification task by learning a pathological solution found by accident in the dataset. In particular, the horse images happen to contain watermarks on the bottom-left corner while the dog images do not. The model hence learnt to use the presence of the watermarks to make its prediction. This

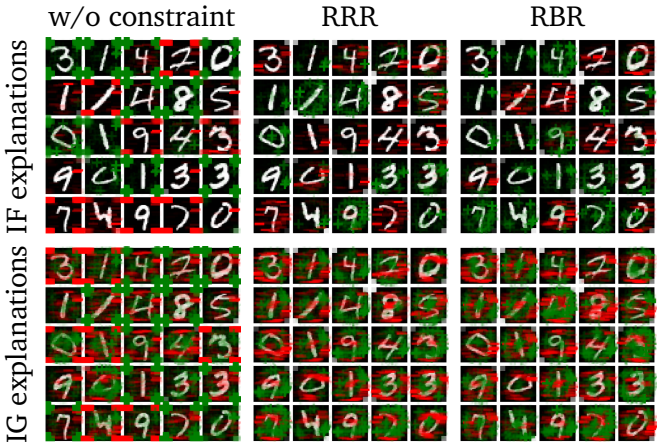


Figure 8.6.: IF and IG explanations on randomly selected samples from models trained without constraint, with RRR and RBR, respectively.

confirmed the findings made by Lapuschkin et al. [117], that DNNs are vulnerable to the Clever Hans behavior.

In order to correct the model from this behavior, we gave feedback for penalization on the bottom block of pixels that contain the watermarks and feedback for promotion on the rest pixels, see Figure 8.7 (top left) for an example. The dark overlay over the image indicates 1s in the feedback matrix, and the rest are -1s. In this case, the feedback is quite noisy because it gives only a coarse separation of the right and the wrong features: The annotation of the right features contains not only horse but also background and the annotation of the wrong features may contain a small fraction of the horse. The following experiment empirically shows that this noisy feedback still works quite well.

We fine-tuned a model based on RRR as the baseline and a model based on RBR with λ_1 being $1e+7$ and 1, respectively. λ_2 was set to 1 for both models, using the same feedback. Some samples were randomly chosen to present the machine explanations of the vanilla model, the RRR model and the RBR model using input gradients in Figure 8.8. One can see that the VGG-16 learnt to make its decisions based on the watermarks when no regularization was added. Both RRR and RBR were effective in regularizing the watermarks. However, although RRR did not use watermarks as salient features any more, it used other less interpretable features like the background rather than the horse itself. This may be explained by the fact that the model learnt confounding factors other than the watermarks, such as the grass, the bush or even the rider. In contrast, RBR yielded more interpretable features that made intuitive sense because the salient features overlapped

		Vanilla	RRR	RBR
PASCAL VOC	Counter Examples	71%	81%	84%
	Random Examples	74%	65%	76%
Deep Plant	Counter Examples	50%	54%	62%
	Random Examples	70%	54%	51%

Table 8.1.: Accuracies of three models on the counter examples and the random examples experimented on the PASCAL VOC 2007 dataset and the deep plant dataset, respectively.

more with the horse itself.

The same experiment was then repeated on a scientific task — plant phenotyping. We used the same model architecture, the VGG-16, the same baselines and the same accuracy metric. Again, the vanilla VGG-16 fine-tuned on this task performed quite well according to the accuracy score — 82% on the test set. Its machine explanations using input gradients reveals that the model used mainly the background rather than the leaf tissue to make its decisions. This is of course biologically not plausible. Annotations on the background were then acquired as feedback for both RRR and RBR, see Figure 8.7 (bottom left) for an example. RRR and RBR were trained with λ_1 being $1e-3$ and $1e-4$, respectively. Figure 8.9 shows some random examples with their explanations using input gradients. It is obvious to see that, without any additional regularization, the vanilla model learnt the wrong rules by focusing on the background. This may be explained by the fact that inoculated leaves may show some early signs of infection in their nutrition solution before a sick spot is visible on the leaf tissue. Both RRR and RBR prevented the model from learning the rules from the background. In particular, their explanations focus on the biologically plausible reasons, namely, on the leaf tissue. However, RBR learnt better rules than RRR since its explanations were better aligned with the sick spot on the leaf tissue, highlighted in yellow squares. In addition to plausibility, the explanations of RBR were also more simple and concise.

Following the qualitative experiments from above, we also present quantitative evidence. We measured the effectiveness of user feedback in learning the right rules based on the notion of counter examples [237], which are constructed by randomizing the irrelevant features (annotated as 1s in the feedback matrix) based on the original samples. Examples are given in the right column of Figure 8.7. The intuition is that the model should be able to maintain its prediction accuracy on the counter examples if it learns the right rule, because randomizing the irrelevant features does not affect the decision logic. In contrast

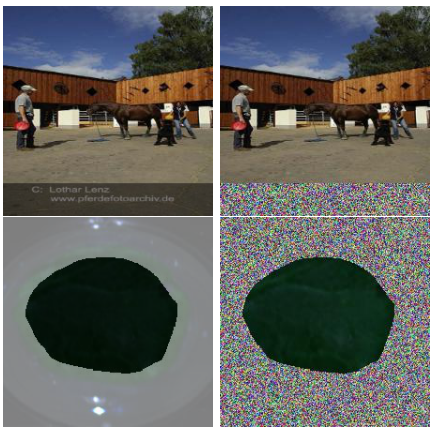


Figure 8.7.: User feedback (left) and the counter examples (right) on the PASCAL VOC 2007 dataset (top) and user feedback (left) and the counter examples (right) on the plant phenotyping dataset (bottom).

to the counter examples, we also defined random examples where only the salient features are randomized. A model that learns the right rule is expected to deteriorate significantly on the random examples.

The balanced accuracy score was measured on both the counter examples and the random examples for the previously trained baselines and the RBR model on both datasets. The results are summarized in Table 8.1. The vanilla model achieved higher accuracy in predicting the random examples than the counter examples on both datasets. This alludes to the fact that the model relies heavily on a set of features that are deemed irrelevant to the problem by humans. Compared to the vanilla model, RRR and RBR both showed higher accuracy on the counter examples and lower accuracy on the random examples. This means they learnt more right features after using explanatory feedback. When comparing RBR with RRR, RBR showed even higher accuracy on the counter examples. This is quantitative evidence that RBR learnt better rules than RRR.

It is easy to conclude from this series of experiments that RBR is also very effective at regularizing the wrong rules in high-dimensional domains. In this case, noisy feedback has also been proven to work sufficiently well. This experiment answers the last research question affirmatively.

This concludes Chapter 8, which highlighted the Clever Hans problem in DNNs and described an approach to counteract this problem. Empirical evidence across several datasets using different model architectures confirmed the effectiveness of our approach. We now turn to Chapter 9, in which we study the Clever Hans behavior further in generative

Figure 8.8.: Revising the VGG-16 on the PASCAL VOC 2007 dataset. Horse images (first row), their saliency maps on the vanilla model (second row), and their saliency maps on RRR (third row) and RBR (last row).

models.

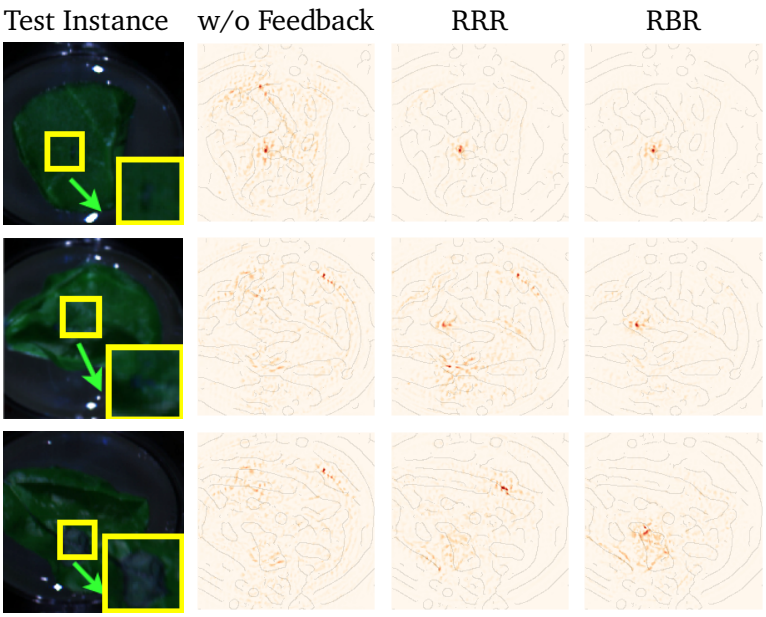


Figure 8.9.: Plant examples (zooming in on biologically plausible features) and their saliency maps before and after user feedback.

9. Right for the Right Latent Factors

In the previous chapter, the Clever Hans phenomenon was illustrated on DNNs. This phenomenon is induced by training a deep classifier on a confounded dataset. In this chapter, we extend our research attention to generative models. In particular, we investigate the effect of confounded datasets on deep generative models and discover the Clever Hans-like phenomenon that is directly linked to entangled latent representations. Based on this finding, we propose a technique to debug generative models from the Clever Hans-like phenomenon by allowing users to interact with the model via a small set of examples.

This chapter is organized as follows. First, we describe the problem that motivated our research and make the connection to entangled latent representations in Section 9.1. Then, we give a brief overview of related work on disentangled representation learning in Section 9.2. Afterwards, we discuss disentangled representations for generative models in Section 9.3, where we present our interactive approach to learning disentangled representations and analyze the properties of our approach. In the end, we evaluate our approach by empirical evidence in Section 9.4.

9.1. Motivation

Statistical machine learning methods are built on the i.i.d. assumption. But in practice this assumption can be easily violated unnoticed, e.g. due to a biased data-collecting process [239, 238, 192, 230, 77]. As a result, spurious artifacts that arise from the data-collecting process can be propagated to the model, which makes the model unable to generalize beyond the training set [202]. Recall the cow detector example [11] in Section 8.1. The cow images are for natural reasons mostly captured in specific environments with grass in the background. Deep classifiers learnt on this training set show poor performance in face of cows in new environments such as at the beach. The horse classifier on the PASCAL VOC 2007 dataset investigated in Section 8.3 is also strongly influenced by the specific context of the images, i.e. the watermarks. Both datasets violate the i.i.d. assumption by serendipitously including spurious artifacts, which in turn prevents the model from learning the true underlying distributions. Apart from the generalization problem, these

models may potentially also inflict unfair treatment on certain groups of people or reinforce the existing social stereotypes[1]. In Chapter 8, a solution was presented to identify and alleviate the aforementioned problem by visualising it and interacting with users for their feedback on machine explanations.

However, the solution in Chapter 8 is restricted to interacting with visual saliency maps. Here, we are motivated by the question: How can one interact with a latent representation that learns latent factors of the visual features? For instance, a facial dataset that exhibits selection bias by including faces of various skin tones disproportionately may leave a footprint of its bias in a model that learns from it. Igniting a lot of controversies, the super-resolution algorithm PULSE [145] is found to upsample a low-resolution input image of Barack Obama into a photo of a white male[2]. In this case, the explanations for skin tone can not be easily visualized in the input space via saliency maps. Therefore, users can not provide feedback on the input features to improve the models. Besides, deep representations are often used for downstream tasks, or shared across multiple tasks in multitask learning. Hence a bug in a deep representation may get propagated to a wider range of tasks and models.

In order to debug on the level of deep representations, we focus on VAEs [108, 187] in this chapter for three reasons. First, the basic assumption behind VAEs is that the observed data is generated from a set of underlying latent factors of variations, so VAEs are designed to learn a compact low-dimensional representation that captures the latent factors. This means, the latent representation is meant to capture the most salient features of the training data [69] and this in turn makes the model as well as the data-generating process easier to explain. Second, VAEs are a generative model. Like discriminative models, generative models are susceptible to exhibit dataset bias as well, for instance the PULSE algorithm [145] that yields a white male given a low-resolution input image of Barack Obama. For generative models, this may incur worse consequences. Trained to understand the world represented in the given training data, a generative model can provide answers to many inference problems [69]. Consequently, more inference tasks could be negatively influenced. Also, generative models can easily amplify the bias by generating more of the biased data at test time [77]. Third, compared to other generative models based on latent representations such as Generative Adversarial Networks (GANs) [70], VAEs are less tricky to train.

A training dataset that contains spurious artifacts in the latent factors violates the i.i.d. assumption and has insufficient sample complexity. We postulate that VAEs trained on this type of data may induce entanglement in its latent representations by associating

[1] Recall the examples of Google Translate and a risk prediction tool in Section 2.
[2] Source: https://twitter.com/Chicken3gg/status/1274314622447820801.

representations of seemingly correlated but actually distinct factors. A direct consequence is that the model is not able to generalize beyond the observed combinations of factors. Montero et al. [153] recently showed that disentangled representations support weak combinatorial generalisation — the ability to understand and produce novel combinations of familiar elements. Therefore, we propose to debug and improve entangled VAEs by interactively learning disentangled representations using user feedback on the latent factors, which is encoded as a small set of additional examples.

9.2. Disentangled Representation

Lacking a formal definition, a representation is by intuition called *disentangled* if its variables correspond directly and separately to the distinct, informative factors of variation underlying the data [12, 229, 128, 36, 27, 86]. Learning disentangled representation is an active area of research that increasingly receives attention [85, 115, 106, 30, 20, 141]. Popular unsupervised approaches are β-VAE [85], FactorVAE [106] and β-TCVAE [30]. β-VAE [85] augments the ELBO with a coefficient β to balance latent channel capacity and independence constraints with reconstruction accuracy. FactorVAE [106] encourages the distribution of representations to be factorial and hence independent across the latent dimensions. β-TCVAE [30] proposes an equivalent objective as FactorVAE but optimizes it differently.

However, the unsupervised learning of disentangled representation was shown to be fundamentally impossible from i.i.d. observations without inductive biases or some form of supervision [129]. Thereafter this research problem started to be addressed with supervision to provide general solutions [189, 17, 93, 29, 131, 220, 225, 103]. Chen et al. [29] used weak supervision by providing paired instances that share a underlying factor and proposed a regularized ELBO to enforce disentanglement. Shu et al. [220] worked out a theoretical framework to analyze the disentanglement guarantees of weak supervision algorithms. In particular, they decomposed disentanglement into consistency and restrictiveness. Locatello et al. [130] also used paired observations as weak supervision, but this method requires weaker assumptions than Shu et al. [220]. Locatello et al. [131] used a small number of labels to learn disentangled representations. Besides, Locatello et al. [131] verified that the supervised regularizer they considered relies on the inductive bias given by the ordinal information present in the labels, and such inductive biases should generally be exploited whenever they are available. Montero et al. [153] also found that disentangling without supervision is not sufficient for supporting more difficult forms of generalisation. Besides, they showed that using labeled data for training is beneficial both in terms of disentanglement and downstream performance.

Unfortunately, none of the aforementioned unsupervised or supervised learning methods for disentangled representation account for confounded training data. In the following, we explicitly deal with confounded training data that could lead to entangled generative models with poor generalization ability. Moreover, we build our work upon the framework of XIL using a small collection of weakly supervised examples as user feedback on the latent factors.

9.3. Disentangled Representation for Generative Models

Let $S_1, \dots S_n$ denote independent latent factors underlying the variation of the data, and let them follow some prior distribution $p(\mathbf{S}) = \prod_i p(S_i)$. We assume that they produce the observations $\mathbf{X}$ via a deterministic function g, i.e. $\mathbf{X} = g(\mathbf{S})$, yielding a distribution $p(\mathbf{X})$, and that $\mathbf{S}$ can be recovered via an encoder $\mathbf{S} = e(\mathbf{X})$. We consider a confounded training set, where the confounder arises from spurious correlations between the factors of variation as expressed by some other distribution $p'(\mathbf{S})$. We merely assume that it is faithful to the true distribution on the level of single variable marginals, i.e. $p'(S_i) = p(S_i)$, which is typically relatively easy to ensure. The resulting distribution of our training data is noted as $p'(\mathbf{X})$.

Our goal is to obtain a generative model of $\hat{p}(\mathbf{X}, \mathbf{S})$ that does not exhibit the spurious correlations between the factors of variation from the training data. In other words, the goal can be understood as twofold: On one hand, we seek a generator $\hat{g}(\cdot)$ that induces higher data likelihood $p(\mathbf{X})$ using MLE. On the other hand, we seek an inference function $\hat{e}(\cdot)$ that encodes the input as $\mathbf{Z}$ to approximately capture $\mathbf{S}$ while representing the factors of variation independently, i.e. $\mathbf{Z} = \hat{e}(\mathbf{X})$ is a disentangled representation. Let $\mathcal{I}(\cdot)$ denote mutual information. Our goal can be in turn regarded as

$$\max \mathcal{I}(S_i, \hat{e}(\mathbf{X})_i) + \min \mathcal{I}(S_{\setminus i}, \hat{e}(\mathbf{X})_i) + \min \mathcal{I}(S_i, \hat{e}(\mathbf{X})_{\setminus i}) \tag{9.1}$$

for each factor S_i, where the subscript $\setminus i$ means the complement of i.

Suppose that we have managed to perfectly disentangle the VAE's latent codes with regard to the true factors of variation, by ensuring that each Z_i captures exactly S_i for $i \leq n$. Then there exist bijections f_i deterministically mapping the two to each other: $Z_i = f_i(S_i)$, and their marginal probabilities will match, i.e.

$$p(S_i) = p'(S_i) = p(f(S_i))|\frac{d}{dS_i} f(S_i)|. \tag{9.2}$$

If, in addition, the VAE's decoder approximately matches the data generating procedure such that $\hat{g}(f(\mathbf{S})) \approx g(\mathbf{S})$, then it follows that the distribution $\hat{p}(\mathbf{X})$ represented by the

VAE matches the desired distribution $p(\mathbf{X})$. Letting $\delta_{\mathbf{X}}[\cdot]$ denote the Dirac delta at $\mathbf{X}$, we have

$$p(\mathbf{X}) = \int \delta_{\mathbf{X}}[g(\mathbf{S})] \prod p(S_i) d\mathbf{S} \tag{9.3}$$

$$\approx \int \delta_{\mathbf{X}}[\hat{g}(f(\mathbf{S}))] \prod p(f(S_i)) \left| \frac{d}{dS_i} f(S_i) \right| d\mathbf{S} \tag{9.4}$$

$$= \int \delta_{\mathbf{X}}[\hat{g}(\mathbf{Z})] \prod p(Z_i) d\mathbf{Z} = \hat{p}(\mathbf{X}), \tag{9.5}$$

where the last line follows from substituting $\mathbf{Z} = f(\mathbf{S})$, and cancelling out the differentials.

9.3.1. Disentangling via User Feedback

The maximum likelihood estimation of VAEs uses the ELBO as a proxy objective function. This is due to the fact that the inference of $p(\mathbf{X})$ is in general intractable, hence optimizing $p(\mathbf{X})$ directly is difficult. Refer to Section 3.4 for more details on the ELBO.

Optimizing the ELBO requires only unlabeled samples. However, recall that the unsupervised learning of disentangled representations is argued by Locatello et al. to be fundamentally impossible without inductive biases or some form of supervision [129]. We propose to combat this issue by interactively taking user feedback on the latent factors, which is encoded as a small collection of examples providing weak supervision[3]. Specifically, users provide pairs of samples that share the same label for the target factor S_i, while the other factors $S_{\setminus i}$ vary at random. In addition, users also provide a small number of samples for which the label of S_i is available.

Given the augmented data from the user feedback, the objective in Equation 9.1 is intractable in practice, and hence it can not be directly optimized. $\mathcal{I}(S_i, \hat{e}(\mathbf{X})_i)$ can be rewritten using marginal and conditional entropy as $H(S_i) - H(S_i|\hat{e}(\mathbf{X})_i)$. Since $H(S_i)$ is constant w.r.t. the parameters, maximizing $\mathcal{I}(S_i, \hat{e}(\mathbf{X})_i)$ then amounts to minimizing $H(S_i|\hat{e}(\mathbf{X})_i)$. We use an auxiliary function $h(.)$ trained on the few augmented labeled samples to minimize the cross-entropy loss $R_i(S_i, \hat{e}(\mathbf{X})_i)$ in order to approximate $p(S_i|\hat{e}(X)_i)$. This term penalizes trivial solutions (i.e. $\hat{e}(\mathbf{X})_i$ being constant) by enforcing $\hat{e}(\mathbf{X})_i$ to be predictive for S_i. By explicitly enforcing non-trivial solutions, we make sure the representation is not only disentangled, but actually meaningful. For an analogous reason as above, minimizing $\mathcal{I}(S_{\setminus i}, \hat{e}(\mathbf{X})_i)$ amounts to maximizing $H(S_{\setminus i}|\hat{e}(\mathbf{X})_i)$, where the maximum is $H(S_{\setminus i})$. Towards this goal, we use the weakly supervised pairs of samples $(\mathbf{X}, \mathbf{X}')$. Following Shu et al., we minimize the match-pairing loss $\mathbb{E}[(\hat{e}(\mathbf{X})_i - \hat{e}(\mathbf{X}')_i)^2]$ for these samples.

[3] Weak supervision is becoming popular in the disentanglement literature [29, 220, 130].

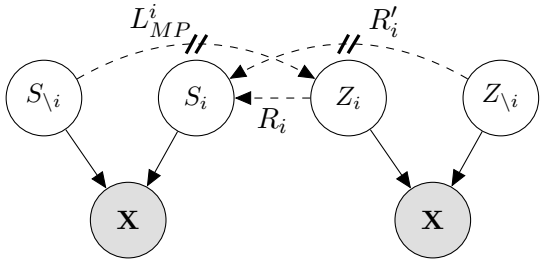


Figure 9.1.: Illustration of the purpose of the disentanglement losses. The match-pairing loss prevents unrelated factors of variation $S_{\setminus i}$ from having an influence on Z_i. In turn, the positive classification loss R_i ensures that Z_i is predictive for S_i, while the negative classification loss R'_i ensures that $Z_{\setminus i}$ is not.

Intuitively, this loss encourages latent representations that are invariant to various labels of factor $S_{\setminus i}$ in certain dimensions. The maximum of $H(S_{\setminus i}|\hat{e}(\mathbf{X})_i)$ can be achieved when $\hat{e}(\mathbf{X})_i$ is invariant to the varying factor $S_{\setminus i}$. Finally, minimizing $\mathcal{I}(S_i, \hat{e}(\mathbf{X})_{\setminus i})$ means we want that S_i is not encoded in $\hat{e}(\mathbf{X})_{\setminus i}$, i.e. S_i is restricted to $\hat{e}(\mathbf{X})_i$. This equals maximizing $H(S_i|\hat{e}(\mathbf{X})_{\setminus i})$. Since $\hat{e}(\mathbf{X})_{\setminus i}$ may contain nuisance variables that we may not be able to account for or wish to purposefully ignore, we use a minmax game. Specifically, we train an auxiliary regression model in parallel with the main model to predict S_i from $\hat{e}(\mathbf{X})_{\setminus i}$ by minimising the cross-entropy loss $R_i(S_i, \hat{e}(\mathbf{X})_{\setminus i})$. Then the encoder tries to maximize $R_i(S_i, \hat{e}(\mathbf{X})_{\setminus i})$ by inducing $\hat{e}(\mathbf{X})_{\setminus i}$ that shares less information with S_i.

Following the arguments above, we design a loss function to disentangle latent representations of VAEs using a small set of examples as user feedback. The loss function $L(\mathbf{X})$ is given by

$$-\,\mathrm{ELBO}(\mathbf{X}) + \sum\nolimits_i \lambda_1 L^i_{MP}(\mathbf{X}) + L^i_{CL}(\mathbf{X}), \tag{9.6}$$

where $L^i_{MP}(\mathbf{X}, \mathbf{X}') = \mathbb{E}[(Z_i - Z'_i)^2]$ and $L^i_{CL}(\mathbf{X}) = \mathbb{E}_{q(Z|\mathbf{X})}[\lambda_2 R_i(S_i, Z_i) - \lambda_3 R'_i(S_i, Z_{\setminus i})]$. $L^i_{MP}(\mathbf{X}, \mathbf{X}')$ and $L^i_{CL}(\mathbf{X})$ are only computed for all data points and factors of variation i for which the user feedback is available. The way these losses interlock is illustrated in Figure 9.1. Figure 9.2 illustrates the resulting computation graph.

9.3.2. Analysis of Property Guarantees

As Locatello et al. [129] theoretically proved, unsupervised disentanglement learning is fundamentally impossible without inductive biases both on models and data sets. Here, we explicitly stress that disentanglement arises from our supervision method instead of

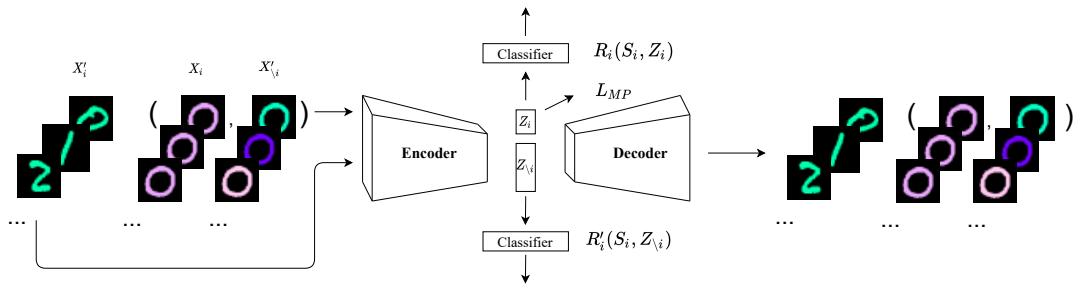


Figure 9.2.: Our approach uses a small amount of user feedback in addition to the training set to learn disentangled latent representations. For the sake of simplicity, only augmented loss terms are illustrated here.

the model's inductive bias. We use the definition of disentanglement by Shu et al. where disentanglement is decomposed into *consistency* and *restrictiveness*. Strong disentanglement is guaranteed when both consistency and restrictiveness hold. According to Shu et al., S_i is *consistent* with Z_i if

$$\mathbb{E}[||\hat{e}_i \circ g(S_i, S_{\setminus i}) - \hat{e}_i \circ g(S_i, S'_{\setminus i})||^2] = 0,$$

and S_i is *restricted* to Z_i if

$$\mathbb{E}\left[||\hat{e}_{\setminus i} \circ g(S_i, S_{\setminus i}) - \hat{e}_{\setminus i} \circ g(S'_i, S_{\setminus i})||^2\right] = 0.$$

Intuitively, consistency implies that Z_i does not capture the factors of variation other than S_i, and restrictiveness implies that S_i is only captured in Z_i. In addition, Shu et al. made it explicit that $\mathbf{Z}$ should be *non-trivial*, i.e., it should have non-zero variance to encode the variation of $\mathbf{S}$. It is a known issue with VAE models that they tend to use only a small subset of the latent dimensions [69].

We consider two cases: disentangling *all* factors, and disentangling only a subset of factors, leaving the rest as arbitrarily encoded nuisance factors. Handling of nuisance factors can be considered as a special case of disentangling a subset of factors. Since our method builds on match pairing, the theoretical guarantees of each property transfer trivially to our method when the guarantee is fulfilled. Therefore, we only need to prove that the additional labels we introduce are sufficient for learning restrictive and non-trivial solutions when disentangling a subset of factors.

Define a hypothesis space $\mathcal{H}$ of models we are willing to consider. Let $(p(\mathbf{S}), p'(\mathbf{S}), g, e) \in \mathcal{H}$ denote an arbitrary ground-truth model which generated the data we are observing. A

supervision method $\mathcal{S} : \mathcal{H} \rightarrow \mathcal{P}$ is called *sufficient* for a certain guarantee if there exists a learning algorithm $\mathcal{A} : \mathcal{P} \rightarrow \mathcal{H}$ which uses the supplied examples to produce a learned model $(p(\mathbf{Z}), \hat{g}, \hat{e})$ for which the desired guarantee holds. In the following, we show that our method of supervision is sufficient for non-trivial and restrictive solutions. We do so by considering an $\mathcal{A}^*$ which obtains the global minimum of the classification loss L^i_{CL}, despite using universal approximators for R and R'.

Non-Triviality Guarantee. We claim that our method guarantees that the learned model $(p(\mathbf{Z}), \hat{g}, \hat{e})$ provides non-trivial solutions for Z_i, i.e. there is a learning algorithm that yields $S_i \not\perp\!\!\!\perp Z_i$. Match pairing alone does not entail this guarantee when disentangling only a subset of factors. To see why, consider a simple counterexample with underlying factors $S_1, S_2 \sim \mathcal{N}(0, 1)$, and generator $g(\mathbf{S}) = S_1 \cdot S_2$. There exists a learning algorithm $\mathcal{A}$ which optimizes the match pairing loss on factor S_1 by setting $(g(S_1, S_2), g(S_1, S_2^{'})) = (\hat{g}(Z_1, Z_2), \hat{g}(Z_1, Z_2^{'}))$, and $\hat{g}(Z) = [Z_1, Z_2] = [0, Z'_1 \times Z'_2]$ where $Z'_1 \sim \mathcal{N}(0, 1)$, $Z'_2 \sim \mathcal{N}(0, 1)$. In this case, match pairing yields a trivial solution for Z_1, i.e. $Z_1 = 0$.

Proof: A learned model $(p(\mathbf{Z}), \hat{g}, \hat{e})$ provided by $\mathcal{A}^*$ minimizes the classification loss, and the positive term $R^i(Z_i, S_i)$ in particular. Consequently, it is possible to accurately predict S_i from Z_i with a classifier $\mathcal{C}_i : \mathcal{Z}_i \rightarrow \mathcal{S}_i$ and $S_i \not\perp\!\!\!\perp Z_i$, that is, Z_i is informative for S_i. Suppose to the contrary that there is a learning algorithm $\mathcal{A} : (\mathcal{S}_i, \mathcal{P}) \rightarrow \mathcal{H}$ such that $Z_i = 0$. Then the variance of $\mathcal{C}_i(Z_i)$ equals 0, therefore Z_i is not informative for S_i. Contradiction. □

Restrictiveness Guarantee. We postulate that our method guarantees that the learned model $(\hat{p}(\mathbf{Z}), \hat{g}, \hat{e})$ restricts the information of S_i to Z_i. That is, $Z_{\setminus i} \perp\!\!\!\perp S_i$.

Proof: Suppose to the contrary that $Z_{\setminus i}$ contains information about S_i. Then we could find a regression model that predicts S_i from $Z_{\setminus i}$ better than random guesses, i.e., the value of $-R'(S_i, Z_{\setminus i})$ would be suboptimal. This is a contradiction, since we assumed that $\mathcal{A}^*$ would achieve the global minimum of L^i_{CL}. □

9.3.3. Interactive Data Augmentation

Data is the fuel for training modern machine learning models. In general, machine learning models tend to overfit when the available training data is insufficient. Among the variety of regularization techniques that intend to reduce overfitting, data augmentation implicitly regularizes the model by encoding inductive bias in an augmented set of samples. This set of samples can be obtained by transforming the available data while keeping their labels invariant based on prior knowledge of domain invariances. For example, the task of natural image classification is invariant to image rotation and translation, which can be utilized to augment the training samples by rotating and translating the available images. This technique is effective and ubiquitously used [136, 49, 200]. A review by Ratner et al.

Table 9.1.: Properties of our and the match pairing approach in case of disentangling a subset of factors and disentangling all factors.

		consistency	restrictiveness	disentanglement (consistency ⋀ restrictiveness)	non-trivial guarantee
Ours	subset of factors	✓	✓	✓	✓
	all factors	✓	✓	✓	✓
Match Pairing	subset	✓	✗	✗	✗
	all factors	✓	✓	✓	✓

states that 10 out of 10 of the top CIFAR-10 results and 9 out of 10 of the top CIFAR-100 results use data augmentation, for average boosts (when reported) of 3.71 and 13.39 points in accuracy, respectively [185].

Data augmentation can be viewed as a form of weak supervision [185]. In our work, a small number of augmented examples are used to emulate user feedback. Specifically, this refers to a set of transformed samples with one underlying factor invariant at a time. For example, in the colored MNIST domain, we assume two underlying factors: shape and color. The augmented data based on a red zero consists of two sets: one set of red various digits and one set of differently colored zeros. This is interactively given by users as their additional feedback signal to the training process, in response to unsatisfactory training results. This way, we allow users to interact with the training process and participate in the training loop, which pushes the VAEs to better align with the goal of the user and potentially also adapt over time. In contrast to the approach in Chapter 8, which forms an interactive training loop with the users for their feedback on the raw features, the approach in this chapter seeks to interactively incorporate user feedback for the latent factors using data augmentation.

9.4. Empirical Evaluation

In this section, we present experiments to empirically investigate the core research question: Does our model learn better latent representations than the mainstream baseline models? This question then further breaks down to the following specific questions:

(Q1) Are the reconstructions from our model less biased towards the samples that are present in the training set?

(Q2) Do the latent representations capture all the possible combinations of factors even if

Table 9.2.: Summary of the datasets. Parentheses give the number of quantized values for each target factor. The last column gives the size of samples we chose from the training set to perform data augmentation.

Name	Training instances	Held-out instances	Image size	ground-truth factor	Data Augmentation
colored MNIST	60k	10k	$28 \times 28 \times 3$	shape(10), color(10)	600
colored dSprites	10k	10k	$64 \times 64 \times 3$	shape(3), color(3), nuisance	1k
3d shapes	10k	10k	$64 \times 64 \times 3$	object shape(4), object color(4), nuisance	1k

some were not present in the training set?

(Q3) Does each latent dimension capture variations of at most one factor?

(Q4) Does our model yield a better disentanglement score?

(Q5) Does our model benefit downstream tasks?

To answer these questions, we ran a series of experiments for both qualitative and quantitative evaluation.

9.4.1. Dataset

We considered several standard benchmark datasets. First, we generated a colored version of MNIST [120] in a similar way as Kim et al. [105]: To inject color bias, ten distinct colors were chosen and each color was assigned to only one digit class in the training set so that the shape class and the color class correlated with each other. In the test set, one could assign random colors to each image. However, we made the problem even more challenging by assigning the color class reversely correlated to the shape class so that none of these combinations of shape and color were seen in the training set. In this domain, we assumed two latent factors of variations: shape and color.

As another dataset, we generated colored dSprites [85] in a similar spirit. We randomly sampled 10k instances as the training set and synthesized confounding in it by consistently

assigning one color to each shape category and the rest factors remained untouched. In the test set, we assigned each color to the shape with one offset as to in the training set, so that none of these two factors' combinations were seen in the training set. Other than shape and color, this dataset has other factors of variation such as scale, orientation, position X and position Y. However, we treated all the factors other than shape and color as nuisances to test the effectiveness of our method when dealing with nuisance factors.

The third dataset is 3D Shapes [106]. We sampled 10k instances to be the training set whereby each object hue category correlated to one object shape category. In the test set, each object hue category correlated to the object shape category with one offset as to in the training set for the same reason as mentioned above. Other than shape and color of the object, this dataset contains also other factors of variation such as floor hue, wall hue, scale and orientation, which were treated as nuisance factors. Table 9.2 summarizes these three datasets.

9.4.2. Experimental Protocol

We followed the experimental protocol of Locatello et al. [131, 130] and aligned with the general literatures on disentanglement [85, 106, 30]. The baseline models we used are β-VAE [85], FactorVAE [106], β-TCVAE [30] and Ada-GVAE [130]. These models are mainstream VAEs for benchmarking disentangled representations. In addition, we also evaluated our VAE with unannotated human feedback to investigate the value of the annotations. We implemented all the models in Python and Pytorch. For each model, we ran experiments with 8 random seeds. For each baseline model, we used 3 fixed hyperparameters on each dataset. Our model's hyperparameters were chosen empirically according to the quality of the reconstructions. Our experience suggests using hyperparameters $\lambda_1 = \lambda_2$ and $\lambda_3 = 1$ or $\lambda_3 = 10$ leads to good results in practice, so we propose to tune the hyperparameters within this range. We used linear classifiers for R_s and R'_s in equation (9.6). We relaxed the condition for R'_s to be a universal classifier in practice and demonstrated that this can still yield promising disentangled representations empirically.

9.4.3. Reconstruction Quality

The following experiment is designed to investigate the research question **(Q1)**. As noted previously, we synthesized the test samples from completely new combinations of factors, different than those used for training the models. The intention is to qualitatively inspect how well each model generalizes under non-i.i.d. distributions. This is based on the assumption that disentangled representations can generalize to all the combinations

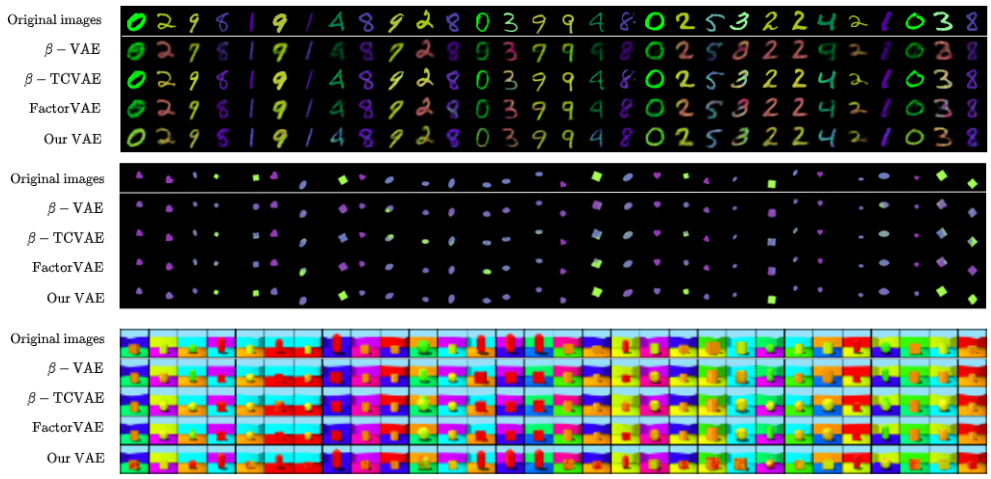


Figure 9.3.: Reconstructed images from different models on the colored MNIST, the colored dSprites and the 3D shapes.

of the underlying factors, otherwise the model would struggle to encode all the target latent factors in the unseen combinations, which in turn would lead to unsatisfactory reconstructions. Figure 9.3 shows some randomly chosen test samples across the three datasets and their reconstructions via different models. Each shape factor is consistently associated with one color factor in the training set, while in the test set the association is different. For all the three datasets, the object shapes and object colors reconstructed by the baseline models still exhibit more or less the same association than in the training set, even if the input images have more generalized combinations of shape and color. The direct consequence is that some of their reconstructions are qualitatively different from the input images. This implies that these models have learnt entangled latent representations. In contrast, the reconstructions by our model are more consistent with the original input images. This robustness w.r.t. generalization on the unseen combinations of factors is achieved by disentangling the latent representations.

9.4.4. Latent Space Arithmetic

The following experiment is designed to answer the research question **(Q2)**. We demonstrate that our model can not only generalize under non-i.i.d. distributions, but can also generate novel content. Figure 9.4 (middle and right) and 9.5 show all possible hybridization of shapes and colors generated by our VAE in each corresponding domain. Each

Figure 9.4.: Left: Our model is trained on a confounded version of the colored MNIST dataset, in which color and shape are fully correlated (diagonal marked in orange). As user feedback, we supply a small number of labelled examples from the first row and the rightmost column. Middle and Right: Despite never observing most combinations of color and shape (observed combinations are marked in orange and green), our model learns to generate to all of them. Here, reconstructions are drawn by combining the shape in the top row with the color of the digit in the leftmost column.

hybridized image was generated by concatenating the latent representations of one image's shape on the leftmost column and another image's color on the first row. When nuisance variables were present, they took the values of the average of both reference images' nuisance representations. The result shows that the hybridized images very plausibly resemble both the shape from the leftmost-column images and the color from the first-row images, and this yields and recovers the whole spectrum of the target domain where only a small fraction of which were seen in the training set. This not only allows us to generate novel contents, but also grants us opportunities to manipulate the latent space and generate specific samples for downstream tasks.

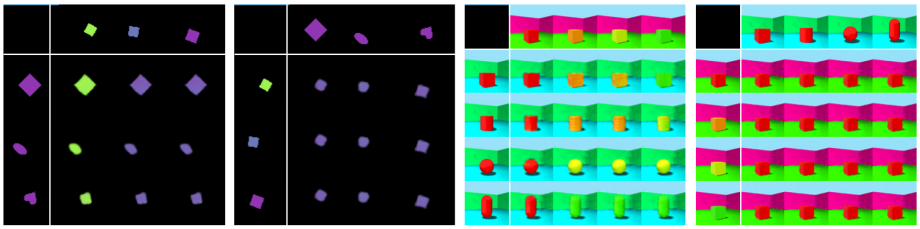


Figure 9.5.: Cross-product of shapes (leftmost column) and colors (top row) generated by the interactively learnt VAE.

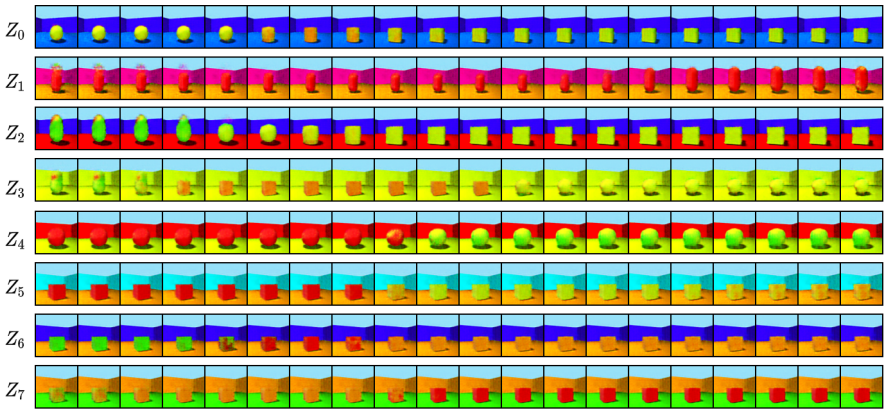


Figure 9.6.: Latent space traversal of our VAE. The first (last) 4 rows are respectively the traversal of Z_0, Z_1, Z_2 and Z_3 (Z_4, Z_5, Z_6 and Z_7) which are collectively intended for the shape (color) factor.

9.4.5. Latent Space Traversal

This experiment is motivated by the research question **(Q3)**. To give another view of the latent space and what the generative models have really learnt for the target factors, we show partial latent space traversals of our model for the 3D shapes dataset. Inspecting latent traversals is a popular way of measuring disentanglement by visualising the change in reconstructions while traversing one dimension of the latent space at a time [106]. The latent variables of the visualized model are in total 50. We designated the first 4 variables for learning the shape factor, and the second 4 variables for the color factor, the rest are considered nuisance variables. Figure 9.6 shows samples generated by traversing the first 8 latent variables. The first (respectively the last) 4 rows correspond to the variables that are supposed to learn the shape (respectively the color) factor. The samples show that varying the first (respectively the last) 4 variables leads to varying object shape (respectively the color) while the rest factors, including the nuisance factors, remain mostly unchanged. This suggests that each unit of the representation has *exclusively* captured the intended factor (restrictiveness property). Moreover, the nuisance factors, e.g. wall color, seem not to be captured by these 8 variables which implies the latent representations are consistent with the target factors (consistency property).

9.4.6. Disentanglement Metrics

Motivated by the research question **(Q4)**, we report four heavily used quantitative metrics for measuring disentanglement, following the literature on disentangled representation: FactorVAE score [106], DCI Disentanglement and DCI Completeness [51] and Mutual Information Gap (MIG) [30].

FactorVAE score measures disentanglement based on ground truth factors. Specifically, a set of samples with one chosen factor k fixed and all other factors varying randomly are used to obtain their representations, which are then normalized for each dimension. The index of the dimension with the lowest variance and the ground truth factor k constitute one input and output signal for training a classifier. The intuition is that the empirical variance of the variable in correspondence to the factor k should be 0 if the representation is disentangled.

DCI involves three aspects of disentangled representations: disentanglement, completeness and informativeness. The evaluation is based on K regressors that are trained to predict the value of K true factors $\mathbf{S}$ given the learnt representation $\mathbf{Z}$. The regressors are used to yield the relative importance of Z_i in predicting S_j, noted as R_{ij}. DCI Disentanglement score quantifies the disentanglement of each latent variable Z_i by $1 - H_K(P_i)$ where $H_K(P_i) = -\sum_{k=0}^{K-1} P_{ik} \log_K P_{ik}$ denotes the entropy and $P_{ij} = R_{ij} / \sum_{k=0}^{K-1} R_{ik}$ denotes the probability of Z_i being important for predicting S_j. This score measures the degree to which each latent variable captures at most one generative factor, which gives a direct hint on the consistency property. DCI Completeness score is given by $1 - H_D(\tilde{P}_{\cdot j})$, where D is the dimension of the latent representations and $H_D(\tilde{P}_{\cdot j}) = -\sum_{d=0}^{D-1} \tilde{P}_{dj} \log_D \tilde{P}_{ij}$ denotes the entropy of the $\tilde{P}_{\cdot j}$ distribution. This score measures the degree to which each factor is captured by a single latent variable. Although our model does not assume a single latent variable for each factor and therefore this metric is not in favor of us, this score still gives a hint on how densely the information of the latent factors is packed in the latent variables, and in turn on the restrictiveness property.

MIG is defined based on the empirical mutual information $\mathcal{I}_n(Z_j; s_k)$ between a latent variable Z_j and a ground truth factor S_k. Let n be the empirical data samples and $q(Z_j, S_k) = \sum_{n=1}^{N} p(S_k) p(n|S_k) q(S_j|n)$ be the joint distribution, then

$$\mathcal{I}_n(Z_j; S_k) = \mathbb{E}_{q(Z_j, S_k)} \left[\log \sum_{n \in \mathcal{X}_{S_k}} q(Z_j|n) p(n|S_k) \right] + H(Z_j)$$

where $\mathcal{X}_{S_k}$ is the support of $p(n|S_k)$ and $H(Z_j)$ is the entropy of Z_j. MIG is in turn given

by

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{H(S_k)}(\mathcal{I}_n(Z_{j^{(k)}};S_k)-\max_{j\neq j^{(k)}}\mathcal{I}_n(Z_j;S_k))$$

where $H(S_k)=\mathbb{E}_{p(S_k)}[-\log p(S_k)]$, $j^{(k)}=\arg\max_j \mathcal{I}_n(Z_j;S_k)$ and K is the number of known factors. Since this quantity enforces axis-alignment and our VAE does not make this assumption, this metric in its original form is not fit for evaluating our VAE. Therefore we slightly adapt MIG by constraining the gap computation between the known unit of the latent representations and the rest representations. In particular, we slightly reformulate it to the following form:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{H(S_k)}(\max_{j\in J}\mathcal{I}_n(Z_j;S_k)-\max_{j\notin J}\mathcal{I}_n(Z_j;S_k))$$

where J is the unit of the latent representations designated for S_k which can contain more than one latent variable. The higher this quantity is, the more densely the information about S_k is packed in Z_j. Since the baseline models make the axis-alignment assumption, we used the original form of MIG to evaluate them.

As these metrics are not conceptualized accounting for biased data, we used the whole spectrum of the target domain, i.e. i.i.d. samples, to evaluate them.

Figure 9.7 shows the distribution of these scores from different models across different datasets via violin graphs. All the metrics are bounded by 0 and 1. Higher values imply better performance. As one can see, our method yields significantly better performance in terms of all the metrics, than the baseline models. In terms of the FactorVAE score, although the baseline models have also achieved almost the upper bound of the FactorVAE score by their best performance, our model still has less variance across different runs, especially on the colored MNIST dataset, where we reached 1 for every random run. This demonstrates that our method is significantly better at learning disentangled representations on the considered datasets.

9.4.7. Downstream Accuracy

Learning data representations can yield features of the data that can be further used in downstream tasks. The quality of data representations can have a significant impact on the performance of downstream tasks. Specifically, more compact and robust representations should in general lead to better downstream performance. Here, we investigate the

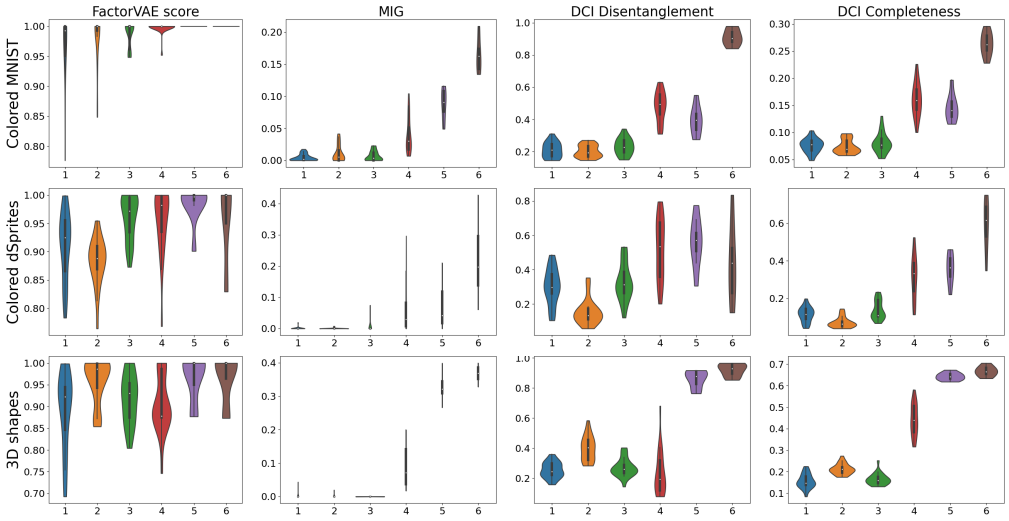


Figure 9.7.: Several disentanglement metrics (FactorVAE score, MIG, DCI disentanglement and DCI completeness) across different models on the colored MNIST dataset (top row), the colored dSprites dataset (middle row) and the 3D Shapes dataset (bottom row). 1: β-VAE, 2: β-TCVAE, 3: FactorVAE, 4: AdaGVAE, 5: Our VAE without annotations, 6: Our VAE with annotations. The higher the value, the better the performance.

research question **(Q5)** by comparing the performance of a downstream classifier learnt from different representations.

For each VAE, we trained a post-hoc linear classifier on each unit of the latent representation to predict the values of the corresponding factor. We used the biased training set without augmented data for all the classifiers so that we can fairly compare the quality of the learnt representations and exclude the influence of differently distributed training data for each classifier. For evaluation, we used the same set as in the reconstruction experiment (see Section 9.4.3). This way, we made the test set a different distribution to the training set on purpose, so we could compare the generalization ability and robustness of different representations under covariate shifts [177]. In addition, labels for different factors did not overlap on the test set so that the prediction of shape and color was not ambiguous and could be distinguished.

Figure 9.8 reports the distribution of the test set accuracies. Our model results in significantly better downstream accuracies, achieving 100% accuracies at its best. This is compelling evidence that our model yields more robust features for downstream classifiers

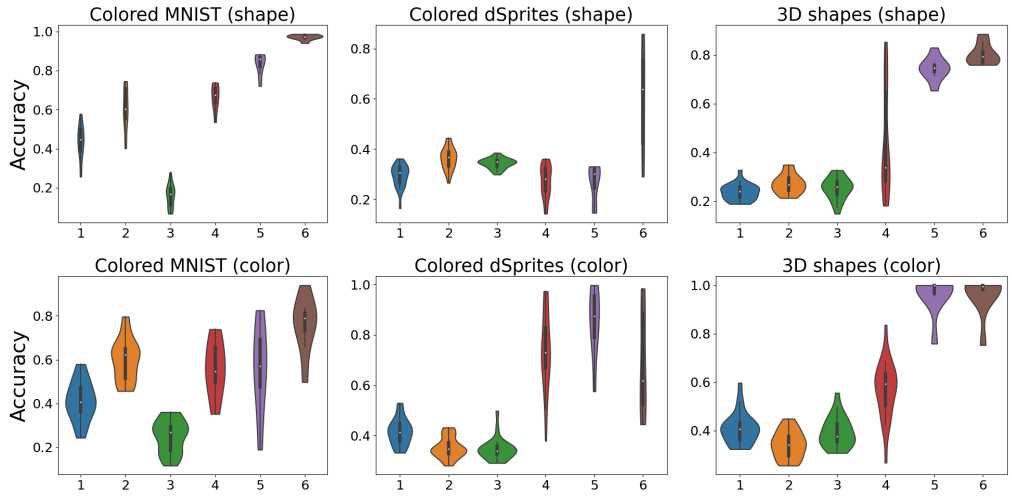


Figure 9.8.: Accuracies of the post-hoc linear classifiers after disentangling the shape factor (top row) and the color factor (bottom row). 1: β-VAE, 2: β-TCVAE, 3: FactorVAE, 4: Ada-GVAE, 5: Our VAE without annotations, 6: Our VAE with annotations. The higher the value, the better the performance.

on the considered datasets.

This concludes Chapter 9. We started by discussing the effect of confounded datasets on deep generative models. A confounded dataset may exhibit bias from the data-collecting process which may be propagated to the models. A biased generative model can exhibit Clever Hans-like phenomenon which is closely connected to its entangled latent representations. Therefore, we proposed to learn disentangled representations via interactive data augmentation to debug deep generative models. Empirical evidence has demonstrated the effectiveness of our approach in learning disentangled representations.

Part IV.

Conclusion

10. Conclusion

Deep neural networks and deep probabilistic models are important components in machine learning. They are used for different scenarios. The former are typically employed for dedicated predictive tasks, the latter are used for descriptive tasks and allow for automatic reasoning in the presence of uncertainty. Deep models have demonstrated high representational and predictive power in many complex problems. However, they lack interpretability, which means the reasons for their decisions are not inherently understandable by a human. Interpretability is useful for multiple reasons, like increasing the social acceptance of automated decisions, gaining knowledge and verifying machine learning systems. In this thesis, we aim to interpret deep models and interactively improve them if necessary. In particular, we propose a tractable conditional probabilistic model that can be used to impose interpretable high-level structures for deep generative models. To interpret deep models on the local level, we present a time-efficient approach to generate counterfactual examples as post-hoc explanations. Finally, we show how to improve machine learning models by inspecting and correcting machine explanations.

10.1. Summary

This thesis is divided into three parts. The first part deals with the fundamentals, followed by a review of a set of deep models in the second part. In the third part, we focus on explaining and interactively debugging deep models.

Chapter 1 gives an introduction of our motivation and the tasks we deal with in this thesis — interpreting and debugging deep models. Then, we give a detailed review of explainable AI in Chapter 2, which includes local and global explainable methods. Based on the machine explanations, some approaches for passively or interactively learning from explanations are introduced. The framework for interactively learning from explanations is also called Explanatory Interactive Machine Learning (XIL), which lays the foundation for our research.

In Chapter 3 we give a brief introduction to DNNs, and how they are learnt from data. Then some commonly used regularization strategies for improving the generalization

error of deep models are presented. Some strategies add hard constraints on the model, some add a penalty term in the objective function as soft constraints, and some use data augmentation [69]. Regularization techniques are an essential component of our research for improving machine learning models. In particular, we adopt both the strategies of soft penalty and data augmentation, which will be elaborated in Chapter 8 and 9. At the end of Chapter 3, we present a deep generative model based on neural networks — Variational Autoencoders (VAEs) which will be explained and improved in Chapter 9.

Following predictive models, Chapter 4 gives a brief introduction to a descriptive model — Sum-Product Networks (SPNs). It starts with the notion of network polynomial, which is the key idea behind SPNs. Then three components of SPNs — model representation, inference routines, and learning algorithm — are individually explained. Afterwards, a special type of SPNs with randomized structure, Random and Tensorized Sum-Product Networks (RAT-SPNs), is introduced. RAT-SPNs provide scalable solutions to construct and learn SPNs, which often perform on par with DNNs. They will be interpreted with counterfactual examples in Chapter 7.

In the following, a conditional counterpart of SPNs — Conditional Sum-Product Networks (CSPNs) — is introduced in Chapter 5. In essence, they are SPNs whose parameters are determined by an external function $f(\cdot)$ and input $\mathbf{X}$. The structure learning algorithm and inference routines are analogous to SPNs. CSPNs are useful for representing conditional probabilistic distributions $P(\mathbf{Y}|\mathbf{X})$ which are essential building blocks in many generative models as well.

After introducing a set of deep models, we start investigating how to interpret them. To make deep models interpretable on the high level, we impose modular structures on deep generative models using CSPNs in Chapter 6. In particular, we build a deep autoregressive model using a collection of CSPNs, as well as VAEs using CSPNs as the encoder and the decoder. In addition to interpretable domain structure, using CSPNs as building blocks in generative models has also the benefit of allowing for a set of tractable inference tasks. Moreover, modularity allows for flexible modelling of different granularities.

Sometimes, interpreting on the high level is not helpful enough and more specific explanations are desired for each individual inference. For example, if your loan application is rejected by a bank based on an automated decision, you probably would like to know how to improve your profile next time. Towards this end, we propose Sum-Product networks Interpretation by Counterfactual Examples (SPICE) in Chapter 7. SPICE generates counterfactual examples based on tractable probabilistic inference provided by SPNs, which makes it much faster to compute than the mainstream iterative counterfactual approaches. Moreover, SPICE naturally yields examples with high likelihood which is usually a desirable feature.

Ultimately, our goal of explaining deep models is to improve them when necessary. Our

approach to this task is to regularize the model estimator by adding a penalty term in the objective function as soft constraints. Built on the framework of XIL, we present in Chapter 8 how to correct deep models from the Clever Hans effect when they have learnt the confounding factors in the training set. We show that by interactively penalizing the wrong machine explanations, deep models can be regularized to learn the right features. This is important for reducing the generalization error and making the model more robust to irrelevant perturbations in the data. Confounding factors in the training set can also be detrimental to generative models. In Chapter 9, we demonstrate that confounding factors can lead to entangled latent representations in VAEs. Then, we show how to learn the right latent factors. Specifically, the right latent factors are learnt by disentangling the independent factors. This is achieved by regularizing the VAE estimator using a small set of examples as interactive user feedback on the latent factors. Disentangled representations have many benefits. First, they are more interpretable than entangled representations. Second, they can lead to better performance in many downstream tasks. Third, they can achieve combinatorial generalization. Empirical evidence in Chapter 8 confirms these benefits of disentangled representations.

10.2. Lessons Learned

There are many lessons that we have learnt during these years of research. Some of them are useful for particular problems, some have more general meaning that can guide us in the future.

The ultimate goal of this thesis is to learn deep models that produce the right machine explanations. Explanations can be used to verify machine learning systems in addition to the task-specific metrics. When a model performs well on a particular metric, but has wrong machine explanations, this could imply that the model has converged to a suboptimal solution. Our approach to counteract this problem essentially is to use an overparameterized model with some amount of regularization to restrict the solutions the model can learn. Loosely defined as any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error [69], regularization can take many forms, for example data augmentation and penalty in objective functions. It can also be done at different levels, for example, on the estimator or on the model. In our work, we primarily regularize estimators, which works by trading increased bias for reduced variance [69]. In particular, in Chapter 8, we add an additional term to the standard objective function for classification to penalize the wrong reasons. In Chapter 9, we add additional terms to the standard VAE loss to prevent the latent representations from learning more than one factor in each axis. The lesson here is that, instead of

trying to design a specialized model that matches well to a particular domain, using an overparameterized model with proper regularization to encode a prior knowledge or assumption is usually much easier. This also gives us more flexibility to adapt to slightly different conditions. More importantly, regularization can be imposed softly to account for noise or wrong information. As we have access to more and more computational resources, this lesson will become increasingly important in designing machine learning systems.

The other lesson is that regularization can be done in an interactive way, instead of being hard-coded throughout the whole training phase. In this work, we use explanations as one possible channel for interaction. By including users in the training loop, machine learning models can adapt to user expectations or preferences which may change over time.

Finally, generating explanations has been mostly treated as a discriminative task, both in the literature and in this work. However, in the end, we learnt that viewing explaining as a generative task and having a density model can be very beneficial. In particular, having an input distribution represented by SPNs allows us to generate much more realistic counterfactual explanations. For other types of explanations, density models may be very useful as well. Many explanation techniques that operate on the input layer, e.g. input gradient, generate very noisy saliency maps that appear similar to adversarial attacks. We conjecture that one of the reasons is because the explanations are not aware of the input distribution and extrapolate beyond the data manifold. Therefore, in retrospect, I think it would make this work even more interesting to research more about the role of density models for explanations.

10.3. Outlook

In this thesis, we have focused on explaining black-box deep models, including deep neural networks and deep probabilistic models. In particular, we have proposed a time-efficient approach — SPICE — to generate counterfactual examples as explanations. One interesting research avenue for future work is to investigate whether counterfactual examples generated by SPICE can be used as an additional channel for interactively regularizing the model. SPICE has a dominant advantage in computation time compared to the mainstream iterative counterfactual approaches, which potentially makes it a good fit as a signal for backpropagation. This would also be beneficial for human users in the training loop, because humans tend to think in counterfactual forms — what would need to be different to get an alternative outcome. Potentially, using counterfactual examples as a channel for interaction may help to reduce the required user feedback. The conjecture is based on the fact that counterfactual explanations are usually more spare and concise.

Furthermore, note that counterfactual examples and adversarial examples are very similar concepts. Specifically, they both are generated by perturbing the input to induce an alternative prediction. Therefore it is natural to study whether regularizing counterfactual examples in some way would make the model more robust to adversarial examples.

Additionally, the SPNs used in SPICE currently do not scale well to very large datasets, therefore they are used on top of the features extracted by convolutional neural networks for the Caltech-UCSD Birds (CUB) dataset. This has the consequence that we can not generate counterfactual examples in the original feature space. Therefore, SPICE can only highlight a coarse region of counterfactual features, but is not able to generate concrete examples. One could explore invertible convolutional layers to impute the examples in the original feature space. Alternatively, more research in enhancing the capabilities of SPNs, for example on the learning algorithm and optimization, could probably also be helpful for this problem.

Based on the machine explanations, we have further researched how to interactively improve deep classifiers when their explanations are wrong. To make a classifier right for the right reasons, we traded increased bias for reduced variance by regularizing the estimator using explanations. By increasing bias, potentially less data is required in the training phase to achieve satisfactory performance. In addition, an explanation can be viewed as a more compact representation of a collection of samples. Therefore, another interesting future project is to study the indirect effect of regularizing explanations on the training sample size. The reduced dependence on large quantities of training data is especially desirable for domains where samples are expensive to collect but prior knowledge is available, for example in the medical domain.

In Chapter 9 we have presented an approach to interactively improve generative models. This approach is based on data augmentation and may potentially be integrated with an additional module for automating the data augmentation process which can be substituted by generative models that take more compact and succinct user feedback. For instance, Ratner et al. proposed a method for programmatically generating domain-specific transformations for data augmentation by learning a generative sequence model over user-specified transformation functions using a generative adversarial approach [185]. As another example, Ramesh et al. proposed a generative model that creates images from text captions for a wide range of concepts expressible in natural language [181]. Integrating one of these models for user feedback could presumably benefit the users by allowing them to directly and flexibly leverage their domain knowledge of invariances in a more user-friendly way and in other modalities than image data.

In our work, the user, as well as the interactions between the user and the model in the training loop, are simulated. It would be interesting to include real human users in the training loop in the future to assess the effectiveness of our approach. Possibly, user

studies could be conducted to investigate whether the explanatory interactions used in our work boost the predictive and explanatory powers of the learned model and the trust of the human users in the model.

Machine learning must always deal with uncertain quantities and sometimes stochastic quantities [69]. Machine explanations are uncertain as well. The uncertainty comes from both the underlying machine learning models and the explanation process. However, the explanations we deal with in this work are deterministic. If the goal of explanations is to gain domain understanding or scientific insights, then the explanations on an uncertain model are probably not very helpful. Therefore, it is interesting to explore the distribution of explanations and quantify the uncertainty of explanations.

In this work, the explanations for black-box models are extracted by a separate fixed model in a post-hoc fashion. An interesting future project would be to learn the explanations along with the predictions. This way, explanations may be adapted to their audience. Moreover, the consistency between explanations and predictions may be constrained as well.

11. Selected Papers and Contributions

- **Xiaoting Shao**, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional Sum-Product Networks: Imposing Structure On Deep Probabilistic Architectures. *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM), Aalborg, Denmark*. Proceedings of Machine Learning Research. 2020.

  I was involved in developing the ideas and writing this paper. The ideas and the content have been discussed among all the authors. Together with Alejandro Molina, we have implemented the algorithms used in this paper and ran the experiments. The idea of gating nodes is proposed by Alejandro Molina.

- **Xiaoting Shao**, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, Kristian Kersting. Conditional Sum-Product Networks: Modular Probabilistic Circuits via Gate Functions. *International Journal of Approximate Reasoning*. 140, 298-313, 2022.

  I was involved in developing the ideas and writing this paper. I have implemented the differentiable sampling for CSPNs and the end-to-end training of CSPVAE. The ideas and the content have been discussed among all the authors.

- **Xiaoting Shao**, Zhongjie Yu, Arseny Skryagin, Tjitze Rienstra, Matthias Thimm, Kristian Kersting. Modelling Multivariate Ranking Functions with Min-Sum Networks. *International Conference on Scalable Uncertainty Management*, 281-288, 2020.

  I was involved in developing the ideas and writing this paper. The ideas and the content have been discussed among all the authors. I have implemented the core idea in this paper and ran the experiments.

- **Xiaoting Shao**, Arseny Skryagin, Wolfgang Stammer, Patrick Schramowski, Kristian Kersting. Right for Better Reasons: Training Differentiable Models by Constraining their Influence Function. *In Proceedings of Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

I was involved in developing the ideas and writing this paper. The ideas and the content have been discussed among both authors. I developed the code for implementing the idea used in this paper and ran the experiments.

- **Xiaoting Shao**, Tjitze Rienstra, Matthias Thimm, Kristian Kersting. Towards understanding and arguing with classifiers: Recent progress. *Datenbank-Spektrum 20* (2), 171-180, 2020.

  I was involved in writing this paper that gives an overview of our research progress.

- **Xiaoting Shao**, Kristian Kersting. Gradient-based Counterfactual Explanations using Tractable Probabilistic Models. 2022. Submitted.

  I have contributed to the ideas of the paper. The ideas and the content have been discussed among both authors. I developed the code for implementing the idea used in this paper and ran the experiments.

- **Xiaoting Shao**, Karl Stelzner, Kristian Kersting. Right for the Right Latent Factors: Debiasing Generative Models via Disentanglement. 2022. Submitted.

  I have contributed to the ideas of the paper. The ideas and the content have been discussed among all the authors. I developed the code for implementing the idea used in this paper and ran the experiments. The loss function was developed together with Karl Stelzner.

- Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, **Xiaoting Shao**, Hans-Georg Luigs, Anne-Katrin Mahlein, Kristian Kersting. Making Deep Neural Networks Right for the Right Scientific Reasons by Interacting with their Explanations. *Nature Machine Intelligence 2(8)*, 476-486, 2020.

  I ran the experiment on the PASCAL VOC 2007 dataset.

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. “Sanity Checks for Saliency Maps”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[2] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. “Debugging Tests for Model Explanations”. In: *Proceedings of the 34th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[3] Gediminas Adomavicius and Alexander Tuzhilin. “Context-Aware Recommender Systems”. In: *Recommender systems handbook*. Springer, 2011, pp. 217–253.

[4] Babak Alipanahi, Andrew Delong, Matthew T. Weirauch, and Brendan J. Frey. “Predicting the Sequence Specificities of DNA-and RNA-Binding Proteins by Deep Learning”. In: *Nature Biotechnology* 33.8 (2015), pp. 831–838.

[5] David Alvarez-Melis and Tommi S. Jaakkola. “On the Robustness of Interpretability Methods”. In: *Proceedings of the 3rd ICML Workshop on Human Interpretability in Machine Learning*. 2018.

[6] Javier Antoran, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. “Getting a CLUE: A Method for Explaining Uncertainty Estimates”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021. URL: https://openreview.net/forum?id=XSLF1XFq5h.

[7] Daniel W. Apley and Jingyu Zhu. “Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.4 (2020), pp. 1059–1086.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, and Richard Benjamins. “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115.

[9] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. “On Pixel-Wise Explanations for Non-linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PlOS One* 10.7 (2015), e0130140.

[10] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. “How to Explain Individual Classification Decisions”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.

[11] Sara Beery, Grant Van Horn, and Pietro Perona. “Recognition in Terra Incognita”. In: *Proceedings of the 15th European Conference on Computer Vision (ECCV)*. 2018, pp. 456–473.

[12] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation Learning: A Review and New Perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828.

[13] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. “Explainable Machine Learning in Deployment”. In: *Proceedings of the 3rd Conference on Fairness, Accountability, and Transparency*. 2020, pp. 648–657.

[14] Or Biran and Courtenay Cotton. “Explanation and Justification in Machine Learning: A Survey”. In: *Proceedings of the IJCAI-17 Workshop on Explainable AI (XAI)*. Vol. 8. 1. 2017, pp. 8–13.

[15] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, and Jiakai Zhang. “End to End Learning for Self-Driving Cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[17] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. “Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.

[18] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. “Adversarial Patch”. In: *Machine Learning and Computer Security Workshop, Neural Information Processing Systems (NeurIPS)* (2017).

[19] Joy Buolamwini and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification”. In: *Proceedings of 1st the Conference on Fairness, Accountability and Transparency*. PMLR. 2018, pp. 77–91.

[20] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. “Understanding Disentangling in β-VAE”. In: *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[21] Ruth MJ Byrne. *The Rational Imagination: How People Create Alternatives to Reality*. MIT press, 2007.

[22] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. “E-SNLI: Natural Language Inference With Natural Language Explanations”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 9560–9572.

[23] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *Proceedings of the IEEE symposium on Security and Privacy*. IEEE. 2017, pp. 39–57.

[24] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. “Intelligible Models for Healthcare: Predicting Pneumonia Risk and Hospital 30-day Readmission”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 1721–1730.

[25] Augustin Cauchy. “Méthode générale pour la résolution des systemes d’équations simultanées”. In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[26] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. “Explaining Image Classifiers by Counterfactual Generation”. In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[27] Agisilaos Chartsias, Thomas Joyce, Giorgos Papanastasiou, Scott Semple, Michelle Williams, David E Newby, Rohan Dharmakumar, and Sotirios A Tsaftaris. “Disentangled Representation Learning in Cardiac Image Analysis”. In: *Medical Image Analysis* 58 (2019), p. 101535.

[28] Mark Chavira and Adnan Darwiche. “Compiling Bayesian Networks Using Variable Elimination”. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 2443. 2007.

[29] Junxiang Chen and Kayhan Batmanghelich. “Weakly Supervised Disentanglement by Pairwise Similarities”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2020).

[30] Ricky TQ Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. “Isolating Sources of Disentanglement in Variational Autoencoders”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 2610–2620.

[31] Arthur Choi, Ruocheng Wang, and Adnan Darwiche. “On the Relative Expressiveness of Bayesian and Neural Networks”. In: *International Journal of Approximate Reasoning* 113 (2019), pp. 303–323.

[32] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *Proceedings of the NIPS Workshop on Deep Learning*. 2014.

[33] Diarmaid Conaty, Denis D. Mauá, and Cassio P. De Campos. “Approximation Complexity of Maximum a Posteriori Inference in Sum-Product Networks”. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 2017.

[34] R. Dennis Cook and Sanford Weisberg. “Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression”. In: *Technometrics* 22.4 (1980), pp. 495–508.

[35] Nelson Cowan. “The Magical Mystery Four: How Is Working Memory Capacity Limited, and Why?” In: *Current Directions in Psychological Science* 19.1 (2010), pp. 51–57.

[36] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa A. Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. “Flexibly Fair Representation Learning by Disentanglement”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019.

[37] George Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[38] Piotr Dabkowski and Yarin Gal. “Real Time Image Saliency for Black Box Classifiers”. In: *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017.

[39] Adnan Darwiche. “A Differential Approach to Inference in Bayesian Networks”. In: *Journal of the ACM (JACM)* 50.3 (2003), pp. 280–305.

[40] Adnan Darwiche and Pierre Marquis. “A Knowledge Compilation Map”. In: *Journal of Artificial Intelligence Research* 17 (2002), pp. 229–264.

[41] Anupam Datta, Shayak Sen, and Yair Zick. “Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments With Learning Systems”. In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE. 2016, pp. 598–617.

[42] A. Philip Dawid. “Conditional Independence in Statistical Theory”. In: *JSTOR* (1979).

[43] Rina Dechter and Robert Mateescu. “AND/OR Search Spaces for Graphical Models”. In: *Artificial Intelligence* 171.2-3 (2007), pp. 73–106.

[44] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. “Maximum Likelihood From Incomplete Data via the EM Algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[45] Aaron Dennis and Dan Ventura. “Learning the Architecture of Sum-Product Networks Using Clustering on Variables”. In: *Proceedings of the 26th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Citeseer. 2012, pp. 2033–2041.

[46] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. “Explanations Based on the Missing: Towards Contrastive Explanations With Pertinent Negatives”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/c5ff2543b53f4cc0ad3819a36752467b-Paper.pdf.

[47] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. “Explanations Can Be Manipulated and Geometry Is to Blame”. In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS 2019)*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/bb836c01cdc9120a9c984c525e4b1a4a-Paper.pdf.

[48] Finale Doshi-Velez and Been Kim. “Towards a Rigorous Science of Interpretable Machine Learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[49] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. “Discriminative Unsupervised Feature Learning With Exemplar Convolutional Neural Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9 (2015), pp. 1734–1747.

[50] Michael Downs, Jonathan L. Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. “CRUDS: Counterfactual Recourse Using Disentangled Subspaces”. In: *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*. 2020.

[51] Cian Eastwood and Christopher KI Williams. “A Framework for the Quantitative Evaluation of Disentangled Representations”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[52] Max Ehrlich, Timothy J. Shields, Timur Almaev, and Mohamed R. Amer. “Facial Attributes Classification Using Multi-Task Representation Learning”. In: *Proceedings of the CVPR Workshops*. 2016, pp. 47–55.

[53] Kai Epstude and Neal J. Roese. “The Functional Theory of Counterfactual Thinking”. In: *Personality and Social Psychology Review* 12.2 (2008), pp. 168–192.

[54] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[55] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, and John C. Platt. “From Captions to Visual Concepts and Back”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1473–1482.

[56] Ruth C. Fong and Andrea Vedaldi. “Interpretable Explanations of Black Boxes by Meaningful Perturbation”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3429–3437.

[57] Alex A. Freitas. “Comprehensible Classification Models: A Position Paper”. In: *ACM SIGKDD Explorations Newsletter* 15.1 (2014), pp. 1–10.

[58] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *Annals of Statistics* (2001), pp. 1189–1232.

[59] Kenji Fukumizu, Francis R. Bach, and Michael I. Jordan. “Dimensionality Reduction for Supervised Learning With Reproducing Kernel Hilbert Spaces”. In: *Journal of Machine Learning Research* 5.Jan (2004), pp. 73–99.

[60] Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. “Kernel Measures of Conditional Dependence”. In: *Proceedings of the 22nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2008, pp. 489–496.

[61] Andrea Galassi, Kristian Kersting, Marco Lippi, Xiaoting Shao, and Paolo Torroni. “Neural-Symbolic Argumentation Mining: An Argument in Favor of Deep Learning and Reasoning”. In: *Frontiers in big Data* 2 (2020), p. 52.

[62] Robert Gens and Pedro Domingos. “Discriminative Learning of Sum-Product Networks”. In: *Proceedings of the 26th Conference on Advances in Neural Information Processing Systems (NeurIPS 2012)*. Vol. 25. 2012, pp. 3239–3247.

[63] Robert Gens and Domingos Pedro. “Learning the Structure of Sum-Product Networks”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2013, pp. 873–880.

[64] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. “MADE: Masked Autoencoder for Distribution Estimation”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. PMLR. 2015, pp. 881–889.

[65] *German Credit Dataset*. `https://archive.ics.uci.edu/ml/support/statlog+(german+credit+data)`. Accessed: 2021.

[66] Leilani H Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. “Explaining Explanations: An Overview of Interpretability of Machine Learning”. In: *Proceedings of IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[67] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[68] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[69] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[70] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative Adversarial Networks”. In: *Proceedings of the 28th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 27. 2014.

[71] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 2015.

[72] Bryce Goodman and Seth Flaxman. “European Union Regulations on Algorithmic Decision-Making and a “Right to Explanation””. In: *AI Magazine* 38.3 (2017), pp. 50–57.

[73] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. “Counterfactual Visual Explanations”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 2376–2384.

[74] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech Recognition With Deep Recurrent Neural Networks”. In: *Proceedings of the 38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2013, pp. 6645–6649.

[75] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[76] Brandon M. Greenwell, Bradley C. Boehmke, and Andrew J. McCarthy. “A Simple and Effective Model-Based Variable Importance Measure”. In: *arXiv preprint arXiv:1805.04755* (2018).

[77] Aditya Grover, Kristy Choi, Rui Shu, and Stefano Ermon. “Fair Generative Modeling via Weak Supervision”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020.

[78] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. “XAI — Explainable Artificial Intelligence”. In: *Science Robotics* 4.37 (2019), eaay7120.

[79] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990. ISBN: 9781107720039 1107720036 9781107049994 1107049997.

[80] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[81] Yotam Hechtlinger. “Interpretation of Prediction Models Using the Input Gradient”. In: *arXiv preprint arXiv:1611.07634* (2016).

[82] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. “Dependency Networks for Inference, Collaborative Filtering, and Data Visualization”. In: *Journal of Machine Learning Research* 1.Oct (2000), pp. 49–75.

[83] Fritz Heider and Marianne Simmel. “An Experimental Study of Apparent Behavior”. In: *The American Journal of Psychology* 57.2 (1944), pp. 243–259.

[84] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. “Explaining Collaborative Filtering Recommendations”. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. 2000, pp. 241–250.

[85] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. “Beta-VAE: Learning Basic Visual Concepts With a Constrained Variational Framework”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

[86] Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P. Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. “Darla: Improving Zero-Shot Transfer in Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.

[87] Denis J. Hilton. “Conversational Processes and Causal Explanation”. In: *Psychological Bulletin* 107.1 (1990), p. 65.

[88] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[89] Robert R. Hoffman, Matthew Johnson, Jeffrey M. Bradshaw, and Al Underbrink. “Trust in Automation”. In: *IEEE Intelligent Systems* 28.1 (2013), pp. 84–88.

[90] Robert C. Holte. “Very Simple Classification Rules Perform Well on Most Commonly Used Datasets”. In: *Machine Learning* 11.1 (1993), pp. 63–90.

[91] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. “A Benchmark for Interpretability Methods in Deep Neural Networks”. In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 9737–9748. URL: http://papers.nips.cc/paper/9167-a-benchmark-for-interpretability-methods-in-deep-neural-networks.pdf.

[92] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[93] Haruo Hosoya. “Group-Based Learning of Disentangled Representations With Generalizability for Novel Contents”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 2019, pp. 2506–2513.

[94] Ronald A. Howard and James E. Matheson. “Influence Diagrams”. In: *Decision Analysis* 2.3 (2005), pp. 127–143.

[95] C. Ide, F. Hadiji, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting, and C. Wietfeld. “LTE Connectivity and Vehicular Traffic Prediction Based on Machine Learning Approaches”. In: *Proceedings of Vehicular Technology Conference*. IEEE. 2015.

[96] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. “Adversarial Examples Are Not Bugs, They Are Features”. In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS 2019)*. 2019.

[97] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. “Variational Autoencoder With Arbitrary Conditioning”. In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[98] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization With Gumbel-Softmax”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

[99] Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. “What Is the Best Multi-Stage Architecture for Object Recognition?” In: *Proceedings of the 12th International Conference on Computer Vision*. IEEE. 2009, pp. 2146–2153.

[100] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. “Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems”. In: *arXiv preprint arXiv:1907.09615* (2019).

[101] Kshitij Judah. “Active Imitation Learning via Reduction to IID Active Learning”. In: *Proceedings of the AAAI Fall Symposium Series*. 2012.

[102] Javed Khan, Jun S. Wei, Markus Ringner, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, and Carsten Peterson. “Classification and Diagnostic Prediction of Cancers Using Gene Expression Profiling and Artificial Neural Networks”. In: *Nature Medicine* 7.6 (2001), pp. 673–679.

[103] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. “Variational Autoencoders and Nonlinear ICA: A Unifying Framework”. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020, pp. 2207–2217.

[104] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, and Fernanda Viegas. “Interpretability Beyond Feature Attribution: Quantitative Testing With Concept Activation Vectors (TCAV)”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 2668–2677.

[105] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. “Learning Not to Learn: Training Deep Neural Networks With Biased Data”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9012–9020.

[106] Hyunjik Kim and Andriy Mnih. “Disentangling by Factorising”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018.

[107] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. “The (Un)Reliability of Saliency Methods”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 267–280.

[108] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. arXiv:1312.6114. 2014.

[109] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. “Probabilistic Sentential Decision Diagrams”. In: *Proceedings of the 14th International Conference on the Principles of Knowledge Representation and Reasoning*. 2014.

[110] Jack Klys, Jake Snell, and Richard Zemel. “Learning Latent Subspaces in Variational Autoencoders”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS 2018)*. Vol. 31. 2018.

[111] Pang Wei Koh and Percy Liang. “Understanding Black-Box Predictions via Influence Functions”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1885–1894.

[112] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[113] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet Classification With Deep Convolutional Neural Networks”. In: *Proceedings of the 26th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 25. 2012, pp. 1097–1105.

[114] Alex Kulesza and Ben Taskar. “Determinantal Point Processes for Machine Learning”. In: *arXiv preprint arXiv:1207.6083* (2012).

[115] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. “Variational Inference of Disentangled Latent Concepts From Unlabeled Observations”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[116] J. Lafferty, A. McCallum, and F. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the 18th International Conference on Machine Learning (ICML)*. 2001, pp. 282–289.

[117] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Unmasking Clever Hans Predictors and Assessing What Machines Really Learn”. In: *Nature Communications* 10.1 (2019), p. 1096.

[118] Yann LeCun. “Generalization and Network Design Strategies”. In: *Connectionism in Perspective* 19 (1989), pp. 143–155.

[119] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[120] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST Handwritten Digit Database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[121] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. “Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 3744–3753.

[122] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. “Visualizing and Understanding Neural Models in NLP”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 681–691. DOI: 10.18653/v1/N16-1082. URL: https://aclanthology.org/N16-1082.

[123] Y. Liang and G. Van den Broeck. “Learning Logistic Circuits”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[124] B. Lindsay, R. Pilla, and P. Basak. “Moment-Based Approximations of Distributions Using Mixtures: Theory and Applications”. In: *Annals of ISM* (2000).

[125] Stan Lipovetsky and Michael Conklin. “Analysis of Regression in Game Theory Approach”. In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pp. 319–330.

[126] Peter Lipton. “Contrastive Explanation”. In: *Royal Institute of Philosophy Supplements* 27 (1990), pp. 247–266.

[127] Zachary C. Lipton. “The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery”. In: *Queue* 16.3 (2018), pp. 31–57.

[128] Francesco Locatello, Gabriele Abbati, Thomas Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. “On the Fairness of Disentangled Representations”. In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 14611–14624.

[129] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. “Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 4114–4124.

[130] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. “Weakly-Supervised Disentanglement Without Compromises”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020.

[131] Francesco Locatello, Michael Tschannen, Stefan Bauer, Gunnar Rätsch, Bernhard Schölkopf, and Olivier Bachem. “Disentangling Factors of Variation Using Few Labels”. In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[132] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. “Object-Centric Learning With Slot Attention”. In: *Proceedings of the 34th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2020, pp. 11525–11538.

[133] Tania Lombrozo. “The Structure and Function of Explanations”. In: *Trends in Cognitive Sciences* 10.10 (2006), pp. 464–470.

[134] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[135] Daniel Lowd and Pedro Domingos. “Learning Arithmetic Circuits”. In: *arXiv preprint arXiv:1206.3271* (2012).

[136] Xinghua Lu, Bin Zheng, Atulya Velivelli, and ChengXiang Zhai. “Enhancing Text Categorization With Semantic-Enriched Representation and Training Data Augmentation”. In: *Journal of the American Medical Informatics Association* 13.5 (2006), pp. 526–535.

[137] Scott Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[138] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. “Credit Card Fraud Detection Using Bayesian and Neural Networks”. In: *Proceedings of the 1st International NAISO Congress on Neuro Fuzzy Technologies*. 2002, pp. 261–270.

[139] Aravindh Mahendran and Andrea Vedaldi. “Visualizing Deep Convolutional Neural Networks Using Natural Pre-images”. In: *International Journal of Computer Vision* 120.3 (2016), pp. 233–255.

[140] Radu Marinescu and Rina Dechter. “AND/OR Branch-And-Bound for Graphical Models”. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*. Citeseer. 2005, pp. 224–229.

[141] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. “Disentangling Disentanglement in Variational Autoencoders”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 4402–4412.

[142] John McCarthy. “What is Artificial Intelligence?” 2004.

[143] Peter McCullagh. “Generalized Linear Models”. In: *EJOR* 16.3 (1984), pp. 285–292.

[144] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S. Corrado, and Ara Darzi. “International Evaluation of an AI System for Breast Cancer Screening”. In: *Nature* 577.7788 (2020), pp. 89–94.

[145] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. “PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2437–2445.

[146] George A. Miller. “The Magical Number Seven, Plus or minus Two: Some Limits on Our Capacity for Processing Information”. In: *Psychological Review* 63.2 (1956), p. 81.

[147] Tim Miller. “Explanation in Artificial Intelligence: Insights From the Social Sciences”. In: *Artificial intelligence* 267 (2019), pp. 1–38.

[148] Brent Mittelstadt, Chris Russell, and Sandra Wachter. “Explaining Explanations in AI”. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 2019, pp. 279–288.

[149] Alejandro Molina, Sriraam Natarajan, and Kristian Kersting. “Poisson Sum-Product Networks: A Deep Architecture for Tractable Multivariate Poisson Distributions”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[150] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. “Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[151] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. `https://christophm.github.io/interpretable-ml-book/`. 2019.

[152] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. “Explaining Nonlinear Classification Decisions With Deep Taylor Decomposition”. In: *Pattern Recognition* 65 (2017), pp. 211–222.

[153] Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. “The Role of Disentanglement in Generalisation”. In: *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. 2021.

[154] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. “On the Number of Linear Regions of Deep Neural Networks”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*. 2014, pp. 2924–2932.

[155] Todd K. Moon. “The Expectation-Maximization Algorithm”. In: *IEEE Signal Processing Magazine* 13.6 (1996), pp. 47–60.

[156] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. “Explaining Machine Learning Classifiers Through Diverse Counterfactual Explanations”. In: *Proceedings of the 3rd Conference on Fairness, Accountability, and Transparency*. 2020, pp. 607–617.

[157] W James Murdoch, Peter J. Liu, and Bin Yu. “Beyond Word Importance: Contextual Decomposition to Extract Interactions From Lstms”. In: *Proceedings of 6th International Conference on Learning Representations (ICLR)*. 2018.

[158] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[159] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010.

[160] Raymond S. Nickerson. “Confirmation Bias: A Ubiquitous Phenomenon in Many Guises”. In: *Review of General Psychology* 2.2 (1998), pp. 175–220.

[161] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. “Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations”. In: *Science* 366.6464 (2019), pp. 447–453.

[162] Matthew L. Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. “Counterfactual State Explanations for Reinforcement Learning Agents via Generative Deep Learning”. In: *Artificial Intelligence* 295 (2021), p. 103455.

[163] A. van den Oord, N. Kalchbrenner, L. Espeholt, and O. Vinyals. “Conditional Image Generation With PixelCNN Decoders”. In: *Proceedings of the 30th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[164] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016.

[165] Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and Arun K. Majumdar. “Credit Card Fraud Detection: A Fusion Approach Using Dempster — Shafer Theory and Bayesian Learning”. In: *Information Fusion* 10.4 (2009), pp. 354–363.

[166] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. “Learning Model-Agnostic Counterfactual Explanations for Tabular Data”. In: *Proceedings of the Web Conference 2020*. 2020, pp. 3126–3132.

[167] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Elsevier, 2014.

[168] Robert Peharz, Bernhard C. Geiger, and Franz Pernkopf. “Greedy Part-Wise Learning of Sum-Product Networks”. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 612–627.

[169] Robert Peharz, Robert Gens, and Pedro Domingos. “Learning Selective Sum-Product Networks”. In: *Proceedings of the ICML Workshop on Learning Tractable Probabilistic Models (LTPM)*. Vol. 32. 2014.

[170] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. “On the Latent Variable Interpretation in Sum-Product Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.10 (2016), pp. 2030–2044.

[171] Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. “On Theoretical Properties of Sum-Product Networks”. In: *Proceedings of The 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2015, pp. 744–752.

[172] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. “Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning”. In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*. Ed. by Ryan P. Adams and Vibhav Gogate. Vol. 115. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 334–344.

[173] Oskar Pfungst. *Clever Hans: (the Horse of Mr. von Osten.) A Contribution to Experimental Animal and Human Psychology*. Holt, Rinehart and Winston, 1911.

[174] Hoifung Poon and Pedro Domingos. “Sum-Product Networks: A New Deep Architecture”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE. 2011, pp. 689–690.

[175] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. “FACE: Feasible and Actionable Counterfactual Explanations”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 344–350.

[176] Marcelo OR Prates, Pedro H Avelar, and Luis C. Lamb. “Assessing Gender Bias in Machine Translation: A Case Study With Google Translate”. In: *Neural Computing and Applications* (2019), pp. 1–19.

[177] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[178] Lawrence R. Rabiner and Biing-Hwang Juang. “An Introduction to Hidden Markov Models”. In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16.

[179] T. Rahman, S. Jin, and V. Gogate. “Cutset Bayesian Networks: A New Representation for Learning Rao-Blackwellised Graphical Models”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 2019, pp. 5751–5757.

[180] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. “Cutset Networks: A Simple, Tractable, and Scalable Approach for Improving the Accuracy of Chow-LIU Trees”. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pp. 630–645.

[181] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. “Zero-Shot Text-To-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. PMLR. 2021, pp. 8821–8831.

[182] Rajesh Ranganath, Dustin Tran, and David Blei. “Hierarchical Variational Models”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 324–333.

[183] Pramila Rani, Changchun Liu, Nilanjan Sarkar, and Eric Vanman. “An Empirical Study of Machine Learning Techniques for Affect Recognition in Human — Robot Interaction”. In: *Pattern Analysis and Applications* 9.1 (2006), pp. 58–69.

[184] Carl Edward Rasmussen. “Gaussian Processes in Machine Learning”. In: *Proceedings of the Summer School on Machine Learning*. Springer. 2003, pp. 63–71.

[185] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. “Learning to Compose Domain-Specific Transformations for Data Augmentation”. In: *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. NIH Public Access, 2017, p. 3239.

[186] D. J. Rezende and S. Mohamed. “Variational Inference With Normalizing Flows”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 1530–1538.

[187] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 2014, pp. 1278–1286.

[188] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ““Why Should I Trust You?” Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[189] Karl Ridgeway and Michael C. Mozer. “Learning Deep Disentangled Embeddings With the F-Statistic Loss”. In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 185–194.

[190] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. “Interpretations Are Useful: Penalizing Explanations to Align Neural Networks With Prior Knowledge”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 8116–8126.

[191] Tjitze Rienstra, Matthias Thimm, Kristian Kersting, and Xiaoting Shao. “Independence and D-separation in Abstract Argumentation”. In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 17. 1. 2020, pp. 713–722.

[192] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M. Botvinick. “Cognitive Psychology for Deep Neural Networks: A Shape Bias Case Study”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 2940–2949.

[193] Neal J. Roese. “Counterfactual Thinking”. In: *Psychological Bulletin* 121.1 (1997), p. 133.

[194] Kohavi Ronny and Becker Barry. *UCI Machine Learning Repository*. 1996. URL: https://archive.ics.uci.edu/ml/datasets/adult.

[195] A. Rooshenas and D. Lowd. “Discriminative Structure Learning of Arithmetic Circuits”. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016.

[196] Ribana Roscher, Bastian Bohn, Marco F. Duarte, and Jochen Garcke. “Explainable Machine Learning for Scientific Insights and Discoveries”. In: *IEEE Access* 8 (2020), pp. 42200–42216.

[197] Andrew Ross and Finale Doshi-Velez. “Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[198] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. “Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence, (IJCAI)*. 2017, pp. 2662–2670. DOI: 10.24963/ijcai.2017/371.

[199] Cynthia Rudin. “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[200] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. “Regularization With Stochastic Transformations and Perturbations for Deep Semi-supervised Learning”. In: *Proceedings of the 30th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 29. 2016, pp. 1163–1171.

[201] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. “Evaluating the Visualization of What a Deep Neural Network Has Learned”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (2016), pp. 2660–2673.

[202] Bernhard Schölkopf. “Causality for Machine Learning”. In: *Probabilistic and Causal Inference: The Works of Judea Pearl*. 2022, pp. 765–804.

[203] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. “Making Deep Neural Networks Right for the Right Scientific Reasons by Interacting With Their Explanations”. In: *Nature Machine Intelligence* 2.8 (2020), pp. 476–486.

[204] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. “Quantum-Chemical Insights From Deep Tensor Neural Networks”. In: *Nature Communications* 8.1 (2017), pp. 1–8.

[205] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. “Grad-Cam: Visual Explanations From Deep Networks via Gradient-Based Localization”. In: *Proceedings of the IEEE International Conference On Computer Vision (ICCV)*. 2017, pp. 618–626.

[206] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. “Grad-Cam: Why Did You Say That?” In: *arXiv preprint arXiv:1611.07450* (2016).

[207] Ramprasaath R. Selvaraju, Stefan Lee, Yilin Shen, Hongxia Jin, Shalini Ghosh, Larry Heck, Dhruv Batra, and Devi Parikh. “Taking a HINT: Leveraging Explanations to Make Vision and Language Models More Grounded”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2591–2600.

[208] Burr Settles. “Active Learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–114.

[209] Xiaoting Shao and Kristian Kersting. “Gradient-Based Counterfactual Explanations Using Tractable Probabilistic Models”. In: *arXiv preprint arXiv:2205.07774* (2022).

[210] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. “Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures”. In: *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*. 2020.

[211] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. “Conditional Sum-Product Networks: Modular Probabilistic Circuits via Gate Functions”. In: *International Journal of Approximate Reasoning* 140 (2022), pp. 298–313.

[212] Xiaoting Shao, Arseny Skryagin, P Schramowski, W Stammer, and Kristian Kersting. “Right for Better Reasons: Training Differentiable Models by Constraining their Influence Function”. In: *Proceedings of 35th AAAI Conference on Artificial Intelligence (AAAI)*. 2021.

[213] Xiaoting Shao, Karl Stelzner, and Kristian Kersting. “Right for the Right Latent Factors: Debiasing Generative Models via Disentanglement”. In: *arXiv preprint arXiv:2202.00391* (2022).

[214] Xiaoting Shao, Zhongjie Yu, Arseny Skryagin, Tjitze Rienstra, Matthias Thimm, and Kristian Kersting. “Modelling Multivariate Ranking Functions With Min-Sum Networks”. In: *Proceedings of the 14th International Conference on Scalable Uncertainty Management*. Springer. 2020, pp. 281–288.

[215] O. Sharir and A. Shashua. “Sum-Product-Quotient Networks”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2017.

[216] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Quoc Le, G. Hinton, and J. Dean. “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-Of-Experts Layer”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017.

[217] Andy Shih and Stefano Ermon. “Probabilistic Circuits for Variational Inference in Discrete Graphical Models”. In: *Proceedings of the 34th Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 4635–4646.

[218] Pannaga Shivaswamy and Thorsten Joachims. “Coactive Learning”. In: *Journal of Artificial Intelligence Research* 53 (2015), pp. 1–40.

[219] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. “Not just a black box: Learning important features through propagating activation differences”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016.

[220] Rui Shu, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole. “Weakly Supervised Disentanglement With Guarantees”. In: *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. 2020.

[221] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[222] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 2015.

[223] Jeffry A. Simpson. “Psychological Foundations of Trust”. In: *Current Directions in Psychological Science* 16.5 (2007), pp. 264–268.

[224] Chandan Singh, W. James Murdoch, and Bin Yu. “Hierarchical Interpretations for Neural Network Predictions”. In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[225] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. “Disentanglement by Nonlinear ICA With General Incompressible-Flow Networks (GIN)”. In: *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. 2020.

[226] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. “Striving for Simplicity: The All Convolutional Net”. In: *Proceedings of the Workshop of the 3rd International Conference on Learning Representations (ICLR)*. 2015.

[227] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. “Credit Card Fraud Detection Using Hidden Markov Model”. In: *IEEE Transactions on Dependable and Secure Computing* 5.1 (2008), pp. 37–48.

[228] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. “Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[229] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. “Are Disentangled Representations Helpful for Abstract Visual Reasoning?” In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 14245–14258.

[230] Pierre Stock and Moustapha Cisse. “Convnets and ImageNet Beyond Accuracy: Understanding Mistakes and Uncovering Biases”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 498–512.

[231] Eric V. Strobl, Kun Zhang, and Shyam Visweswaran. “Approximate Kernel-Based Conditional Independence Tests for Fast Non-parametric Causal Discovery”. In: *Journal of Causal Inference* 7.1 (2019).

[232] Erik Štrumbelj and Igor Kononenko. “Explaining Prediction Models and Individual Predictions With Feature Contributions”. In: *Knowledge and Information Systems* 41.3 (2014), pp. 647–665.

[233] Irene Sturm, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. “Interpretable Deep Neural Networks for Single-Trial Eeg Classification”. In: *Journal of Neuroscience Methods* 274 (2016), pp. 141–145.

[234] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 3319–3328.

[235] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going Deeper With Convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

[236] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[237] Stefano Teso and Kristian Kersting. “Explanatory Interactive Machine Learning”. In: *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society*. AAAI. 2019.

[238] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. “A Deeper Look at Dataset Bias”. In: *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 37–55.

[239] Antonio Torralba and Alexei A. Efros. “Unbiased Look at Dataset Bias”. In: *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2011, pp. 1521–1528.

[240] M. Trapp, R. Peharz, C. Rasmussen, and F. Pernkopf. “Learning Deep Mixtures of Gaussian Process Experts Using Sum-Product Networks”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018.

[241] Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. “Bayesian Learning of Sum-Product Networks”. In: *Proceedings of the 33rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. 2019.

[242] Berk Ustun, Alexander Spangher, and Yang Liu. “Actionable Recourse in Linear Classification”. In: *Proceedings of the 2nd Conference on Fairness, Accountability, and Transparency*. 2019, pp. 10–19.

[243] G. Van den Broeck, N. Di Mauro, and A. Vergari. “Tractable Probabilistic Models: Representations, Algorithms, Learning, and Applications”. In: *Tutorial at UAI* (2019).

[244] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. “Visualizing and Understanding Sum-Product Networks”. In: *Machine Learning* 108.4 (2019), pp. 551–573.

[245] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. “Show and Tell: A Neural Image Caption Generator”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3156–3164.

[246] Paul Voigt and Axel Von dem Bussche. “The EU General Data Protection Regulation (GDPR)”. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10 (2017), p. 3152676.

[247] Sandra Wachter, Brent Mittelstadt, and Chris Russell. “Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR”. In: *Harv. JL & Tech.* 31 (2017), p. 841.

[248] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. “Phoneme Recognition Using Time-Delay Neural Networks”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339.

[249] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[250] Julia K. Winkler, Christine Fink, Ferdinand Toberer, Alexander Enk, Teresa Deinlein, Rainer Hofmann-Wellenhof, Luc Thomas, Aimilios Lallas, Andreas Blum, and Wilhelm Stolz. “Association Between Surgical Skin Markings in Dermoscopic Images and Diagnostic Performance of a Deep Learning Convolutional Neural Network for Melanoma Recognition”. In: *JAMA Dermatology* 155.10 (2019), pp. 1135–1141.

[251] Young Joo Yang and Chang Seok Bang. “Application of Artificial Intelligence in Gastroenterology”. In: *World journal of gastroenterology* 25.14 (2019), p. 1666.

[252] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. “Adversarial Examples: Attacks and Defenses for Deep Learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.9 (2019), pp. 2805–2824.

[253] Osmar R. Zaiane. “Building a Recommender Agent for E-learning Systems”. In: *Proceedings of the International Conference on Computers in Education*. IEEE. 2002, pp. 55–59.

[254] Omar Zaidan, Jason Eisner, and Christine Piatko. “Using “Annotator Rationales” to Improve Machine Learning for Text Categorization”. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2007, pp. 260–267.

[255] Han Zhao, Mazen Melibari, and Pascal Poupart. “On the Relationship Between Sum-Product Networks and Bayesian Networks”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. PMLR. 2015, pp. 116–124.

Acronyms

ABCSPNs Autoregressive Block-wise Conditional Sum-Product Networks. 77, 78

AI Artificial Intelligence. vii, 3

ALE Accumulated Local Effects. 20

BNs Bayesian Networks. 57

C-CHVAE Counterfactual Conditional Heterogeneous Autoencoder. 87

CAVs Concept Activation Vectors. 19

CCNs Conditional Cutset Networks. 58

CD Contextual Decomposition. 23

CDEP Contextual Decomposition Explanation Penalization. 23

CEM Contrastive Explanations Method. 18, 87, 92

CI Conditional Independence. 61

CLL Conditional Log-Likelihoods. 67, 68

CLUE Counterfactual Latent Uncertainty Explanations. 88

CNets Cutset Networks. 45

CNNs Convolutional Neural Networks. 15, 36, 39

CRFs Conditional Random Fields. 57, 58

CRUDS Counterfactual Recourse Using Disentangled Subspaces. 87

CSPNs Conditional Sum-Product Networks. 33, 42, 57, 138

CSVAE Conditional Subspace Variational Autoencoder. 87

DACs Discriminative Arithmetic Circuits. 58

DAG directed acyclic graphs. 35

DNNs Deep Neural Networks. vii

DTD Deep-Taylor Decomposition. 28

ELBO evidence lower bound. 41, 121

EM Expectation-Maximization. 49

FACE Feasible and Actionable Counterfactual Explanations. 87, 92

FGSM Fast Gradient Signed Method. 108

GANs Generative Adversarial Networks. 118

GD Gradient Descent. 49

GDPR General Data Protection Regulation. 4, 11

GLMs Generalized Linear Models. 61

GPs Gaussian Processes. 57, 58

Grad-CAM Gradient-weighted Class Activation Mapping. 15

HINT Human Importance-aware Network Tuning. 22

HMMs Hidden Markov Models. 58

HVPs Hessian-vector products. 14

IG Integrated Gradients. 28

LCs Logistic Circuits. 58

LIME Local Interpretable Model-agnostic Explanations. 13

LRP Layer-wise Relevance Propagation. 6

LSTM Long Short-Term Memory. 24, 37

LV Latent Variable. 75

MADE Masked Autoencoders. 52

MAP Maximum A Posteriori. 47

MDN Mixture Density Network. 68

MF Mean Field. 67

MIG Mutual Information Gap. 131

ML Machine Learning. vii, 3

MLE Maximum Likelihood Estimation. 38, 49, 120

MLPs Multi-Layer Perceptrons. 52

MoEs Mixture of Experts. 59, 62

MPE Most Probable Explanation. 85

MSE Mean Squared Error. 29

NeSy XIL Neuro-Symbolic Explanatory Interactive Machine Learning. 26

PCs Probabilistic Circuits. 33, 45, 75

PDFs Probability Density Functions. 46, 52

PGMs Probabilistic Graphical Models. 44, 75

PSDDs Probabilistic Sentential Decision Diagrams. 45

RAT-SPNs Random and Tensorized Sum-Product Networks. 51, 138

RBFs Radial Basis Functions. 62

RBR Right for Better Reasons. 103, 104

RCoT Randomized conditional Correlation Test. 59, 62

ReLU Rectified Linear Unit. 15, 36

RKHS Reproducing Kernel Hilbert Spaces. 62

RMSE Root Mean Squared Error. 66

RNNs Recurrent Neural Networks. 37

ROAR RemOve And Retrain. 30

RRR Right for Right Reasons. 21, 103

SGD Stochastic Gradient Descent. 63

SHAP SHapley Additive exPlanations. 16

SOP Structured Output Prediction. 58, 65

SPICE Sum-Product networks Interpretation by Counterfactual Examples. 86, 88, 138

SPNs Sum-Product Networks. 33, 43, 45, 57, 138

SPQNs Sum-Product-Quotient Networks. 58

SPVAEs Sum-Product Variational Autoencoders. 79

TCAVs Testing with Concept Activation Vectors. 19

VAEAC Variational Autoencoder with arbitrary conditioning. 88

VAEs Variational Autoencoders. 18, 40, 77, 78, 82, 87, 138

VQA Visual Question Answering. 23

XAI Explainable Artificial Intelligence. 11

XIL Explanatory Interactive Machine Learning. vii, 24, 72, 137, 139