On the Two-fold Role of Logic Constraints in Deep Learning

UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

On the Two-fold Role of Logic Constraints in Deep Learning

Gabriele Ciravegna

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

On the Two-fold Role of Logic Constraints in Deep Learning

Gabriele Ciravegna

Advisor:

Prof. Marco Gori

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Franco Scarselli, *Univerisità di Siena*
Prof. Alfonso Gerevini, *Univerisità di Brescia*
Prof. Fabrizio Riguzzi, *Univerisità di Ferrara*

XXXIII ciclo — October 2024

To Cinzia, Danilo and Federico,
My family, My safe harbour

And to Irene.
Of course to Irene.

Acknowledgments

This thesis is the final outcome of a process lasted three years, during which I had the luck and the honour to work with many wonderful people. Even though this thesis has a single author, I have always used "we" instead of "I" throughout the chapters, because this thesis would not have been the same without their contribution.

First I want to thank my advisor **Prof. Marco Gori**, who has been a constant source of motivation and inspiration. Many of the ideas presented in this thesis come from him. He first formulated deep learning as both a learning *from constraints* and a learning *of constraints* problems as reported in Chapter 3. He had the intuition of extracting constraints via an auxiliary network (see Chapter 6). He understood that the same network could explain its predictions (which eventually led to the *explainable-by-design* network in Chapter 7).

Without the meticulous guidance of **Prof. Stefano Melacci**, however, I wouldn't have been able to produce high-quality works. Whenever you ask him something, he always answers: "Well, I know little about it, ..."; and then he tells you everything about it. Also, his writing skills have been incredibly helpful in many of the works presented in this thesis, particularly those in Chapter 5, 6, and 7.

The mathematical and logic knowledge of **Francesco Giannini** have been also of crucial importance in several works, especially those in Chapters 6 and 7. Further, his climbing and movie expertise have cheered up many weekends.

I want deeply thank my host and member of the Supervisory Committee **Prof. Frederic Precioso**, from the Universitè Côte d'Azur for inviting me in France and for introducing me to the Active Learning problem (Chapter 4). I really hope that this productive collaboration is just at the beginning.

For the adversarial defence work in Chapter 5, I want to thank **Prof. Fabio Roli** and **Prof. Battista Biggio** from the University of Cagliari, which introduced us to the Adversarial Attack issues and helped us in devising a possible solution.

For working together from many years and being a friend from more, I thank **Pietro Barbiero** from the University of Cambridge. In these years we particularly collaborated for the work presented in Chapter 7, together with **Prof. Pietro Lio**.

I thank **Prof. Giansalvo Cirrincione** from the Université Picardie Jules Verne for being member of the Supervisory Committee and for the past and present collaborations together with **Vincenzo Randazzo** from the Politecnico di Torino.

I want to also thank the people from SaiLab for these joyful years, particularly **Matteo Tiezzi** for the incredible trip to the AAAI2020 conference, **Andrea Zugarini** for the many lunches and the many laughs together and **Vincenzo Laveglia** for introducing me to many wonderful places in Siena.

Finally, I thank **Simona Altamura** for helping me respecting all the duties on time and thanks also to the Regione Toscana for initially giving me the Pegaso grant.

Abstract

Deep Learning (DL) is a special class of Artificial Intelligence (AI) algorithms, studying the training of Deep Neural Networks (DNNs). Thanks to the modularity of their structure, these models are effective in a variety of problems ranging from computer vision to speech recognition. Particularly in the last few years, DL has achieved impressive results. Nonetheless, the excitement around the field may remain disappointed since there are still many open issues. In this thesis, we consider the learning *from constraints* framework. In this setting, learning is conceived as the problem of finding task functions while respecting a number of constraints representing the available knowledge. This setting allows considering different types of knowledge (including, but not exclusively, the supervisions) and mitigating some of the DL limits. DNN deployment, indeed, is still precluded in those contexts where manual labelling is expensive. Active Learning aims at solving this problem by requiring supervision only on few unlabelled samples. In this scenario, we propose to take consider domain knowledge. Indeed, the relationships among classes offer a way to spot incoherent predictions, i.e., predictions where the model may most likely need supervision. We develop a framework where first-order-logic knowledge is converted into constraints and their violation is checked as a guide for sample selection. Another DL limit is the fragility of DNNs when facing adversarial examples, carefully perturbed samples causing misclassifications at test time. As in the previous case, we propose to employ domain knowledge since it offers a natural guide to detect adversarial examples. Indeed, while the domain knowledge is fulfilled over the training data, the same does not hold true outside this distribution. Therefore, a constrained classifier can naturally reject predictions associated to incoherent predictions, i.e., in this case, adversarial examples.

While some relationships are known properties of the considered environments, DNNs can also autonomously develop new relation patterns. Therefore, we also propose a novel learning *of constraints* formulation which aims at understanding which logic constraints are present among the task functions. This also allow explaining DNNs, otherwise commonly considered black-box classifiers. Indeed, the lack of transparency is a major limit of DL, preventing its application in many safety-critical domains. In a first case, we propose a pair of neural networks, where one learns the relationships among the outputs of the other one, and provides First-Order Logic (FOL)-based descriptions. Different typologies of explanations are evaluated in distinct experiments, showing that the proposed approach discovers new knowledge and can improve the classifier performance. In a second case, we propose an end-to-end differentiable approach, extracting logic explanations from the same classifier. The method relies on an entropy-based layer which automatically identifies the most relevant concepts. This enables the distillation of concise logic explanations in several safety-critical domains, outperforming state-of-the-art white-box models.

Contents

Chapter 1

Contribution

In this thesis, I resume the most important works that I conducted with my supervisor and other colleagues during the PhD studies. The leitmotif is the idea of merging symbolic reasoning, generally represented by logic rules, with machine perception, usually a neural network. The creation of a hybrid model allows tackling some of the most critical issues of deep learning. Among several possible choices, in these works we followed the approach of representing First-order-Logic (FOL) rules as logic constraints. In this way, we can exploit available domain knowledge to enhance neural networks within the paradigm of learning *from constraints*. On the other side, when no knowledge is available, we can extract it from a deep neural network by analysing which logic constraints are satisfied, following a learning *of constraints* approach.

This thesis is organized as follows. In Chapter 2, we briefly summarize what is Deep Learning with its strengths, and most importantly, its issues. A formal background on how to combine it with logic constraints is given in Chapter 3. The following chapters focus on the learning *from constraints* approach and how it can be exploited to tackle two different deep learning issues. Chapter 4 presents a way to minimize the number of supervisions required to train a model, in the context of active learning. Chapter 5 describes how we can defend a neural network from adversarial attacks by enforcing logic constraints and checking their violation. In the second part of the thesis, we show two possible approaches to learning *of constraints*. The first focuses on explaining a given classifier by means of another network (see Chapter 6). The latter introduces a novel neural layer which allows building explainable-by-design networks (see Chapter 7). A resume of the experimental chapters is provided in Chapter 8 together with possible future works. Appendix A.1 and A.4 contain extra results of the experiments. Finally, a complete list of the works produced during the doctorate is available in Appendix B. These have been published (or submitted) in journals like Neural Networks, IEEE TNNLS, Artificial Intelligence, Neurocomputing and IEEE TPAMI, or international conference such as IJCAI, AAAI, IJCNN and ECML.

Part I

Background

Chapter 2

Deep Learning: Strengths and Issues

In this chapter we introduce Deep Learning (DL), with its strengths and issues. In Section 2.1 we briefly cover Deep Neural Networks (DNNs) structure, the problems in which they are typically employed and the way they are trained. In Section 2.2, we report some of the most important achievements of DL, only considering 2020. Finally, in Section 2.3 we analyse the remaining limits of DL, focusing on those covered in this thesis.

2.1 Introduction to Deep Learning

Deep Neural Networks are a special type of machine learning classifiers composed of several layers of neurons. When equipped with a sigmoid activation function, each of these neurons can be regarded as a logistic regressor, whose output models the probability distribution of a certain event. Their computational capability is summed up in a deep neural network to predict very complex phenomena. In Figure 2.1, an example of a deep neural network is reported.

DNNs are commonly employed in different learning problems, ranging from supervised learning (both in classification and regression tasks) to unsupervised learning but also as generative models, by means of the adversarial learning paradigm. For instance, DNNs are capable to generate images of fictional persons that are undistinguishable from real ones[1]. In this thesis, we will mainly encounter classification tasks, therefore, in the following we will deepen this learning setting. Neural networks are trained through a stochastic optimization process, which iteratively compares the prediction of the network $f(x)$ with an available ground-truth label y. Their distance is usually computed by means of a Cross-Entropy Loss calculated either directly on the output of the network (in case of binary or multi-label classification), or after applying a softmax normalization (in case of mutually exclusive multi-class classification). At each iteration, the optimization process is gener-

[1] `https://thispersondoesnotexist.com/`

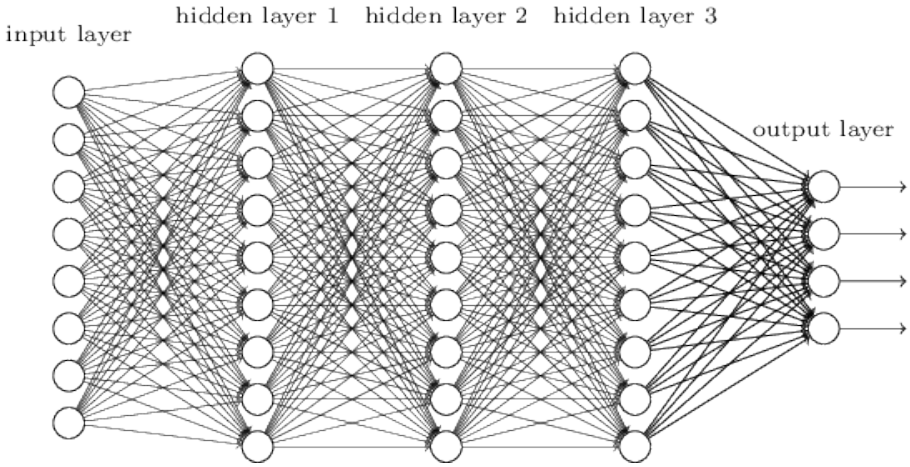


Figure 2.1: Deep Neural Network structure. Image from Nielsen (2015).

ally conducted on sub-portions of the whole training set (mini-batch) though the Stochastic Gradient Descent (SGD) algorithm (or a variant). The optimizer calculates the gradient of the loss for the single batch of data approximating the loss computed over all the training set[2]. The weights and the biases of the whole network are afterward updated by means of the Backpropagation algorithm, which computes the loss function gradient with respect to each neural network parameter.

Neural architectures may differ significantly according to the type of input data. Feed-Forward Networks (FFNs) are used when the input data is structured, i.e., when each feature has a meaning per se, and it can be stored in a precise field of a database. As an example we may consider the gender of a person or its age, in problems like loan allocation or insurance estimation. Different type of networks are instead required when we deal with unstructured data. Convolutional Neural Networks (CNNs) are commonly used when the input data are images in problems like face recognition or traffic sign detection. On the contrary, when sequential input data are given (such as voice sample or over-time records), we commonly employ Recurrent Neural Networks (RNNs), in problems ranging from speech understanding to spam detection and stock trading. The activation function of the output nodes mostly depends on the learning task. In classification, the most common activation is the sigmoid, which returns a $[0, 1]$ probability for each class. For what concern hidden neurons, instead, the most common activation function is the Rectified Linear Unit (ReLu), which has been proved to significantly reduce the vanishing gradient problem of deep neural networks (mainly due to the saturation of the previously employed functions, hyperbolic tangents and sigmoids).

[2]This is mainly done to reduce the computational burden required to store the gradients with respect to each parameter of a deep neural network. Just to give an idea, the commonly employed ResNet50 model has $\sim 2.3 \times 10^5$ parameters.

2.2 Deep Learning Achievements in 2020

By quoting the famous paper Marcus (2018) "In principle, given infinite data, deep learning systems are powerful enough to represent any finite deterministic mapping between any given set of inputs and a set of corresponding outputs". And every year we can see the extent of this statement, with remarkable progresses in very different fields. In the following, we will analyse some of the most important breakthroughs of Deep Learning, only considering the 2020.

In Natural Language Processing (NLP), the release of the Generative Pre-trained Transformer 3 (GPT-3) (Brown et al., 2020) significantly raised the level of machine natural language understanding. In order to get an idea of the capability of this model, it is sufficient to know that it is capable of writing a newspaper article[3], which is difficult to distinguish from one written by a human. To be more precise, the model has written an article talking about why humans should not fear robots, starting from a short introduction given by the Guardian which also edited the output. However, by quoting the editor's note: "Editing GPT-3's op-ed was no different to editing a human op-ed. We cut lines and paragraphs, and rearranged the order of them in some places. Overall, it took less time to edit than many human op-eds."

In the field of protein-structure prediction, DeepMind has developed AlphaFold (Jumper et al., 2021). This model is capable to determine the 3D shapes of proteins from its amino-acid sequence. The potentiality of this breakthrough is described in a Nature article,[4] stating: "The ability to accurately predict protein structures from their amino-acid sequence would be a huge boon to life sciences and medicine. It would vastly accelerate efforts to understand the building blocks of cells and enable quicker and more advanced drug discovery."

During the last year Covid-19 pandemic, Artificial Intelligence (AI) has demonstrated how it can detect efficiently COVID-19 pneumonia. In particular, in Harmon et al. (2020) the authors show how an AI model can be nearly as accurate as a physician, with up to 90 percent accuracy in detecting COVID-19 pneumonia from computed tomography (CT) scans. Even though the result may not seem surprising, the challenge of detecting COVID-19 pneumonia from this type of scans is not an easy task[5]. Indeed, the pneumonia needs to be both detected and distinguished from one related to influenza, a task in which also good physicians struggle.

Lastly, Ichnowski et al. (2020) demonstrated how the employment of DL methods significantly improve robots performances and how this may revolutionize warehouse environments. Last year, the rising popularity of online shopping has increased the demand for robots. Indeed, intelligent robots could help employees in fulfilling the orders in warehouse environments. And in this work, the authors show

[3] https://amp.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3.

[4] https://www.nature.com/articles/d41586-020-03348-4

[5] https://www.sciencedaily.com/releases/2020/09/200930144426.htm

how combining neural networks with motion planning software allows robots to grasps and move objects faster and more fluently. More precisely: "When applied to grasp-optimized motion planning, the results suggest that deep learning can reduce the computation time by two orders of magnitude (300×), from 29 s to 80 ms, making it practical for e-commerce warehouse picking."(Ichnowski et al., 2020).

2.3 The Limits of Deep Learning

However, notwithstanding the incredible achievements we have seen only during 2020, it may still be possible that the great excitement around DL may remain disappointed (as also suggested in Marcus (2018) and Sabour et al. (2017)). This is mainly due to the fact that the statement at the beginning of Section 2.2 is hardly ever valid in a real-world scenario. We live in a world with finite data in which DL models not only have to correctly interpolate training data, but more importantly they need to generalize to unseen data which may significantly differ. In Marcus (2018), the author identifies 10 DL issues which are strictly connected with this last statement, but also with the intrinsic nature of deep learning, which can be only thought as a mapping between some pre-defined input data and a set of categories. He states that DL is data hungry, it has limited capacity to transfer, it has no natural way to deal with hierarchical structure, it struggles with open-ended-inference, it is not sufficiently transparent, it is not well integrated with prior knowledge, it cannot distinguish causation from correlation, it presumes a stable world, it cannot be fully trusted (i.e., it is susceptible to adversarial attacks) and it is difficult to engineer with. Many of these issues are correlated, as for example the fact that DL presumes a stable world is one of the reason it is susceptible to adversarial attacks. Still, these are all valid issues and unless in the next few years the research community will be able to devise some solutions, DL will not be able to meet its expectations.

Starting from this assumption, in this thesis we focus on combining prio r knowledge with DL to overcome the data hungry and the adversarial attack issues and on extracting prior knowledge from DL to solve the transparency one. In the following, we will cover in more detail the issues that this work aims to mitigate.

2.3.1 DL is data hungry

In the last few years, most of the research in the DL field has focused on improving model performances, while little attention has been paid to reduce the huge amount of data required to train deep learning algorithms. As also shown in Figure 2.2, deep neural networks need many more labelled data to be effective and avoid the common over-fitting problem, when compared to standard machine learning algorithms and shallow neural networks.

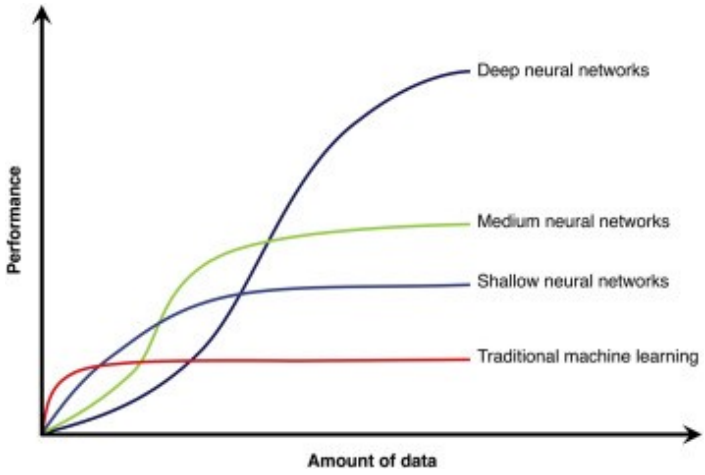


Figure 2.2: An illustration showing how DNNs are efficient only when large collection of labelled data are available. Figure from Agrawal (2019).

With the advent of Big Data, sample collection does not represent an issue any more. Nonetheless, the number of *supervised* data in some contexts is limited, and manual labelling can be expensive and time-consuming (Yu et al., 2015). Therefore, a common situation is the unlabelled pool scenario (McCallumzy and Nigamy, 1998), where many data are available, but only some are annotated. Historically, two strategies have been devised to tackle this situation: semi-supervised learning which focus on improving feature representations by processing the unlabelled data with unsupervised techniques (Zhu and Goldberg, 2009); active learning in which the training algorithm indicates which data should be annotated to improve the most its performances. Nowadays, the latter seems the most promising field in deep learning research to tackle the data hungry issue. In Chapter 4, we will see an example of an active learning strategy which exploit the available knowledge to select the data in the unlabelled pool on which the network is more mistaken.

2.3.2 DL is susceptible to adversarial attacks

Despite the impressive results reported in many application domains, neural networks have been shown to be easily misled by adversarial examples, i.e., carefully perturbed input samples that cause misclassifications at test time (Szegedy et al., 2014a; Biggio et al., 2013). In Figure 2.3, the very famous example of Szegedy et al. (2014a) is reported. In this, a network which confidently predicted a bus in the image (left), get completely fooled when imperceptible noise is added (centre and right), to the point that it classifies the bus as an ostrich.

In the last few years, a growing number of studies on the properties of adversarial attacks and of the corresponding defences have been produced by the scientific community (Papernot et al., 2016a; Goodfellow et al., 2018; Carlini et al., 2019)

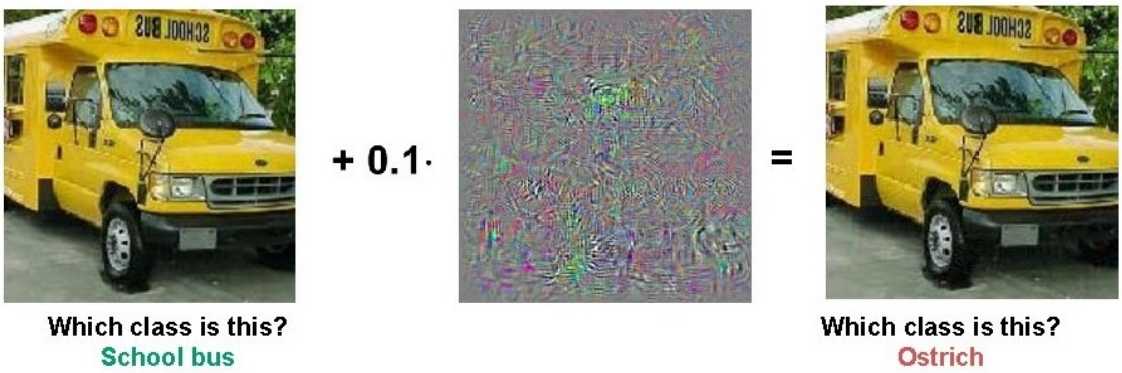


Figure 2.3: An example of an adversarial attack from Szegedy et al. (2014a).

(see Miller et al. (2020) for an in-depth review on this topic). Most of the existing approaches either propose methods for improving classifier robustness by modifying the learning algorithm to explicitly account for the presence of adversarial data perturbations (Goodfellow et al., 2015; Papernot et al., 2016b), or developing specific detection mechanisms for adversarial examples (Carlini and Wagner, 2017a; Ma et al., 2018; Miller et al., 2020). These approaches are developed against a specific class of attacks and usually are not robust against adversarial examples generated with different techniques (Araujo et al., 2020).

In Chapter 5 we report a method to make DNNs more robust to adversarial attacks by leveraging a given domain knowledge. More in detail, we will show how it is possible to detect adversarial examples at test time by checking logic constraint violation, with an almost null computational cost. We will also see how this defence is effective against a variety of attacks in the black-box context, and it is undermined only when we suppose that the attacker is equipped with the same domain-knowledge.

2.3.3 DL models are black-box

The lack of transparency in the decision process of some machine learning models, such as neural networks, limits their application in many safety-critical domains (EUGDPR, 2017). For this reason, eXplainable Artificial Intelligence (XAI) research has focused either on *explaining* black box decisions (Zilke et al., 2016; Ras et al., 2018; Ying et al., 2019) or on developing machine learning models *interpretable by design* (Breiman et al., 1984; Letham et al., 2015; Molnar, 2020). However, while interpretable models engender trust in their predictions (Doshi-Velez and Kim, 2017; Samek et al., 2020; Rudin et al., 2021), only black box models, such as neural networks, provide state-of-the-art task performances. Therefore, research to address this imbalance is urgently needed for the deployment of cutting-edge technologies.

Most of the techniques *explaining* black boxes focus on finding or ranking the

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 2.4: The explanation in terms of pixel importance of an image prediction with the LIME algorithm, as shown in Ribeiro et al. (2016).

most relevant features used by the model to make predictions (Simonyan et al., 2013; Ribeiro et al., 2016; Selvaraju et al., 2017). An example of this type of explanation is given in Figure 2.4, where the LIME algorithm proposed in Ribeiro et al. (2016) explain the predictions of a network in terms of the most important super-pixels for each classification.

Such feature-scoring methods are very efficient and widely used, but they cannot explain how neural networks compose such features to make predictions (Kindermans et al., 2019). Indeed, a key issue of most *explaining* methods is that the explanations are given in terms of input features (e.g., pixel intensities) that do not correspond to high-level categories that humans can easily understand (Kim et al., 2018b; Su et al., 2019). To overcome this issue, *concept-based* approaches have become increasingly popular as they provide explanations in terms of human-understandable categories (i.e., the *concepts*) rather than raw features (Kim et al., 2018b; Yeh et al., 2020; Koh et al., 2020). However, fewer approaches are able to explain how such concepts are leveraged by the classifier and even less provide concise logic explanations whose validity can be assessed quantitatively (Ribeiro et al., 2016; Guidotti et al., 2018a; Das and Rad, 2020).

In Chapters 6 and 7, we will see two different approaches to this type of problems. The first introduces an *explaining* method which show how the output concepts are correlated for a certain prediction and describes them through First-Order Logic (FOL) rules. The latter propose an *explainable-by-design* neural network, which is capable of explaining its own predictions in terms of the input concepts by means of a first layer assessing feature importance.

Chapter 3

Learning with Constraints

In this chapter we introduce the learning principles used in the thesis, aiming at combining logic constraints with neural networks. In Section 3.1, we first define the common notation. In Section 3.2, we cover the learning *from constraints* approach, which converts domain-knowledge in the form of mathematical constraints, and enables its employments in the learning problem. Finally, in Section 3.3 we introduce the learning *of constraints* approach, which aims at extracting FOL rules from a given neural network. Some principles reported in this chapter have been published in the work (Ciravegna et al., 2020b), which we presented at the AAAI 2020 conference together with some colleagues from the University of Siena.

In this thesis, we explore a learning paradigm simulating the behaviour of intelligent agents that interact in the environment in which they live. Agents acquire sensory information, and they interpret it by means of task functions. While some of the relationships among tasks are usually known proprieties of the considered environment, agents can also autonomously develop new relation patterns that are not known in advance. In this multi-task learning setting, we distinguish among the problem of learning task functions f in the *input space*, subject to given constraints ϕ, that implement the available knowledge on the problem at hand, and the problem of learning "new" constraints ψ in the *task space* to which the task functions belong. Both the already available constraints and the newly devised ones belong to what we refer to as the *rule space*, as sketched in Fig. 3.1 (a). Learning is the outcome of a developmental process in which our intelligent agents progressively learn task functions in the input space and, at the same time, they learn how the task functions are related in the task space.

Also, in this thesis we consider the generic framework of learning *from constraints* (Gnecco et al., 2015), that nicely generalizes the most popular learning settings by means of the unifying notion of "constraint" (see Table 4 in (Gnecco et al., 2015) for a list of examples). Learning is conceived as the problem of finding those task functions that are subject to a number of constraints that represent the available knowl-

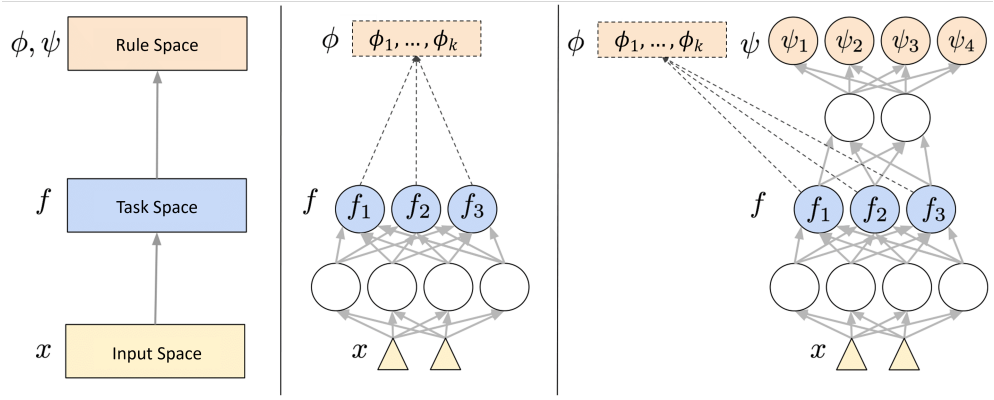


Figure 3.1: (a) Input, task, and rule spaces are associated to input data x, task functions $f_i(x)$, i=1...,r, and constraints $\phi_j(f(x)) = 0$, $j = 1, \ldots, k$, and $\psi_z(f(x)) = 0$, $z = 1, \ldots, m$, respectively. (b) Learning task functions *from constraints* $\phi_j(f(x)) = 0$. (c) Learning task functions while also learning *of constraints* $\psi_z(f(x)) = 0$, with $m = 4$.

edge on the considered problem (Fig. 3.1 (b)). The optimal solution is the one that is the most parsimonious (regular, smooth), making the problem well-posed. A key feature of such a framework is that symbolic knowledge, unambiguously expressed by means of First-Order Logic formulas, can be converted into constraints among the task functions and enforced on a subset or on all the training set (being it supervised or not) (Diligenti et al., 2017). Supervisions can also be represented as a simple form of pointwise constraint, i.e. $f_i(x_h) - y_h = 0$,[1] where (x_h, y_h) is a supervised pair and f_i is a certain task function.

While it is pretty common to search for solutions that are consistent with a given set of constraints (and some nice applications are reported in Chapter 4 and 5), learning new constraints to adapt to the environment is still an open challenge. Therefore, in this thesis, we also propose a novel approach to learning *of constraints* in the task space. The basic idea consists in understanding which logic constraints (further than those already available) are present among the task functions (Fig. 3.1 (c)). This can be done by minimizing a loss function (e.g., the Mutual Information), which transfer information from the task space to the rule space where another set of learnable functions live. Several possible loss functions and more information about how this can be practically obtained are reported in Chapter 6. Another approach consist in physically enumerating all the possible combinations of concepts (i.e., symbolic data or tasks). Even though it may seem unfeasible, in Chapter 7 we show that we can achieve it by means of an entropy-based input layer. This not only allows to explain another neural network, but also to create an *explainable-*

[1] Or also, employing the standard cross-entropy loss, $-log(f_i(x_h)) \cdot y_h = 0$.

by-design network when working on structured input data (i.e., data that can be already regarded as symbolic concepts). Both processes yield the discovering of new constraints among the tasks, that are fulfilled in different portions of the task space. Such newly devised knowledge is not immediately explainable using formal descriptions. For this reason, we need to devise a simple procedure that can explain the learned constraints in terms of FOL formulas, thus extracting symbolic knowledge on the relationships between the original tasks. This procedure is generally based on the idea of reducing the number of paths in the networks that implement the new constraints (e.g, by exploiting an L_1 regularization) and allows producing compact FOL formulas.

3.1 Notation

Formally, in this thesis we will consider a multi-task learning environment composed of r task learnable functions, $f_1, \ldots, f_r$, compactly represented by the vectorial function $f = [f_1, \ldots, f_r]$, where $f : X \subset \mathbb{R}^d \rightarrow Y \subset \mathbb{R}^n$. The perceptual information $x \in X$ is processed by the agent, and it is mapped to the task space by $f(x)$ (Fig 3.1 (b)). Each function f_i is responsible for implementing a specific task on the input domain X.[2] In a classification problem, the function f_i predicts the membership degree of x to the i-th class. Moreover, when we restrict the output of f_i to $[0, 1]$, we define $\mathbf{f_i}$ as the fuzzy logic predicate that models the truth degree of belonging to class i. The same also holds for structured features, i.e., when we map feature x^j in the $[0, 1]$ interval, we can define with $\mathbf{x^j}$ the logic predicate associated to it, determining the presence (or activation) of that feature.

By considering each logic predicate $\mathbf{f}^i$, First-Order Logic (FOL) becomes the natural way of describing relationships between data and classes or only among the classes. Let us illustrate this with an example of the first case,

$$\forall x \in X,\ \mathbf{x^1}(x) \wedge \mathbf{x^2}(x) \Rightarrow \mathbf{f}(x), \tag{3.1}$$

where $\mathbf{x^1}(x), \mathbf{x^2}(x)$ respectively represent the logic predicates associated to the first and second feature, and meaning that when both features are present also the output function $\mathbf{f}(x)$ needs to be true. Now, let us provide an example for the second case,

$$\forall x \in X,\ \mathbf{f_v}(x) \wedge \mathbf{f_z}(x) \Rightarrow \mathbf{f_u}(x), \tag{3.2}$$

for some v, z, u, meaning that the intersection between the v-th and the z-th class is always included in the u-th one. Finally, we also assume to have a discrete collection of n data $\mathcal{X} = \{x_1, \ldots, x_n \ : \ x_k \in \mathcal{X}\}$, of task-specific knowledge (i.e., supervisions y) on some of the available data, and of other knowledge that is about

[2] We could also easily extend it to the case in which task functions operate in different domains.

the environment in which the system operates, such as relationships among the task functions. For a complete list of the symbols and of the notation used in this chapter and throughout the thesis, please refer to Tab. 3.1.

3.2 Learning from Constraints

Several ad-hoc solutions can be considered to inject the available knowledge into the learning process that yield the f_i's. The framework of Learning from Constraints (Gnecco et al., 2015; Diligenti et al., 2017) follows the idea of converting domain knowledge into constraints on the learning problem and it studies, amongst a variety of other knowledge-oriented constraints (see, e.g., Table 2 in Gnecco et al. (2015)), the process of handling FOL formulas so that they can be both injected into the learning problem or used as a knowledge verification measure (Gori and Melacci, 2013). Such knowledge is implemented using a number of constraints that involve (a subset of) the logic predicates, and they are represented with[3]:

$$\phi_j(f(x)) = 0, \qquad j = 1, \ldots, c, \tag{3.3}$$

where $\phi = [\phi_1, \ldots, \phi_k]$, $\phi : Y \subset \mathbb{R}^r \rightarrow Z \subset \mathbb{R}^k$, being k the number of constraints.

Going into more detail, the FOL formulas representing the domain knowledge are converted into numerical constraints using the Triangular Norms (T-Norms, (Klement et al., 2013)). These binary functions generalize the conjunction operator $\wedge$ and offer a way to mathematically compute the satisfaction level of a given rule. Following the previous example, $\forall x \in X, \mathbf{x^1}(x) \wedge \mathbf{x^2}(x) \Rightarrow \mathbf{f}(x)$ is converted into a bilateral constraint that, by employing the product T-Norm, is:

$$1 - \mathbf{x^1}(x)\mathbf{x^2}(x)(1 - \mathbf{f}(x)) = 0 \tag{3.4}$$

It is pretty common to embed each ϕ_j into a non-negative penalty function that we indicate with $\hat{\phi}_j(f(x))$. In the simplest case (the one followed in this paper) such loss is $\hat{\phi}(f(x)) = 1 - \phi(f(x))$, which measures the level of satisfaction of the given constraints and has its minimum value in zero. Again, recalling the previous example, the associated loss function would be $\hat{\phi}_j(f(x)) = \mathbf{x^1}(x)\mathbf{x^2}(x)(1 - \mathbf{f}(x))$, which indeed is satisfied when either $\mathbf{x^1}(x)$ or $\mathbf{x^2}(x)$ are zeros or $\mathbf{f}(x)$ is approximately one. The quantifier $\forall x \in X$ is translated by enforcing the constraints on a discrete dataset of samples $\mathcal{X} \subset X$. The loss function $\varphi(f, \mathcal{K}, \mathcal{X})$ associated to all the available knowledge $\mathcal{K}$ is obtained by aggregating the losses of all the corresponding constraints in $\mathcal{X}$:

$$\varphi(f, \mathcal{K}, \mathcal{X}) = \sum_{\hat{\phi}_j \in \mathcal{K}} \sum_{x_k \in \mathcal{X}} \hat{\phi}_j(f(x_k)), \tag{3.5}$$

[3]Inequality constraints could be available as well, but we focus on equality constraints since inequality constrains $\phi_j(f(x)) \leq 0$ can be also converted into an equality by $\max\big(0, \phi_j(f(x))\big) = 0$.

As previously introduced, the loss function $\varphi(f, \mathcal{K}, \mathcal{X})$ can be employed as a knowledge verification measure in several contexts. In this thesis, it is used to find the test data on which the network is more mistaken in Chapter 4, and to detect adversarial examples in Chapter 5.

However, Eq. 3.5 can be also optimized during the learning problem, to train f over a subset of the input space. If $\mathcal{X}_{\phi_j} \subseteq X$ are the data points belonging to each subset for which some knowledge is available, then we aim at minimizing the penalties together with a parsimony index that enforces smoothness. Formally,

$$f^\star = \arg\min_f U(f, \mathcal{K}, \mathcal{X}) = \arg\min_f \left\{ \sum_{\hat{\phi}_j \in \mathcal{K}} \sum_{x_k \in \mathcal{X}_{\phi_j}} \hat{\phi}_j(f(x_k)) + \lambda_f \|f\| \right\}, \tag{3.6}$$

where the generic regularization term based on the norm of f is weighed by $\lambda_f > 0$. This formulation is particularly useful when we face a non fully-supervised learning problem, with a limited number of labelled samples $\mathcal{S} \subset \mathcal{X}$, defined as:

$$\mathcal{S} = \bigcup_i^{|\mathcal{K}_s|} \mathcal{X}_{\phi_i}, \tag{3.7}$$

where we indicate with $|\cdot|$ the cardinality operator and being $\mathcal{K}_s$ the knowledge associated to the supervisions. Indeed, we may also be given other knowledge K_f valid on all the input samples ($\bigcup_j^{|\mathcal{K}_f|} \mathcal{X}_{\phi_j} = \mathcal{X}$), regarding the relation of some input feature and the output classes or between the classes. Optimizing this second set of constraints allows the network to acquire information also over the unlabelled data, resulting in more robust and accurate networks, as we show in Chapter 5.

3.3 Learning of Constraints

Enforcing the *given* constraints $\phi_j(f(x)) = 0$, results in enforcing regularities in the task space that are dictated by ϕ. Learning *new* constraints $\psi_z(f(x)) = 0$, corresponds with discovering regularities in the task space, modelled by the vectorial function $\psi = [\psi_1, \ldots, \psi_m]$, with $\psi : Y \subset \mathbb{R}^r \to Z \subset \mathbb{R}^m$, where m is the number of learnable constraints. We propose to implement ψ using a feed-forward neural network with m output neurons, each of them associated to a component of ψ (Fig. 3.1 (c)). Such network is not the only element that the system is expected to learn, since also the subsets of data on which each constraint is enforced are unknown and must be estimated. We indicate with $\mathcal{X}_{\psi_z} \subseteq X$ the subset associated to ψ_z, and $\mathcal{X}_\psi = \{\mathcal{X}_{\psi_z},\ z = 1, \ldots, m\}$.

We can define a problem similar to Eq. 3.6, where the variables to be optimized are ψ and $\mathcal{X}_\psi$, that is,[4],

$$\psi^\star, \mathcal{X}_\psi{}^\star = \arg\min_{\psi, \mathcal{X}_\psi} D(\psi, \mathcal{X}_\psi, f) = \arg\min_{\psi, \mathcal{X}_\psi} \left\{ \sum_{z=1}^{m} \sum_{x_k \in \mathcal{X}_{\psi_z}} \hat{\psi}_z(f(x_k)) + \gamma_\psi \|\psi\| \right\}, \quad (3.8)$$

where $\gamma_\psi > 0$. This problem is clearly ill-posed, being it solved by ψ constantly equal to 0, or by m equivalent constraints. Moreover, we are posing no conditions on the portion of X covered by each $\mathcal{X}_\psi$, so that several not-useful solutions are possible.

In order to devise a valid formulation of Eq. 3.8, we have to optimize either an unsupervised or a supervised criterion that allow to precisely define both the constraints ψ and their supports $\mathcal{X}_\psi$. Indeed, the relations among the tasks may be unknown in advance, and we simply want to discover the most frequent co-occurrences. Among other possibilities, we follow the intuition that ψ should maximize the information transfer from the task space to the rule space, to capture the regularities of the task space itself. On the other side, we may want to extract specific relations of the tasks space, as for example the co-occurrences of some tasks with respect to a given one. In this case, a possible solution is to train a ψ model to exactly mimic the activation of a certain task. For further details on both approaches, please refer to Chapter 6.

The latter intuition is further analysed in Chapter 7. Indeed, if we are capable to explain the activation of a neural network w.r.t. a certain task (and we will better see how in the above-mentioned Chapter), we can also explain the prediction of a neural network w.r.t. the input data, in case they can be treated as symbolic concepts (i.e., when we are working with structure data). As in the previous case, this can be used to explain another classifier. However, it also paves the way to create an *explainable-by-design* neural network.

[4]In order to simplify the notation, here and in the rest of the thesis we will not make explicit the dependence of the objective function D on the data on which it is evaluated $\mathcal{X}$, but only on the support $\mathcal{X}_\psi$ it has to define.

Table 3.1: List of the main symbols used through out the thesis.

Notation	*Description*
$x \in X \subseteq \mathbb{R}^d$	Input sample living in a d-dimensional input space.
$\mathbf{x^j}(x)$	Value of the predicate associated to the j-th feature
$x_1, x_2, \ldots, x_n, x_k \in \mathcal{X}$	Single samples of the discrete set of samples $\mathcal{X}$
n	Number of samples in the considered dataset
$\mathcal{S}$	Set of supervised samples
$y \in Y \subseteq \mathbb{R}^r$	Labels associated to the samples $x \in \mathcal{S}$
$f_i(x)$	Membership score of x to the i-th class.
$\mathbf{f_i}(x)$	Fuzzy logic predicate associated to the i-th class.
r	Number of classes.
$\wedge$, $\vee$, $\neg$, $\Rightarrow$	Logical connectives.
$\mathcal{K}$	Domain knowledge.
$\phi(f(x)) \in \mathcal{Z} \subseteq \mathbb{R}^k$	Knowledge converted into constraint thorugh the T-Norm.
$\phi_s(f(x_h)) \in \mathcal{Z} \subseteq \mathbb{R}^k$	Pointwise constraint of a supervised sample $x_h \in \mathcal{S}$.
$\hat{\phi}(f(x))$	Penalty (≥ 0) associated to constraint $\phi(f(x)) = 0$.
$\varphi(f, \mathcal{K}, \mathcal{X})$	Loss function associated to a certain knowledge $\mathcal{K}and$.
k	Number of formulas in $\mathcal{K}$.
$\psi(f(x))$	Logic Constraints learnt from the task function f.
m	Number of learnable constraints.
$\mathcal{X}_{\phi_j}, \mathcal{X}_{\psi_z} \in X$	Support of a given constraint ϕ_j and of a learnt one ψ_z.

Part II

Exploiting Logic Constraints to Improve Deep Learning

Chapter 4

Active Learning by Logic Constraints

In this chapter, we show how logic constraints, derived from a given domain knowledge, can be exploited to minimize the number of training data required to train a neural network — i.e., to tackle the data-hungry issue introduced in Section 2.3.1. Some of the contents of this chapter are part of a work submitted to the ICLR 2022 conference together with Prof. Frederic Precioso from the Université Côte d'Azur.

In Section 4.1 the proposed active learning approach is introduced. Section 4.2 reports in details the proposed method, with first an example on inferring the XOR operation and then contextualized in a more realistic active learning domain. The experimental results on three datasets are described in Section 4.3, comparing the proposed technique with a standard active learning strategy. Finally, Section 4.4 resumes the related work.

4.1 Introduction

The main assumption behind active learning strategies is that there exists a subset of samples that allows to train a model with a similar accuracy as when fed with all training data. Iteratively, the model indicates the optimal samples to be annotated from the unlabelled pool. This is generally done by ranking the unlabelled samples w.r.t. a given measure and by selecting the samples associated to the highest scores. In this chapter, we propose an active learning strategy that compares the predictions over the unsupervised data with the available domain knowledge and exploits the inconsistencies as an index for selecting data to be annotated. To the best of our knowledge, however, domain-knowledge (converted in the form of logic constraints) violation has never been used as an index in the selection process of an active learning strategy.

A straightforward intuition in active learning is that the algorithm should select the data on which model predictions significantly differ from those produced on the training data. Uncertain sample selection follows this intuition by picking the

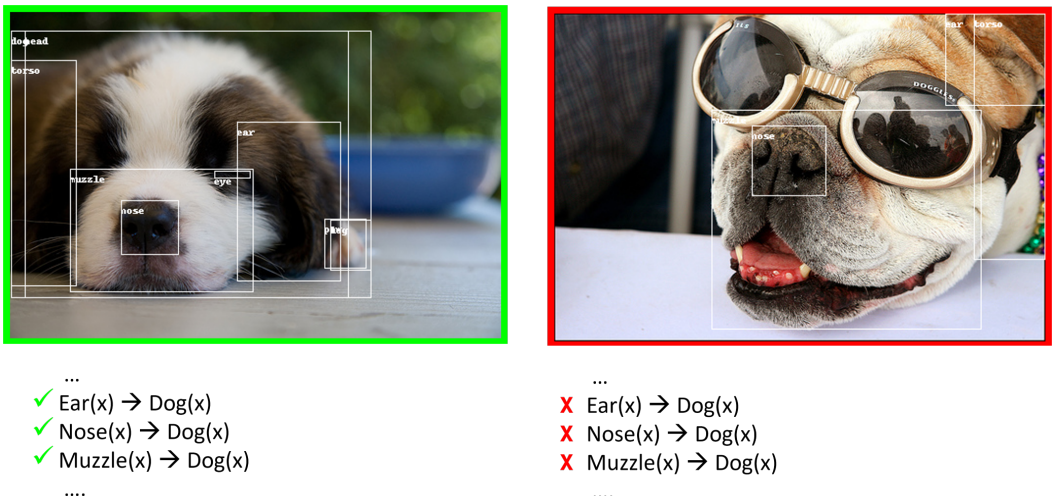


Figure 4.1: An example of how the proposed method works. Networks predictions on unseen data are compared with a given knowledge (in the form of FOL formulas). Knowledge violation is used as a metric to select samples (figure on the right) which require annotations in an active learning strategy. Images from the *PASCAL-Part* dataset (Chen et al., 2014).

points on which, e.g., the prediction entropy is high, in contrast to the entropy on the training set which is very low. We also follow this idea by taking into consideration the violation of a given knowledge on unseen data. Indeed, while the logic constraints derived from the domain knowledge are mostly satisfied on the training data, the same does not hold outside the training distribution. To give an example, let us consider the case of Dog image recognition (Figure 4.1). A model may have learnt on a set of dog images where dogs'muzzle were identifiable. However, on an image coming from an unseen distribution (e.g., a dog belonging to a different species), the model may still recognize the muzzle, but it may not recognize the dog (Figure 4.1, on the right). The proposed method would detect this image as requiring an annotation because it violates (among others) the logic constraint $\forall x : Muzzle(x) \Rightarrow Dog(x)$. A main assumption of this strategy is that the model does not only output the main object in every image, but it also recognizes some properties of the object or the parts it is composed of. In other words, we define the problem as multi-label classification. Please notice that this assumption does not limit the application to the standard image-classification scenario, since also object-detection can be regarded as a multi-label classification problem (Gong et al., 2019; Zhao et al., 2020).

In Section 4.3, we show that the proposed strategy outperforms the standard uncertain sample selection method, particularly in those contexts where domain-knowledge is rich. We empirically demonstrate that this is mainly due to the fact that the proposed strategy allows discovering data distributions lying far from training data, unlike uncertainty-based approaches. Neural networks, indeed, are known

to be over-confident of their prediction, and they are generally unable to recognize samples lying far from the training data distribution. This issue, beyond exposing them to adversarial attacks (Szegedy et al., 2014b; Goodfellow et al., 2014), prevents uncertainty-based strategies from detecting these samples as points that would require an annotation. On the contrary, even though a neural network may be confident of its predictions, the interaction between the predicted classes may still offer a way to spot out-of-the-distribution samples. As reported in the example above, a neural network may predict with high level of confidence the presence of the muzzle in the image and, as certainly, it may not recognize the dog. The missing interaction of the two classes, however, allows spotting an incoherent prediction. Finally, as anticipated above, the Knowledge-driven Active Learning (KAL) strategy can be also employed in the object-detection context where standard uncertainty-based ones are difficult to apply (Choi et al., 2021; Haussmann et al., 2020).

4.2 Knowledge-driven Active Learning

Following the notation introduced in Section 3.1, in this chapter, we focus on multi-label classification problems $f : X \rightarrow Y$, where $X \subseteq \mathbb{R}^d$ represents the feature space which may also comprehend non-structured data (e.g. input images) and $Y \subseteq \{0,1\}^r$ is the output space composed of $r \geq 1$ dimensions. When considering an object-detection problem, for a given class and a given image, we consider as class membership probability the maximum score value among all predicted bounding boxes around objects belonging to that class. Formally, $f_i(x_j) = \max_{b_h \in \mathcal{B}^i(x_j)} b_h(x_j)$ where $B^i(x_j)$ is the set of the confidence scores of the bounding boxes predicting the i-th class for sample x_j. Obviously, in case a certain class is not predicted in any of the bounding box, we set $f_i(x_j) = 0$. We also consider the case in which additional domain knowledge $\mathcal{K}_f$[1] is available for the problem at hand. How to exploit such knowledge to select the best points for training a classifier is the main subject of this section, and it follows the principles already introduced in Section 3.2. Using domain knowledge in the selection process provides precious information to define a criterion for identifying examples on which the model requires supervision.

Let us also consider the case in which there exists $\mathcal{S} \subset X$, representing the portion of input data already associated to an annotation $y_i \in \mathcal{Y} \subset Y$. We define with s the dimensionality of the starting set of labelled data. At each iteration, a set of p samples $\{x_1, x_2, ...x_p\} = \mathcal{X}_p \subset \mathcal{X}_u \subset X$ is selected by the active learning strategy to be annotated, being $\mathcal{X}_u$ the set of (still) unlabelled input data. This process is repeated for a number of iterations b, after which the training terminates. The maximum number of annotations therefore amount to $s + b * p$.

[1] In the context of this chapter we only consider the knowledge associated to the logic formulas $\mathcal{K}_f \subset \mathcal{K}$, excluding, e.g., the knowledge associated to pointwise constraints $\mathcal{K}_s$

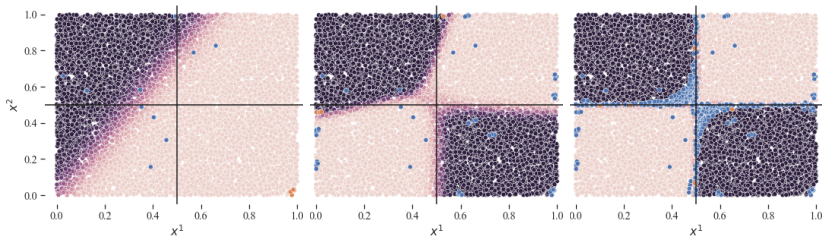


Figure 4.2: A visual example of the principles of the KAL strategy on the *XOR-like* problem. We depict network predictions with different colour degrees (light colours negative predictions, dark colours positive prediction). Also, we depict in orange the points selected in the current iteration, in blue those selected in previous iterations. From left, the scenario at the 1^{st}, 10^{th} and 100^{th} iteration.

In the proposed method, we consider the learning *from constraints* scenario defined in Section 3.2. Based on this, the proposed active learning strategy detect whether the predictions made by the model on out-of-sample data are coherent with the domain knowledge or not. More precisely, in this case Equation 3.5 is computed considering all the available FOL formulas $\mathcal{K}_f$ for the given problem to select the points which violate the most the constraints. In particular, this is done by aggregating the lossess of all the corresponding constraints for all the samples $x \in \mathcal{X}_u$:

$$\text{KAL}: \qquad x^\star = \arg\max_{x \in X_u} \sum_{k}^{\mathcal{K}} \hat{\phi}_k(f(x)) \tag{4.1}$$

4.2.1 A paradigmatic example: the *XOR-like* problem

A well-known problem in machine learning is the inference of the eXclusive OR (XOR) operation. To show the working principles of the proposed approach, we propose here a variant of this experiment, in which a neural network learns a *XOR-like* operation from a distribution of non-boolean samples. Specifically, we sampled 10000 points from the distribution $x \in [0,1]^2$, and we assigned a label $y(x)$ as following:

$$y(x) = \begin{cases} 1, & \text{if } x_1 > 0.5 \wedge x_2 \leq 0.5 \\ 1, & x_1 \leq 0.5 \wedge x_2 > 0.5, \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

Also, we can express the XOR operation through a FOL formula $(\mathbf{x_1} \wedge \neg \mathbf{x_2}) \vee (\neg \mathbf{x_1} \wedge \mathbf{x_2}) \Leftrightarrow \mathbf{f}$.

For the sake of clarity, here and in the following we drop the argument (x) of the logic predicates. As seen before, through the T-Norm operation we can convert the logic rule into a numerical constraint, and we can calculate its violation through the

loss functions:

$$\begin{aligned}\hat{\phi}_1 &= \mathbf{f}(1-\mathbf{x_1}(1-\mathbf{x_2}))(1-\mathbf{x_2}(1-\mathbf{x_2})),\\ \hat{\phi}_2 &= (1-\mathbf{f})(1-(1-\mathbf{x_1}(1-\mathbf{x_2}))(1-\mathbf{x_2}(1-\mathbf{x_2})))\end{aligned} \tag{4.3}$$

where $\hat{\phi}_1$ represents the violation of the rule $\mathbf{f} \rightarrow (\mathbf{x_1} \oplus \mathbf{x_2})$, while $\hat{\phi}_2$ represents the violation of $(\mathbf{x_1} \oplus \mathbf{x_2}) \rightarrow \mathbf{f}$, with $\oplus$ being logic XOR. For an automatic computation of the loss functions associated to the violation of a rule, please refer to Marra et al. (2019). The KAL strategy can therefore compute for each sample the associated loss (Eq. 4.3) and at each iteration select the points with higher violations (Eq. 4.1).

In Fig. 4.2, we reported an example of the proposed strategy starting from $n = 10$ labelled data and by selecting $p = 5$ points at each iteration for a total of $k = 100$ iterations. We depict network predictions with different colour degrees, from light beige (negative predictions) to black (positive predictions); we depict in orange the points selected at current iterations and in blue those selected previously. Notice how the proposed method immediately discovers novel data distributions by selecting samples from the right-bottom quadrant (orange points—figure on the left); after 10 iterations (figure at the centre) the network has mostly learnt the correct data distribution and later refines the predictions of the network sampling along the decision boundaries (blue points—figure on the right), allowing the network to completely solve the learning problem (accuracy $\sim 100\%$).

4.2.2 Real-life scenario: partial knowledge and different type of rules

It is clear that, in the case of the *XOR-like* problem, the knowledge is complete: if we compute the predictions directly through the rule, we already solve the learning problem. However, the purpose of that simple experiment is to show the potentiality of the proposed approach in integrating the available symbolic knowledge into a learning problem. In real-life scenarios, such a situation is unrealistic, but still we might have access to some useful knowledge that may allow solving more quickly a given learning problem.

More precisely, when we consider structured data (e.g., tabular data), we may know some very simple relations taking into consideration few features and the output class. This knowledge may not be sufficient to solve the learning problem, but a KAL strategy can still exploit it to drive the network to a fast convergence, as we will see in Sec. 4.3.

On the opposite, when we consider unstructured data (e.g., images or audio signals) the knowledge that we employ cannot directly rely on the input features. Nonetheless, if we consider a multi-label learning problem, we may know in advance some relations between the output classes. Let us consider, as an example,

a Dog-vs-Man classification: we might know that one of the main object (e.g., a dog) is composed of several parts (e.g., a muzzle, a body, a tail, four paws). A straightforward translation of this compositional property into a FOL rule might be $\mathbf{Dog} \Rightarrow \mathbf{Muzzle} \vee \mathbf{Body} \vee \mathbf{Tail} \vee \mathbf{Paws}$. Formulating the composition in the opposite way is also correct, i.e., from the parts to the main object (e.g., $\mathbf{Muzzle} \Rightarrow \mathbf{Dog}$).

Moreover, in all problems at least one of the classes needs to be predicted ($\mathbf{Dog} \vee \mathbf{Man}$), with main classes being mutually exclusive in standard multi-class problems ($\mathbf{Dog} \oplus \mathbf{Man}$). Finally, we can always incorporate an uncertainty-like rule requiring each predicate to be either true or false, $\mathbf{Dog} \oplus \neg\mathbf{Dog}$.

4.2.3 Adding diversity sampling

As anticipated, uncertainty-based methods in DL may not be very effective in case they are not paired with a diversity-sampling strategy. Also in the case of KAL this holds true: given a set of rules $\mathcal{K}$, the proposed method might select p samples all violating the same rule $\phi_k(f(x))$. Even though a neural network may need different samples to learn a novel distribution of data, picking a batch of samples belonging to the same distribution might be a poor strategy and slow down the overall training process. To avoid this issue, we select a maximum number r of samples violating a certain rule k, similarly to Brinker (2003) introducing diversity in Tong and Koller (2001) margin-based approach. Specifically, we group samples $x \in X_u$ according to the rule they violate the most, and we select a maximum number r of samples from each cluster (still following the ranking given by Eq. 4.1).

4.3 Experiments

In this work, we considered six different learning scenarios, comparing the proposed technique with other active learning strategies. We evaluated the proposed method on two standard machine learning problems, the inference of the *XOR-like* problem (already introduced in Section 4.2.1), and the classification of *IRIS* plants given their characteristics; on two image-classification tasks, the *ANIMALS* and the *CUB200* Wah et al. (2011a) datasets; and on two object-recognition tasks, the *DOGvsPERSON* and the *PASCAL-Part* Chen et al. (2014) datasets. The *DOGvsPERSON* dataset is a newly devised, publicly available dataset that we extracted from *PASCAL-Part* . For more details regarding this dataset, please refer to Appendix A.2.1. In Section 4.3.1 the compared methods are briefly described and analysed; in Section 4.3.2 a quantitative analysis of the different active learning strategies is reported for the six learning problems; in Section 4.3.3 a qualitative analysis on the *XOR-like* task is conducted; in Section 4.3.4 we compare the correlation of the losses associated to the UNCERTAIN and the KAL strategy with the SUPERVISED one. All the details regard-

ing each experimental problem as well as the type of knowledge are reported in the supplementary material in Appendix A.2.2. All the experiments have been repeated at least three times (ten in the two standard machine learning problems), starting from different weight initializations of the models. The code required to run the experiments is published on a publicly available repository[2]. Also, a simple code example is reported in Appendix A.3 showing how to solve the *XOR-like* problem following the KAL strategy.

4.3.1 Compared methods

We compared KAL to four active learning strategies: standard UNCERTAIN-sample selection, an enhanced version coupled with a diversity-sampling strategy (UNCERTAIN+), RANDOM selection, but also a SUPERVISED strategy, i.e., a utopian active learning strategy evaluating the supervision loss on the whole pool of the unlabelled samples. Obviously, this latter is an unfeasible active learning strategy because it would require to already have the labels of all the samples in the unlabelled pool. However, as the RANDOM selection can be regarded as a lower-bound, we can consider SUPERVISED as a possible upper-bound of the quality of an active learning strategy. Specifically, SUPERVISED selects the samples with the highest cross entropy loss $H(y_i, f_i)$:

$$\text{SUPERVISED}: \qquad x^\star = \underset{x \in X_u, y \in Y_u}{\arg\max} \sum_{i=1}^{c} H(f_i, y_i), \tag{4.4}$$

where Y_u is the set of labels associated to the samples X_u. Regarding the UNCERTAIN-sample selection, among other possibilities, we consider here the closest samples to the decision boundaries. Formally:

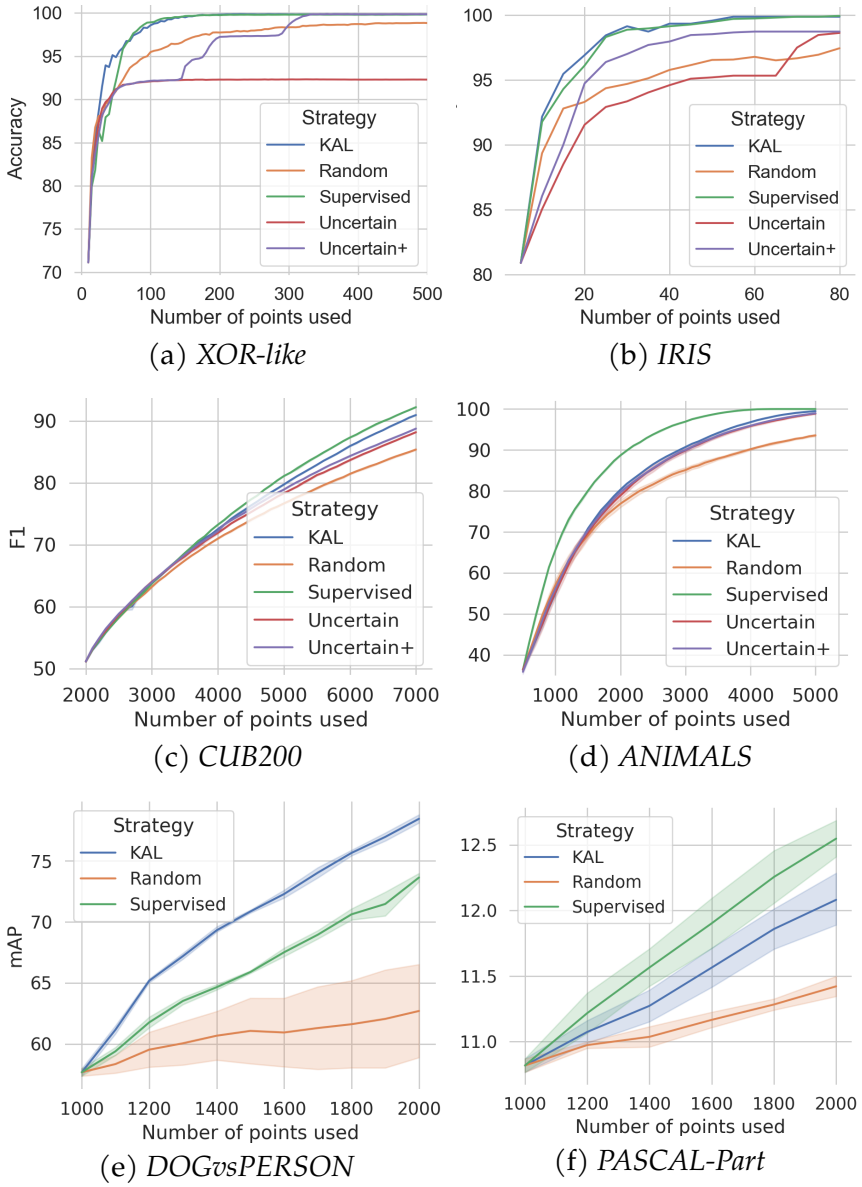
$$\text{UNCERTAIN}: \qquad x^\star = \underset{x \in X_u}{\arg\min}(\|f(x) - 0.5\|_1), \tag{4.5}$$

where an L_1 norm is employed to compute the distance from the decision boundary (0.5). Finally, UNCERTAIN+ is computed requiring that the samples selected with Eq. 4.5 belong to different clusters, computed by means of a K-Means on the input features (or on the featured extracted by the convolutional filters in the computer vision problems).

4.3.2 Experimental Results

For a quantitative comparison of the different methods, we evaluated the performance growth of a neural network when increasing the number of selected labelled data. In both the standard machine learning problems (Figures 4.3a, 4.3b), we can

[2]https://github.com/gabrieleciravegna/Constrained-Active-Learning



(a) *XOR-like* (b) *IRIS*

(c) *CUB200* (d) *ANIMALS*

(e) *DOGvsPERSON* (f) *PASCAL-Part*

Figure 4.3: Average performance growth on the six experiments when increasing the number of labelled samples. 75% confidence intervals not reported in the first two figures for better readability.

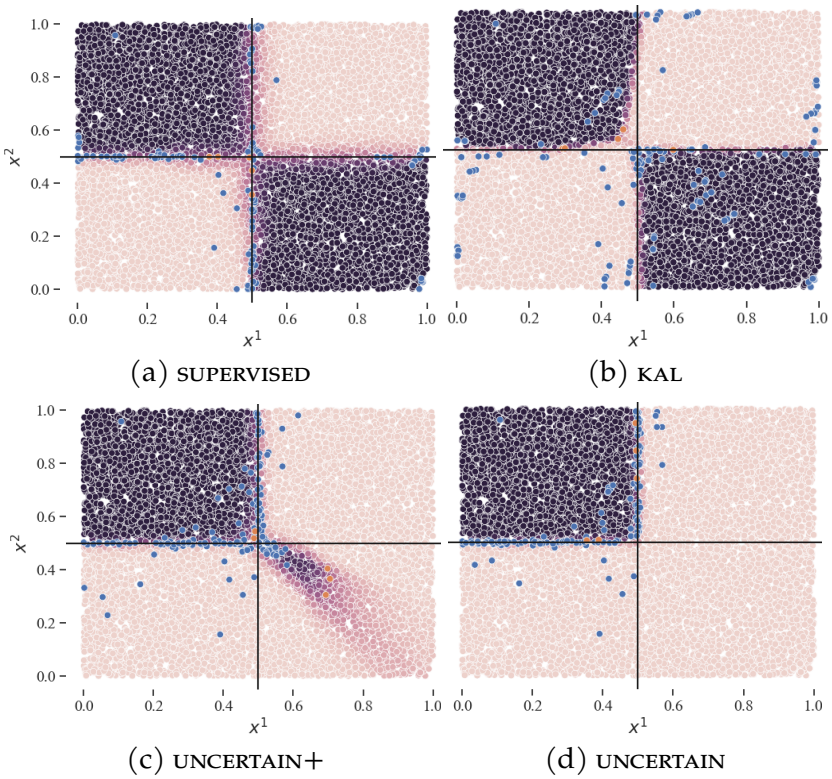
observe how the proposed method allows the network to solve the learning problem (i.e., reaching 100% accuracy) similarly to using a SUPERVISED strategy. While this was an expected behaviour on the *XOR-like* task since the provided rules completely explain the learning problem, on the *IRIS* classification task it is surprising since only 3 simple rules considering 2 features each are given (e.g., $\neg$**Long_Petal** $\Rightarrow$ **Setosa**). On the contrary, when using UNCERTAIN strategies, even when coupled with a diversity-based sampling, the network reaches the same level of performances much later (on *IRIS* when most of the dataset is labelled).

It is also worth noticing how, with RANDOM selection, the model does not reach 100 % accuracy in neither task, highlighting the need of an active learning strategy to perfectly solve the problems with these amounts of training data. A similar situation is also repeated on the image classification tasks *CUB200* and *ANIMALS* (Figures 4.3c, 4.3d). Here we notice the importance of employing well-structured knowledge. In the *ANIMALS* task, only 17 rules are employed relating 33 classes about animal species and their characteristics (e.g., $\mathbf{Fly} \Rightarrow \neg\mathbf{Penguin}$). In this case, the results with KAL are only slightly better w.r.t. using UNCERTAIN or UNCERTAIN+ strategies (mostly overlapped in the image classification problems). In the *CUB200* task instead, the performances are much better than when using UNCERTAIN strategies and closer to using the utopic SUPERVISED one. In *CUB200*, indeed, 311 rules are employed in the KAL strategy to help the model classify 308 classes considering bird species and their attributes (e.g., $\mathbf{White\_Pelican} \Rightarrow \mathbf{Black\_Eye} \vee \mathbf{Solid\_Belly\_Pattern} \vee \mathbf{Solid\_Wing\_Pattern}$). At last, in Figures 4.3e, 4.3f, we report the mean Average Precision (mAP) in two tasks of object recognition. In both cases the results of the proposed method are very good and, in the case of the *DOGvsPERSON* task, even better than using a SUPERVISED strategy. This result is unexpected and probably due to the fact that the supervision loss in object recognition is composed of several terms, some of which are inherently difficult to optimize. In the *PASCAL-Part* task, the overall performances are poor since it is a very difficult task (a Faster R-CNN trained on 90% of the dataset reaches a test mAP $\sim$ 13.94) with many small objects and object-parts to be recognized within the same image.

As anticipated, it is not straightforward applying UNCERTAIN (Eq. 4.5) sample selection to the object-detection context, thus, in this case, we restricted the comparison to the other methods only.

4.3.3 Qualitative Analysis

For a qualitative evaluation of the proposed approach, in Figure 4.4 we reported the samples selected by the different strategies at the 25^{th} iteration on the *XOR-like* task, starting from the same randomly selected samples of Figure 4.2. More figures showing the training prediction at different iterations are reported in Appendix A.2.3. When equipped with the KAL strategy and with this amount of training data, the network has already fully covered the data distribution in the right-bottom quadrant not represented by the starting samples. This behaviour mimics the one obtained when selecting the points through the SUPERVISED technique. On the contrary, when employing UNCERTAIN sample selection, novel data distributions are difficult to discover, leading to poorer performances. Even UNCERTAIN+, coupled with diversity sampling, at the 25^{th} iteration has only started discovering the novel data distribution. As shown in Figure 4.3a, indeed, UNCERTAIN+ requires more points on average to solve the learning problem.



(a) SUPERVISED (b) KAL

(c) UNCERTAIN+ (d) UNCERTAIN

Figure 4.4: A comparison of the sample selection process on the *XOR-like* task at the 25^{th} iteration (starting from the same points as in Figure 4.2). Notice how both uncertainty-based strategies have not covered yet the novel data distribution (right-bottom quadrant).

4.3.4 Correlation with Supervised Loss

We further investigated the advantage of using the KAL strategy w.r.t. the UNCERTAIN one, by analysing the correlation of the supervision loss with the knowledge-violation loss and with the uncertainty loss (respectively, arguments of Eqs. 4.4, 4.1, 4.5) on the *CUB200* problem. In Figure 4.5, we report two scatter plots comparing the two pairs of losses calculated over X_u at the 30^{th} training iteration. Samples selected by the strategies are depicted in orange, in grey the remaining samples in the unlabelled pool. The knowledge-violation loss is slightly more correlated with the SUPERVISED loss than the UNCERTAIN one, and the average supervision loss of the samples selected by the KAL strategy is higher (blue lines). Considering all iterations across all seeds on the *CUB200* task, the points selected by the KAL strategy have an average supervision loss of $\sim$ 154.31, while it is only of $\sim$ 117.9 for those selected by the UNCERTAIN one.

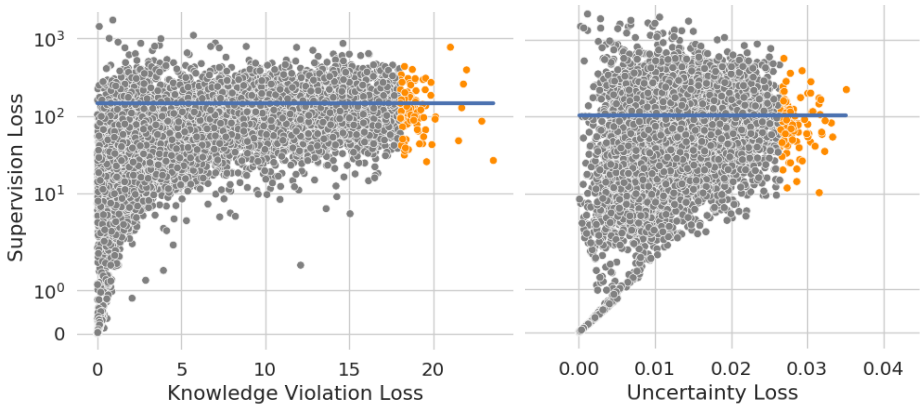


Figure 4.5: Correlation of the KAL (left) and the UNCERTAIN (right) losses (horizontal axes) with the SUPERVISED loss (vertical axis). In orange, the points selected by the active learning strategy. In blue, the average supervision loss of the selected points.

4.4 Related work

Devising an active learning strategy is not an easy task. As previously introduced, an approach could be to simply select the points on which the model is just wrong about. Since this is obviously not possible, in the literature, two main approaches have been followed: uncertainty sampling which select the data on which the model is the least confident, possibly extended with diversity sampling which maximizes the data distribution exploration among selected samples; curriculum learning which instead focuses first on easy samples then extending the training set to incorporate more and more difficult ones while also targeting more diversity. Standard uncertain strategies consist in choosing samples that maximize the prediction entropy (Houlsby et al., 2011; Cao and Tsang, 2021), the distance from the hyperplane in SVM (Schohn and Cohn, 2000), or the variation ratio in Query-by-committee with ensemble methods (Burbidge et al., 2007; Ducoffe and Precioso, 2017; Beluch et al., 2018). Establishing prediction uncertainty is more difficult with DL models. Indeed, they generally tend to be over-confident, particularly when employing softmax activation functions (Thulasidasan et al., 2019). Furthermore, there is no easy access to the distance to the decision boundary as for SVM, so it needs to be computed. This problem has been tackled by devising different uncertain strategies, such as employing Bayesian Neural Network with Monte Carlo Dropout (Gal et al., 2017), predicting the loss associated to each sample (Yoo and Kweon, 2019), or calculating the minimum distance required to create an adversarial example (Ducoffe and Precioso, 2018). As pointed out by Pop and Fulop (2018), however, uncertain strategy alone may choose the same categories many times and may create unbalanced datasets. In order to solve this, uncertain sample selection needs to be coupled with diversity sampling strategies. Diversity is generally obtained by preferring batches of data maximizing the mutual information between model parameters and predic-

tions (Kirsch et al., 2019), or core-set points (Sener and Savarese, 2018), or, also, samples nearest to k-means cluster centroids (Zhdanov, 2019). At last, by considering gradient parameters with respect to the predicted category, one can compute at the same time prediction uncertainty (by selecting samples with higher gradient norm) and sample diversity (by maximizing the diversity among the selected samples gradients) (Ash et al., 2019).

It has been pondered that human beings' cognition mainly consists in two different tasks: perceiving the world and reasoning over it (Solso et al., 2005). While in humans they take place at the same times, in artificial intelligence these two tasks are separately conducted by machine learning and logic programming. In the literature, there exists a variety of proposals aiming at joining these two fields (to create a so-called hybrid model), ranging from Statistical Relational Learning (SRL) (Koller et al., 2007) and Probabilistic Logic Programming (De Raedt and Kimmig, 2015) which focus on integrating learning with logic reasoning, to enhanced networks focusing on relations (Santoro et al., 2017) or with external memories (Graves et al., 2016). To the best of our knowledge, however, none of these methods can be directly applied in the standard active learning scenario. On the contrary, we have shown that the learning from constraints framework (Gnecco et al., 2015; Diligenti et al., 2017) can be naturally leveraged to devise active learning strategies in context where a domain-knowledge is available.

Chapter 5

Detecting Adversarial Attacks with Logic Constraints

In this chapter, we show how logic constraints, derived from a given domain knowledge, can be exploited to defend neural networks from adversarial attack issue introduced in Section 2.3.2). Some contents of this chapter are part of the work (Melacci et al., 2020) that we submitted to the IEEE TPAMI journal together with some colleagues from the University of Siena and Cagliari.

This chapter is organized as follows. In Section 5.1 we introduce the proposed adversarial defense approach. Section 5.2 deepen some learning *from constraints* principles (previously introduced in Section 3.2). Section 5.3 shows how domain knowledge can be used to defend against adversarial attacks, together with a knowledge-aware attack procedure. A detailed experimental analysis is reported in Section 5.4, evaluating the quality of our defense mechanisms, also considering state-of-the-art attacks and existing defense schemes. Finally, related works are resumed and compared to the proposed method in Section 5.5.

5.1 Introduction

In this Chapter, we focus on multi-label classification and, in particular, in the case in which domain knowledge on the relationships among the considered classes is available. which can be used to improve the classifier by enforcing FOL-based constraints on the unsupervised or partially labelled portions of the training set. A well-known intuition in adversarial machine learning suggests that a reliable model of the distribution of the data could be used to spot adversarial examples, being them not sampled from such distribution, but it is not a straightforward procedure (Grosse et al., 2017). We borrow such intuition, and we intersect it with the idea that semi-supervised examples can help learn decision boundaries that better follow the marginal data distribution, coherently with the available knowledge (Melacci

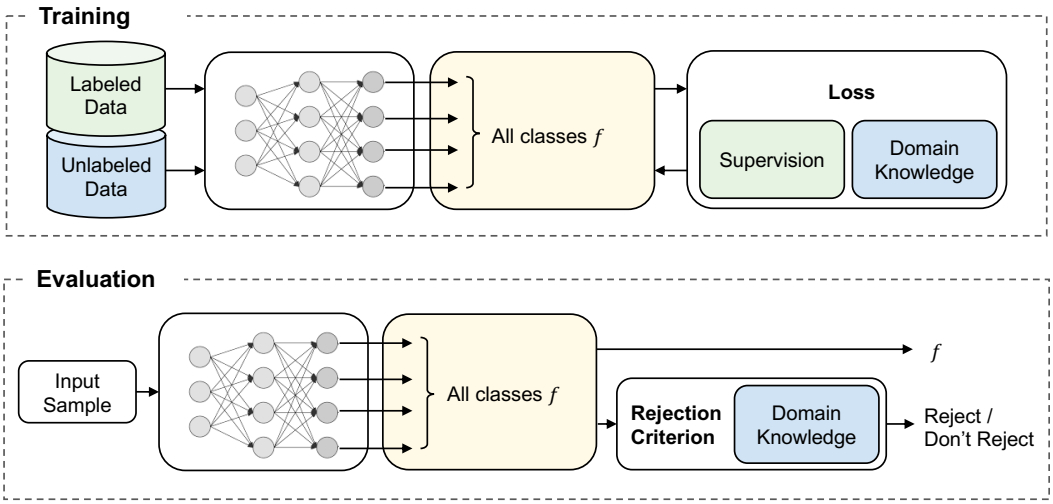


Figure 5.1: Leveraging domain knowledge to improve robustness of multi-label classifiers. At training time, domain knowledge is used to enforce constraints on the learning process using unlabeled or partially-labeled data. At evaluation time, domain-knowledge constraints are used to detect and reject samples outside of the training data distribution.

and Belkin, 2011; Diligenti et al., 2017), and we investigate the role of such knowledge in the context of data generated in an adversarial manner. While the generic idea of considering domain information in adversarial attacks has been recently followed by other authors to different extents (Naseer et al., 2019; Joshi et al., 2019; Sheatsley et al., 2021), to the best of our knowledge we are the first to use domain knowledge expressed in FOL and converted into polynomial constraints to improve adversarial robustness of multi-label classifiers.

In detail, this chapter contributes in showing that domain knowledge is a powerful feature (i) to improve robustness of multi-label classifiers, and (ii) to help detect adversarial examples. The underlying idea of our approach is conceptually represented in Fig. 5.1. At training time, domain-knowledge constraints are enforced on the unlabeled (or partially-labeled) data to learn decision boundaries which better align with the marginal distributions. At test time, the same constraints can be efficiently evaluated on the test samples to identify and reject incoherent predictions, ideally outside of the training data distribution, potentially including adversarial examples. Our approach can be also used in single-classification tasks where domain knowledge and auxiliary classes are present, and can be exploited internally by the classifier to implement the rejection mechanism based on domain-knowledge constraints. We will show some concrete examples of this latter setting in our experiments, reporting comparisons with state-of-the-art adversarial attacks and concurrent defenses developed for single-classification tasks.

To properly evaluate the robustness of our approach, which remains one of the

most challenging problems in adversarial machine learning (Carlini et al., 2019; Athalye et al., 2018; Biggio and Roli, 2018), we also propose a novel multi-label attack that can implement both black-box and white-box adaptive attacks, being driven by the domain knowledge in the latter case. While we show that an adaptive attack having access to the domain knowledge exploited by our classifier can bypass it, even though at the cost of an increased perturbation size, it remains an open issue to understand how hard for an attacker would be to infer such knowledge in practical cases. For this reason, we believe that our work can provide a significant contribution towards both evaluating and designing robust multi-label classifiers.

Indeed, most of the approaches in literature works in the fully supervised context. Only a few approaches leverage also unlabeled data to improve adversarial robustness (Miyato et al., 2016; Park et al., 2018; Akcay et al., 2018; Carmon et al., 2019; Miyato et al., 2018; Zhai et al., 2019; Najafi et al., 2019; Alayrac et al., 2019), although the semi-supervised learning setting provides a natural scenario for real-world applications in which labelling data is costly while unlabelled samples are readily available. More importantly, the case of multi-label classification, in which each sample can belong to more classes, is only preliminary discussed in the context of adversarial learning in (Song et al., 2018), while using adversarial examples to improve the accuracy on legitimate (non-adversarial) samples of some multi-label classifiers is studied in (Wu et al., 2017; Babbar and Schölkopf, 2018).

5.2 Learning with Domain Knowledge

Following the notation introduced in Section 3.1, in the context of this chapter we focus on multi-label classification problems with r classes, in which each input $x \in \mathcal{X}$ is associated to one or more classes. We consider the case in which additional domain knowledge $\mathcal{K}_f$ is available for problem at hand, represented by a set of relationships that are known to exist among (a subset of) the r classes. The introduction of domain knowledge in the learning process provides precious information only when the training data are not fully labeled, as in the classic semi-supervised framework. Some examples might be partially labeled (i.e., for each data points a subset of the r classes participates to the ground truth) or a portion of the training set might be unsupervised. Of course, if the data are fully labeled then all the class relationships are already encoded in the supervision signal. However, in this chapter we also consider domain knowledge as a mean to define a criterion that can spot potentially adversarial examples at test time, as we will discuss in Section 5.3, and that is also feasible in fully-supervised learning problems.

Learning from Constraints Following the learning *from constraints* principles introduced in Section 3.2, we better define here how constraints can be employed in a

learning problem and to detect adversarial examples. Indeed, we have seen how to compute the overall loss considering all the constraints in Equation 3.5. However, since we usually have k formulas whose relative importance could be uneven, with a little abuse of notation we define:

$$\varphi(f, \mathcal{K}, \mathcal{X}, \mu) = \frac{1}{|\mathcal{X}|} \sum_{\hat{\phi}_j \in \mathcal{K}} \sum_{x_h \in \mathcal{X}} \mu_j \hat{\phi}_j(f(x_h)) \in [0, \gamma], \tag{5.1}$$

with φ now also depending on μ which is the vector that collects the scalar weights $\mu_j > 0$ of the FOL formulas, while $\gamma = \sum_{j=1}^{k} \mu_j$.

We distinguish between the use of Eq. (5.1) as a loss function in the training stage and its use as a measure to evaluate the constraint fulfillment on out-of-sample data. In detail, the classifier is trained on the training set $\mathcal{L}$ by minimizing

$$f^\star = \arg\min_f \left\{ \varphi_s(f, \mathcal{K}_s, \mathcal{S}, \mu^{\mathcal{S}}) + \lambda \cdot \varphi_f(f, \mathcal{K}_f, \mathcal{L}, \mu^{\mathcal{L}}) \right\},^1 \tag{5.2}$$

where we explicitly distinguish the loss related to the pointwise-constraints φ_s derived from the supervised knowledge K_s and calculated on the supervised set $\mathcal{S}$, from the loss φ_f related to the knowledge given as FoL formulas $\mathcal{K}_f$ calculated on the overall training set $\mathcal{L}$. Furthermore, $\mu^{\mathcal{S}}$ is the importance of the pointwise constraints, $\mu^{\mathcal{L}}$ is the importance of the FOL formulas at training time, and $\lambda > 0$ modulates the weight of the constraint loss related to the formulas with respect to one related to the supervisions. Please notice that since we are in a semi-supervised setting (i.e., $\mathcal{S} \subset \mathcal{L}$) not all the training samples are labelled. The optimal λ is chosen by cross-validation, maximizing the classifier performance.

When the classifier is, instead, evaluated on a test sample $\overline{x}$, the measure

$$\varphi_f(f, \mathcal{K}_f, \{\overline{x}\}, \mu^{\mathcal{T}}) \in [0, \gamma^{\mathcal{T}}], \tag{5.3}$$

with weights $\mu^{\mathcal{T}}$ and $\gamma^{\mathcal{T}} = \sum_{j=1}^{k} \mu_j^{\mathcal{T}}$, returns a score that indicates the fulfillment of the domain knowledge $\mathcal{K}_f$ on $\overline{x}$ (the lower the better). Note that $\mu^{\mathcal{L}}$ and $\mu^{\mathcal{T}}$ might not necessarily be equivalent, even if certainly related. In particular, one may differently weigh the importance of some formulas during training to better accommodate the gradient-descent procedure and avoid bad local minima.

It is important to notice that Eq. (5.2) enforces domain knowledge only on the training data $\mathcal{L}$. There are no guarantees that such knowledge will be fulfilled in the whole input space $\mathcal{X}$. This suggests that optimizing Eq. (5.2) yields a stronger fulfillment of knowledge $\mathcal{K}_f$ over the space regions where the training points are distributed (low values of φ_f), while φ_f could return larger values when departing from the distribution of the training data. The constraint enforcement is soft, so that the second term in Eq. (5.2) is not necessarily zero at the end of the optimization.

[1] In order to simplify the notation, here and in the following we do not explicitly represent the dependency of f to its internal weights W.

5.3 Exploiting Domain Knowledge against Adversarial Attacks

The basic idea behind this chapter is that the constraint loss of Eq. (5.1) is not only useful to enforce domain knowledge into the learning problem, but also (*i*) to gain some robustness with respect to adversarial attacks and (*ii*) as a tool to detect adversarial examples at no additional training cost, that are, as anticipated, the main directions of this chapter.

A Paradigmatic Example. The example in Fig. 5.2 illustrates the main principles followed in this work, in a multi-label classification problem with 4 classes (cat, animal, motorbike, vehicle) for which the following domain knowledge is available, together with labeled and unlabeled training data:

$$\forall x, \text{CAT}(x) \Rightarrow \text{ANIMAL}(x)\,, \tag{5.4}$$
$$\forall x, \text{MOTORBIKE}(x) \Rightarrow \text{VEHICLE}(x)\,, \tag{5.5}$$
$$\forall x, \text{VEHICLE}(x) \Rightarrow \neg\text{ANIMAL}(x)\,, \tag{5.6}$$
$$\forall x, \text{CAT}(x) \vee \text{ANIMAL}(x) \vee \text{MOTORBIKE}(x) \vee \text{VEHICLE}(x). \tag{5.7}$$

Such knowledge is converted into numerical constraints, as described in Section 5.2, while the loss function φ_f is enforced on the training data predictions during classifier training (Eq. 5.2). Fig. 5.2 shows two examples of the learned classifier.

Considering point (*i*), in both cases, the decision boundaries are altered on the unlabeled data, enforcing the classifier to take a knowledge-coherent decision over the unlabeled training points and to better cover the marginal distribution of the data. This knowledge-driven regularity improves classifier robustness to adversarial attacks, as we will discuss in Section 5.4. Going into further details to illustrate claim (*ii*), in (a) we have the most likely case, in which decision boundaries are not always perfectly tight to the data distribution, and they might be not closed (ReLU networks typically return high-confidence predictions far from the training data (Hein et al., 2019)). Three different attacks are shown (purple). In attack 1, an example of motorbike is perturbed to become an element of the cat class, but Eq. (5.4) is not fulfilled anymore. In attack 2, an example of animal is attacked to avoid being predicted as animal. However, it falls in a region where no predictions are yielded, violating Eq. (5.7). Attack number 3 consists of an adversarial attack to create a fake cat that, however, is also predicted as vehicle, thus violating Eq. (5.4) and Eq. (5.6). In (b) we have an ideal and extreme case, with very tight and closed decision boundaries. Some classes are well separated, it is harder to generate adversarial examples by slightly perturbing the available data, while it is easy to fall in regions for which Eq. (5.7) is not fulfilled. The pictures in (c-d) show the unfeasible regions in which the constraint loss φ_f is significantly larger, thus offering a

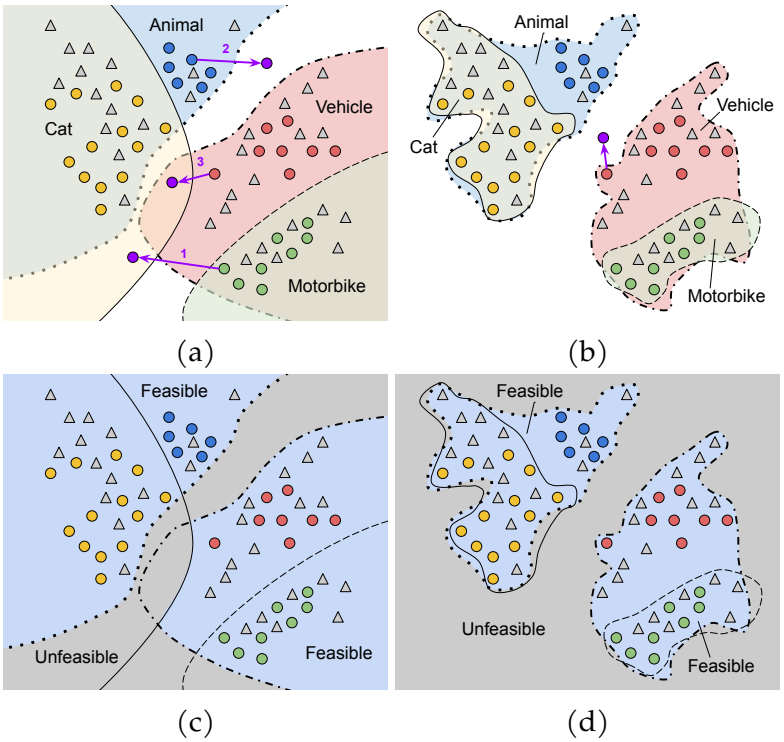


Figure 5.2: Toy example using the domain knowledge of Eqs. (5.4-5.7): cat (yellow), animal (blue), motorbike (green), vehicle (red). Labeled/unlabeled training data are depicted with coloured dots/gray triangles. (a,b) The decision regions in two sample outcomes of the training procedure: (a) open/loose decision boundaries; (b) tight/closed decision boundaries. White area are associated with no predictions. Adversarial examples (purple arrows/dots) are detected when they end up in regions that violate the constraints. Moreover, in (c,d) The feasible/unfeasible regions (blue/gray) that fulfill/violate the constraints for (a,b) are shown.

natural criterion to spot adversarial examples that fall outside of the training data distribution.

Domain Knowledge-based Rejection. Following these intuitions, and motivated by the approach of (Hendrycks and Gimpel, 2017; Hendrycks and Gimpel, 2017), we define a rejection criterion Ω as the Boolean expression

$$\Omega(\bar{x}, \tau | f, \mathcal{K}_f, \mu^{\mathcal{T}}) = \varphi_f(f, \mathcal{K}_f, \{\bar{x}\}, \mu^{\mathcal{T}}) > \tau \tag{5.8}$$

where $\tau > 0$ is estimated by cross-validation in order to avoid rejecting (or rejecting a small number of[2]) the examples in the validation set $\mathcal{V}$. Eq. (5.8) evaluates the constraint loss on the validation data $\mathcal{V}$, using the importance weights $\mu^{\mathcal{T}}$ (that we will discuss in what follows), as in Eq. (5.3). The rationale behind this idea is that those

[2]10% in our experiments.

samples for which the constraint loss is larger than what it is on the distribution of the data that are available when training/tuning the classifier, should be rejected. The training samples are the ones over which domain knowledge was enforced during the training stage, while the validation set represents data on which knowledge was not enforced, but that are sampled from the same distribution from which the training set is sampled, making them good candidates for estimating τ. Notice that Ω is measured at test time on an already trained classifier, and it can be used independently on the nature of the training data (fully or partially/semi-supervised). Differently from ad-hoc detectors, that usually require to train generative models, this rejection procedure comes at no additional training cost.[3]

Pairing Effect. The procedure is effective whenever the functions in f are not too strongly paired with respect to $\mathcal{K}_f$, and we formalize the notion of "pairing" as follows.

Definition 5.3.1. *Pairing. We consider a classification problem whose training data are distributed accordingly to the probability density $p(x)$. Given $\mathcal{K}_f$ and $\mu^{\mathcal{T}}$, the functions in f are strongly paired whenever $\zeta(\mathcal{H}, \mathcal{L}) = \|\varphi_f(f, \mathcal{K}_f, \mathcal{H}, \mu^{\mathcal{T}}) - \varphi_f(f, \mathcal{K}_f, \mathcal{L}, \mu^{\mathcal{T}})\| \approx 0$, being $\mathcal{H}$ a discrete set of samples uniformly distributed around the support of $p(x)$.*

This notion indicates that if the constraint loss is fulfilled in similar ways over the training data distribution and space areas close to it, then there is no room for detecting those examples that should be rejected. While it is not straightforward to evaluate pairing before training the classifier, the soft constraining scheme of Eq. (5.2) allows the classification functions to be paired in a less strong manner that what they would be when using hard constraints.[4] Note that a multi-label system is usually equipped with activation functions that do not structurally enforce any dependencies among the classes (e.g., differently from what happens with softmax), so it is naturally able to respond without assigning the input to any class (white areas in Fig. 5.2). This property has been recently discussed as a mean for gaining robustness to adversarial examples (Shafahi et al., 2019; Bendale and Boult, 2016). The formula in Eq. (5.7) is what allows our model to spot examples that might fall in this "I don't know" area. Dependencies among classes are only introduced by the constraint loss φ_f in Eq. (5.2) on the training data.

The choice of $\mu^{\mathcal{T}}$ is crucial in the definition of the reject function Ω. On the one hand, in some problems we might have access to the certainty degree of each FOL formula, that could be used to set $\mu^{\mathcal{T}}$, otherwise it seems natural to select an unbiased set of weights $\mu^{\mathcal{T}}$, $\mu_h = 1$, $\forall h$. On the other hand, several FOL formulas

[3] Generative models on the fulfillment of the single constraints could be considered too.

[4] See (Teso, 2019) for a discussion on hard constraints and graphical models in an adversarial context.

involve the implication operator $\Rightarrow$, that naturally implements if-then rules (if class v then class z) or, equivalently, rules that are about hierarchies, since $\Rightarrow$ models an inclusion (class v included in class z). However, whenever the premises are false, the whole formula holds true. It might be easy to trivially fulfill the associated constrains by zeroing all the predicates in the premises, eventually avoiding rejection. As rule of thumb, it is better to select μ_h's that are larger for those constraints that favor the activation of the involved predicates.

Single-label Classifiers. The type of domain knowledge described so far usually involves logic formulas that encode relationships among multiple classes, thus it is naturally associated with multi-label problems. Let us focus our attention on multi-label scenarios in which there exists a subset of categories that are known to be mutually exclusive, that we will refer to as *main classes*, while the remaining categories will be referred to as *auxiliary classes*. If we restrict the original classification problem to the main classes only, we basically end-up in a single-label scenario. Let us assume that the available logic formulas introduce relationships between (some of) the main classes and (some of) the auxiliary ones. As a result, in order to setup our defense mechanism (Eq. 5.8) or to learn with domain knowledge (Eq. 5.2), predictions on both the main and auxiliary classes must be available, so that the truth degree of the logic formulas can be evaluated. This consideration can be exploited to design classifiers that expose single label predictions on the main classes, thus acting as single-label classifiers, and include predictions on the auxiliary classes that are not exposed to the user at all, but that are internally used to setup our defense mechanism or to improve the quality of whole classifier when learning in a semi-supervised context. Formally, let us assume that the components $\{f_i, i = 1, \ldots, c\}$ of the vector function f are partitioned into two disjoint subsets, where the first one considers the components about the mutually-exclusive main classes and the second subset is about the auxiliary classes. We define with f^v the vector function with the elements in the first subset, while f^h is the vector function based on the elements of the second one, as shown in Fig. 5.3. The system only exposes to the user predictions computed by means of f^v, while the computations of f^h are hidden. Overall, the system can still exploit domain knowledge that consists in relationships between the classes associated to f^v (main classes) and the ones associated to f^h (auxiliary classes), or among the ones in f^h only, thus leveraging the learning principles that were described in Section 5.2. Moreover, the system can exploit the hidden predictions and the available knowledge to implement the knowledge-based rejection mechanism that we proposed in this section, as sketched in Fig. 5.3.

Due to the single-label nature of the visible portion of the classifier, existing state-of-the-art attacks, specifically designed for single-label models, can be used to fool the classifier in a black-box scenario. In Section 5.4, when the consider data are

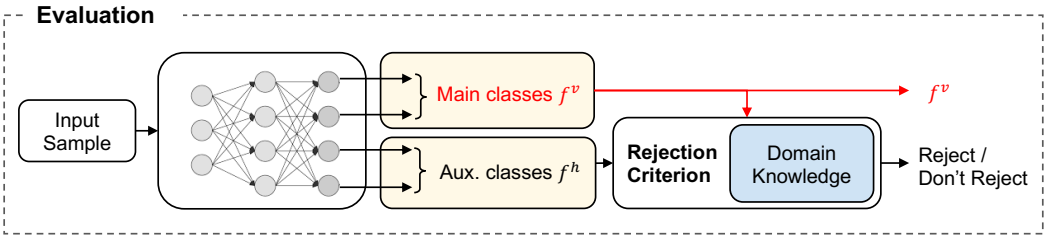


Figure 5.3: Single-label classifier on a set of mutually exclusive classes (*main classes*), computing the class activations by f^v and exposing them to the user (red path). It internally computes by f^h additional predictions over *auxiliary classes* that are involved in the domain knowledge (together with the main classes). Training considers *all* the classes, Fig. 5.1.

compatible with this special setting, we will exploit recent attack procedures to generate adversarial examples and evaluate the proposed knowledge-based rejection mechanism. Of course, differently from what we previously stated about the real multi-label setting, we cannot consider the cost of the rejection mechanism negligible in this case, since the system must learn the functions in f^h in order to be able to compute the rejection criterion.

5.3.1 Attacking multi-label classifiers

Robustness against adversarial examples is typically evaluated against *black-box* and *white-box* attacks (Biggio and Roli, 2018; Miller et al., 2020). In the black-box setting, the attacker is assumed to have only black-box query access to the target model, ignoring the presence of any defense mechanisms and without having access to any additional domain knowledge and related constraints. However, a surrogate model can be trained on data ideally sampled from the same distribution of that used to train the target model. Within these assumptions, gradient-based attacks can be optimized against the surrogate model, and then transferred/evaluated against the target one (Papernot et al., 2016a; Demontis et al., 2019). In the white-box setting, instead, the attacker is assumed to know everything about the target model, including the defense mechanism. White-box attacks are thus expected to also exploit the available domain knowledge to try to bypass the knowledge-based defense.

The existing literature on the generation of adversarial examples is strongly focused on single-label classification problems (see (Miller et al., 2020) and references therein). In such context, the classifier is expected to take a decision that is only about one of the r classes, and, in a nutshell, attacking the classifier boils down to perturb the input in order to make the classifier predict a wrong class. The whole procedure is subject to constraints on the amount of perturbation that the system is allowed to apply. Formally, given $x \in \mathcal{T}$, being $\mathcal{T}$ the test set, the attack generation

procedures in single-label classification commonly solve the following problem,

$$\begin{aligned} x^\star = \ & \arg\min_{x'}[-\varphi_s(f, \mathcal{K}_s, \{x'\}, \mu^{\mathcal{S}})], \\ \text{s.t.} \ \ & \|x - x'\| < \epsilon, \end{aligned} \tag{5.9}$$

being $\|\cdot\|$ an L_p-norm and $\epsilon > 0$. Each x has a unique class label/index attached to it and stored in $\mathcal{S}$, and φ_s is usually computed as a cross-entropy loss. Different attacks and optimization techniques for solving the problem of Eq. (5.9) have been proposed (Croce and Hein, 2020b). While there are no ambiguities on the class on which we want the classifier to reduce its confidence, i.e., the ground-truth (positive) class of the given input x, the class that the classifier will predict in input $x^\star$ might be given or not, thus each of the remaining $r-1$ classes could be a valid option. When moving to the multi-label setting, each $x \in \mathcal{T}$ is associated to multiple ground-truth positive classes, collected in set P_x, and we indicate with N_x the set of ground-truth negative classes of x. Differently to the previous case, due to the lack of mutual-exclusivity of the predictions, creating an adversarial example out of x is more arbitrary. For example, the optimization procedure could focus on making the classifier not able to predict any of the classes in P_x, or a subset of them. Similarly, the optimization could focus on making the classifier positively predict one or more classes of N_x.

Departing from the overwhelming majority of existing attacks for single-label classifiers, we propose a multi-label attack that focuses on the classes on which the classifier is less confident (thus easier to attack), that are selected and re-defined during the optimization procedure in function of the way the predictions of the classifier progressively change. Of course, in the *black-box* case, this attack is not considering that classes are related, and it is not taking care that, perhaps, changing the prediction on a certain class should also trigger a coherent change in other related classes. Differently, in the *white-box* setting, the previously introduced domain knowledge and, in particular, the corresponding loss of Eq. (5.1) is what encodes such relationships in a differentiable way, so that we can easily exploit it when crafting attacks. We first introduce the proposed multi-label attack in a *black-box* setting, in which domain knowledge is not available. To make gradient computation numerically more robust, as in (Carlini and Wagner, 2017b), we consider the activations (logits) of the last layer of f to compute the objective function, instead of using the cross-entropy loss. Let us define $p = \arg\min_i[f_i(x),\ i \in P_x]$, and $n = \arg\max_i[f_i(x),\ i \in N_x]$, i.e., p (n) is the index of the positive (negative) class with the smallest (largest) output score. These are essentially the indices of the classes for which x is closer to the decision boundaries. Our attack optimizes the

following objective,

$$\begin{aligned} x^\star = \ & \arg\min_{x'} [\max(l_p(x'), -\kappa) - \min(l_n(x'), \kappa)] \\ \text{s.t.} \ & \|x - x'\| < \epsilon, \end{aligned} \tag{5.10}$$

where l_j is the value of the logit of f_j, $\|\cdot\|$ is an L_p-norm (L_2 in our experiments), and in the case of image data with pixel intensities in $[0, 1]$ we also have $x' \in [0, 1]$. The scalar $\kappa \geq 0$ is used to threshold the values of the logits, to avoid increasing/decreasing them in an unbounded way (in our experiments, we set $\kappa = 2$). Optimizing the logit values is preferable to avoid sigmoid saturation. While the definition of Eq. (5.10) is limited to a pair of classes, we *dynamically* update p and n whenever logit l_p (l_n) goes beyond (above) the threshold $-\kappa$ (κ), thus multiple classes are considered by the attack, compatibly with the maximum number of iterations of the optimizer. This strategy resulted to be more effective than jointly optimizing all the classes in P_x and N_x. Moreover, the classes involved in the attack can be a subset of the whole set, as in (Song et al., 2018). In a *white-box* scenario, when the attacker has the use of the domain knowledge, the information in $\mathcal{K}_f$ provides a comprehensive description on how the predictions of the classifier should be altered over several classes in order to be coherent with the knowledge. In such a scenario, we enhance Eq. (5.10) to implement what we refer to as multi-label knowledge-driven adversarial attack (MKA), including the differentiable knowledge-driven loss φ_f in the objective function,

$$\begin{aligned} x^\star = \ & \arg\min_{x'} [\max(l_p(x'), -\kappa) - \min(l_n(x'), \kappa) + \alpha \cdot \varphi_f(f, \{x'\}, \mathcal{K}_f, \mu^{\mathcal{T}})], \\ & \text{s.t.} \ \|x - x'\| < \epsilon \end{aligned} \tag{5.11}$$

in which we set $\alpha > 0$ to enforce domain knowledge and avoid rejection. When crafting adversarial examples, MKA softly enforces the fulfillment of domain knowledge by means of the loss function φ_f. For *black-box* attacks, instead, we set $\alpha = 0$ to recover Eq. (5.10). MKA naturally extends the formulation of single-label attacks (when P_x is composed of a single class) and it allows staging both black-box and white-box (adaptive) attacks against our approach. Eq. (5.11) is minimized via projected gradient descent (1000 samples and 50 iterations in our experiments).

5.3.2 Impact of domain knowledge and main issues

Our approach is built around the idea of exploiting the available domain knowledge $\mathcal{K}_f$ on the target classification problem, both in the cases of rejection and multi-label attack. Several existing works use additional knowledge on the learning problem with different goals, being it represented by logic (d'Avila Garcez et al., 2019; Diligenti et al., 2017; Gnecco et al., 2015; Gori and Melacci, 2013), inherited by knowledge graphs or other external resources (Melacci et al., 2018; Yu and Dredze, 2014),

and encoded in multiple ways to face specific tasks (Pi et al., 2017; Morgado and Vasconcelos, 2017; Melacci et al., 2018). For instance, Semantic-based Regularization (Diligenti et al., 2017) and the theory formalized in (Gnecco et al., 2015) focus on the same approach we use here to convert generic FOL knowledge. On one hand, $\mathcal{K}_f$ might not always be available, thus limiting the applicability of what we propose and of the other aforementioned approaches. On the other hand, $\mathcal{K}_f$ is about relationships among classes that, in the case of the universal quantifier, hold $\forall x$. As a result, such knowledge is more generic than specific example-level supervisions. Human experts can produce FOL rules to a lesser effort than what is needed to manually label large batches of examples, since $\mathcal{K}_f$ naturally represents the type of high-level knowledge on the target domain that a human would develop during a concrete experience on the considered task (e.g., *if A happens, then also B or C are triggered, but not D*). Moreover, we are currently working on on Explainable AI methods (Ciravegna et al., 2020a; Barbiero et al., 2021b) extracting the type of knowledge that we consider in this chapter by means of special neural architectures. These methods will be explained in detail in Chapters 6 and 7.

When the number r of FOL formulas in $\mathcal{K}_f$ is large, a larger number of penalty terms $\hat{\phi}_h$ will be considered in φ_f of Eq. (5.1). Of course, every approach that exploits additional knowledge usually incurs in increased complexity when the knowledge base is large (d'Avila Garcez et al., 2019; Diligenti et al., 2017; Gnecco et al., 2015; Gori and Melacci, 2013). In our case, the T-Norm-based conversion does not represent an issue, since it is computed only once in a pre-processing stage, and, similarly, the output of the network $f(x, W)$ is computed only once in order to evaluate φ_f for a certain sample x and for given weights W, independently on the size of $\mathcal{K}_f$. However, the computation of φ_f must be repeated at each iteration of the optimization of Eq. (5.2) or Eq. (5.11), and when evaluating whether an input should be rejected or not, Eq. (5.8). From the practical point of view, the computational complexity scales almost linearly with r, but each $\hat{\phi}_h$ has a different structure depending on the FOL formula from which it was generated—roughly speaking, formulas involving more predicates usually yield more complex T-Norm-based polynomials. Several heuristic solutions are indeed possible to overcome these issues. For example, the knowledge base could be sub-selected in order to bound the number of rules in which each class is involved, or a stochastic optimization could be devised to sample the rules included in φ_f at each iteration of the optimization process. However, we remark that, in the experimental activities of this chapter, none of the mentioned issues arose.

The way we convert FOL rules into polynomial constraints, described in Section 5.2, inherits the flexibility of logic in terms of knowledge representation capabilities. Of course, the concrete impact of $\mathcal{K}_f$ in the rejection mechanisms or in MKA depends on the specific information that is encoded by the FOL rules. For instance,

suppose that $f_i(x) = 1$ for a certain x. The formula $f_i(x) \Rightarrow f_v(x) \vee f_z(x) \vee \ldots \vee f_u(x)$ is "more likely" to be fulfilled than the formula with an analogous structure in which $\vee$'s are replaced by $\wedge$'s. In the former, it is enough for a predicate in the conclusions to be 1, while in the latter, all the predicates of the conclusions must be jointly true. The rejection criterion or MKA are likely to be more effective in the latter case, but it cannot be strongly stated in advance, since it depends on the concrete way in which $f(\cdot, W)$ is developed by the learning procedure, as discussed in Section 5.3, and, in the case of MKA, on the difficulty in optimizing Eq. (5.11).

When restricting our attention to the rejection function of Eq. (5.8), a key element to the success of the proposed criterion is the choice of τ. In Section 5.3 we suggested using data in $\mathcal{V}$ to tune τ, that is a valuable solution, but, of course, it strongly depends on the quality of $\mathcal{V}$, similarly to what happens when tuning other hyper-parameters. More generally, a too small τ will result in a reject-prone system that does not reject only those inputs that are strongly coherent with the domain knowledge. A too large τ would end up in not rejecting inputs, being them coherent with $\mathcal{K}_f$ or not. If further information on the formulas in $\mathcal{K}_f$ is available, such as their expected importance with respect to the considered task, one could avoid computing an averaged measure as φ_f, and evaluate the penalty term $\hat{\phi}_h$ of each single formula against its own reject threshold (i.e., multiple τ's), that might be selected accordingly to the importance of the formula itself (i.e., smaller τ's in more important formulas).

5.4 Experiments

In this section, we report our experimental analysis, discussing the experimental setup in Section 5.4.1, and the results of standard and adversarial evaluations for multi-label classifiers in Section 5.4.2. We then show in Section 5.4.3 how our multi-label classifiers can also be adopted to mitigate the impact of adversarial examples in single-label classification tasks, when auxiliary classes are exploited. This allows us to highlight that our approach exhibits competitive performances with respect to other baseline defense methods designed under the same assumptions (i.e., without assuming any specific knowledge of the attacks) and against state-of-the-art attacks that are developed for single-label classification tasks.

5.4.1 Experimental settings

Datasets. We considered three image classification datasets, referred to as ANIMALS, CIFAR-100 and PASCAL-Part respectively. The ANIMALS and PASCAL-Part are the same dataset employed in the previous Chapter 4. The PASCAL-Part dataset, however, is here used in a multi-label classification task instead of object

Table 5.1: Datasets details on the experimental setting. "Classes" reports the total number of categories, main classes in parentheses. The fraction of labeled ($\%L$) samples, the level of partial labeling ($\%P$), and the number of training ($|\mathcal{L}|$), validation ($|\mathcal{V}|$), and test ($|\mathcal{T}|$) examples are reported.

Dataset	*Classes*	$\%L$	$\%P$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$
ANIMALS	33 (7)	30%	90%	5808	1244	1243
CIFAR-100	120 (100)	30%	0%	40000	10000	10000
PASCAL-Part	64 (20)	30%	70%	7072	1515	1515

Table 5.2: Values of the hyperparameter λ selected in our experiments.

Model	ANIMALS	CIFAR-100	PASCAL-Part
TL+C	10^{-2}	3	10^{-1}
TL+CC	1	10	1
FT+C	10^{-2}	3	10^{-1}

Table 5.3: Values of the constraint loss φ_f on the test data $\mathcal{T}$.

Model	ANIMALS	CIFAR-100	PASCAL-Part
TL	0.583 $\pm$0.031	1.444 $\pm$0.008	2.728 $\pm$0.085
TL+C	0.213 $\pm$0.016	1.040 $\pm$0.002	1.742 $\pm$0.051
TL+CC	0.200 $\pm$0.009	0.726 $\pm$0.002	0.738 $\pm$0.015
FT	0.375 $\pm$0.016	0.960 $\pm$0.004	2.447 $\pm$0.072
FT+C	0.089 $\pm$0.011	0.444 $\pm$0.006	0.843 $\pm$0.047

recognition. CIFAR-100, instead, is a popular benchmark composed of RGB images (32×32) belonging to different types of classes (vehicles, flowers, people, etc.),[5]

All datasets are used in a multi-label classification setting, so that the ground truth of each example is composed by a set of binary class labels. In the case of ANIMALS there are 33 categories, where the first 7 ones, also referred to as "main" classes, are about specific categories of animals (albatross, cheetah, tiger, giraffe, zebra, ostrich, penguin) while the other 25 classes are about more generic features (mammal, bird, carnivore, fly, etc.). The CIFAR-100 dataset is composed of 120 classes, out of which 100 are fine-grained ("main" classes) and 20 are superclasses. In the PASCAL-Part dataset, after having processed data as in (Donadello et al., 2017), we are left with 64 categories, out of which 20 are objects ("main" classes) and the remaining 44 are object-parts. We have the use of domain knowledge that holds for all the available examples. In the case of ANIMALS, as we have already seen in the previous Chapter, it is a collection of FOL formulas that were defined in the benchmark of P.H. Winston (Winston and Horn, 1986) and they involve relationships between animal classes and animal properties, such as $\forall x$ FLY$(x) \wedge$ LAYEGGS$(x) \Rightarrow$ BIRD(x). In CIFAR-100, FOL formulas are about the father-son

[5]CIFAR-100: `https://www.cs.toronto.edu/~kriz/cifar.html`

relationships between classes, with the following structure $\forall x \text{ FATHER}(x) \Rightarrow \vee_{i=1}^{\#sons} \text{SON}_i(x)$, $\forall x \text{ SON}_i(x) \Rightarrow \text{FATHER}(x)$, $i = 1, \ldots, \#sons$. As in the previous case, also here in PASCAL-Part FOL formulas either list all the parts belonging to a certain object, (type a) i.e., $\text{MOTORBIKE}(x) \Rightarrow \text{WHEEL}(x) \vee \text{HEADLIGHT}(x) \vee \text{HANDLEBAR}(x) \vee \text{SADDLE}(x)$, or they list all the objects in which a part can be found (type b), i.e., $\text{HANDLEBAR}(x) \Rightarrow \text{BICYCLE}(x) \vee \text{MOTORBIKE}(x)$ In all datasets we also introduced a disjunction or a mutual-exclusivity constraint among the main classes (type c), and another disjunction among the other classes (type d). Each dataset was divided into training and test sets (the latter indicated with $\mathcal{T}$). The training set was divided into a learning set ($\mathcal{L}$), used to train the classifiers, and a validation set ($\mathcal{V}$), used to tune the model parameters. We defined a semi-supervised learning scenario in which only a portion of the training set is labeled, sometimes partially (i.e., only a fraction of the binary labels of an example is known), as detailed in Table 5.1. We indicated with $\%L$ the percentage of labeled training data, and with $\%P$ the percentage of binary class labels that are unknown for each labeled example.[6]

Classifiers. We compared two neural architectures, based on the popular backbone ResNet50, trained using ImageNet data. In the first network, referred to as TL, we transferred the ResNet50 model and trained the last layer from scratch in order to predict the dataset-specific multiple classes (sigmoid activation). The second network, indicated with FT, has the same structure of TL, but we also fine-tuned the last convolutional layer. Each model is based on the product T-Norm, and it was trained for a number of epochs e that we selected as follows: 1000 epochs in ANIMALS, 300 (TL) or 100 (FT) epochs in CIFAR-100, and 500 (TL) or 250 (FT) in PASCAL-Part, using minibatches of size 64. We used the Adam optimizer, with an initial step size of 10^{-5}, except for FT in CIFAR-100, for which we used 10^{-4} to speedup convergence. We selected the model at the epoch that led to the largest F1 in $\mathcal{V}$. We considered unconstrained ($\lambda = 0$) and knowledge-constrained ($\lambda > 0$) models. The latter are indicated with the +C (and +CC) suffix.

Evaluation Metrics. To evaluate performance, we considered the (macro) *F1* score and a metric restricted to the main classes.[7] For ANIMALS and CIFAR-100, the main classes are mutually exclusive, so we measured the accuracy in predicting the winning main class (*AccMain*), while in PASCAL-Part we kept the F1 score (*F1Main*) as multiple main classes can be predicted on the same input.

[6] When splitting the training data into $\mathcal{L}$ and $\mathcal{V}$, we kept the same percentages of unknown binary class labels per example ($\%P$) in both the splits. Of course, in $\mathcal{V}$ there are no fully-unlabeled examples ($\%L$ is 100). Moreover, when generating partial labels, we ensured that the percentages of discarded positive (i.e., 1) and negative (i.e., 0) class labels were the same.

[7] We compared the outputs against 0.5 to obtain binary labels.

Table 5.4: Multi-label classification results in $\mathcal{T}$, for different models, averaged across different repetitions (standard deviations are $< 1\%$). The second block of rows is restricted to the main classes (Accuracy or F1). See the main text for details.

Metric	Dataset	TL	TL+C	TL+CC	FT	FT+C
F1 (%)	ANIMALS	98.3	**98.6**	98.1	98.6	**99.2**
	CIFAR-100	52.0	**55.1**	53.1	59.3	**64.0**
	PASCAL-Part	69.5	**70.0**	69.4	69.1	**71.0**
AccMain (%)[1] F1Main (%)[2]	ANIMALS[1]	98.8	**99.2**	**99.2**	98.5	**99.1**
	CIFAR-100[1]	53.3	**55.6**	52.8	60.5	**61.6**
	PASCAL-Part[2]	73.8	**75.9**	69.5	70.4	**75.0**

Hyperparameter Tuning. In Table 5.2 we report the optimal value of $\lambda \in \{10^{-2}, 10^{-1}, 1, 3, 5, 8, 10, 10^2\}$ for the TL+C and FT+C models used in our experiments, selected via a 3-fold cross-validation procedure. In the case of TL, we also considered a strongly-constrained (+CC) model with inferior performance but higher coherence (greater λ) among the predicted categories (that might lead to a worse fitting of the supervisions).[8] Table 5.3 reports the value of the constraint loss φ_f measured on the test set $\mathcal{T}$. We used $\mu^{\mathcal{L}} = \mu^{\mathcal{T}}$, setting each component $\mu_h^{\cdot}$ to 1, with the exception of the weight of the mutual exclusivity constraint or the disjunction of the main classes, which was set to 10 to enforce the classifier to take decisions.

5.4.2 Experimental results on multi-label classifiers

We discuss here the main experiments related to the evaluation of the considered multi-label classifiers.

Standard Evaluation. In order to assess the behaviors of the classifiers in the considered datasets and the available domain knowledge, we compared classifiers that exploit domain knowledge with the ones that do not exploit it. The results of our evaluation are reported in Table 5.4, averaged over the 3 training-test splits. For each of them, 3 runs were considered, using different initialization of the weights. The introduction of domain knowledge allows the constrained classifiers to slightly outperform the unconstrained ones.

Adversarial Evaluation. To evaluate adversarial robustness, we used the MKA attack procedure described in Section 5.3. and we restricted the attack to work on the

[8] FT+C has more learnable weights: constraint loss is already small.

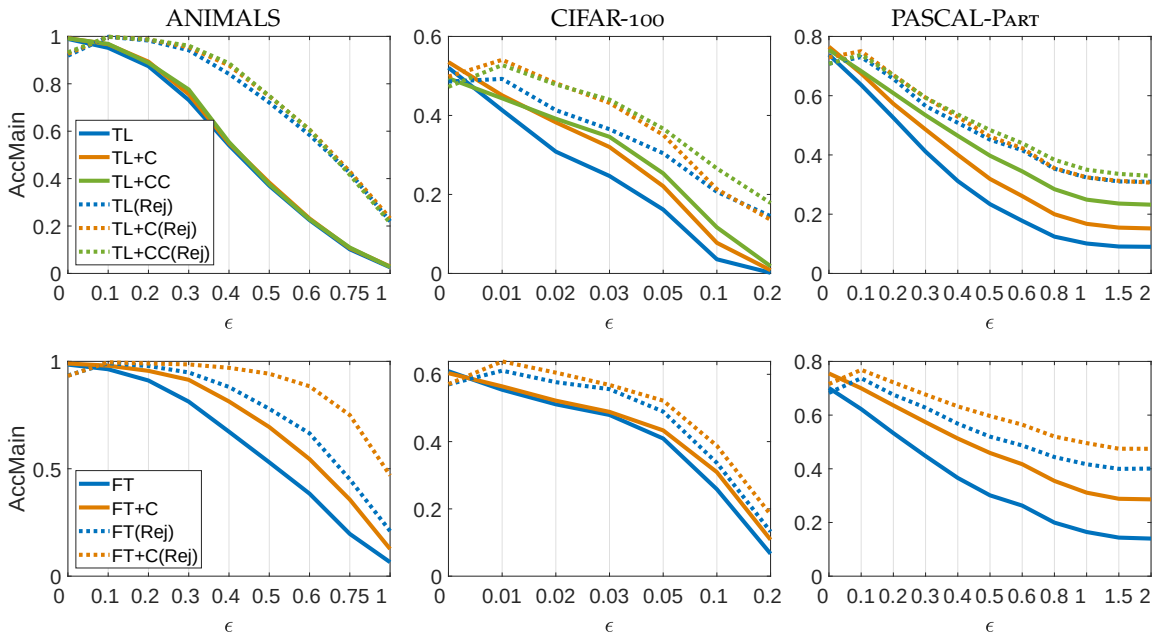


Figure 5.4: Black-box attacks. Classification quality of vanilla and knowledge-constrained models in function of ϵ. Dotted plots include rejection (Rej) of inputs that are detected to be adversarial.

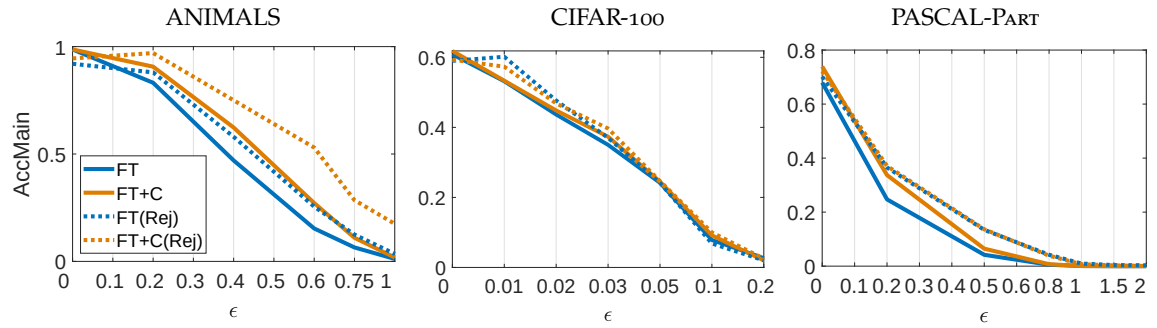


Figure 5.5: White-box attacks in the case of the FT classifiers. Classification quality of vanilla and knowledge-constrained models in function of ϵ. Dotted plots include rejection (Rej) of inputs that are detected to be adversarial.

already introduced main classes, being them associated to the most important categories of each problem. In ANIMALS and CIFAR-100 we assumed the attacker to have access to the information on the mutual exclusivity of the main classes, so that p in Eq. (5.11) is not required to change during the attack optimization. We also set $\kappa = \infty$ to maximize confidence of misclassifications at each given perturbation bound ϵ. All the following results are averaged over the three training runs.

In the *black-box* setting, we assumed the attacker to be also aware of the network architecture of the target classifier, and attacks were generated from a surrogate model trained on a different realization of the training set. Fig. 5.4 shows the clas-

sification quality as a function of the data perturbation bound ϵ, comparing models trained with and without constraints against those implementing the detection/rejection mechanism described in Eq. (5.3). When using such mechanism, the rejected examples are marked as correctly classified if they are adversarial ($\epsilon > 0$), otherwise ($\epsilon = 0$) they are marked as points belonging to an unknown class, slightly worsening the performance. The +C/+CC models show larger accuracy/F1 than the unconstrained ones. Despite the lower results at $\epsilon = 0$, models that are more strongly constrained (+CC) resulted to be harder to attack for increasing values of ϵ. When the knowledge-based detector is activated, the improvements with respect to models without rejection are significantly evident. No model is specifically designed to face adversarial attacks and, of course, there are no attempts to reach state-of-the-art results.[9] However, the positive impact of exploiting domain knowledge can be observed in all the considered models and datasets, and for almost all the values of ϵ, confirming that such knowledge is not only useful to improve classifier robustness, but also as a mean to detect adversarial examples at no additional training cost. In general, FT models yield better results, due to the larger number of optimized parameters. In ANIMALS the rejection dynamics are providing large improvements in both TL and FT, while the impact of domain knowledge is mostly evident on the robustness of FT. In CIFAR-100, domain knowledge only consists of basic hierarchical relations, with no intersections among child classes or among father classes. By inspecting the classifier, we found that it is pretty frequent for the fooling examples to be predicted with a strongly-activated father class and a (coherent) child class, i.e., we have strongly-paired classes, accordingly to Def. 5.3.1. Differently, the domain knowledge in the other datasets is more structured, yielding better detection quality on average, remarking the importance of the level of detail of such knowledge to counter adversarial examples. In the case of PASCAL-Part, the detection mechanism turned out to better behave in unconstrained classifiers, even if it has a positive impact also on the constrained ones. This is due to the intrinsic difficulty of making predictions on this dataset, especially when considering small object-parts. The false positives have a negative effect in the training stage of the knowledge-constrained classifiers.

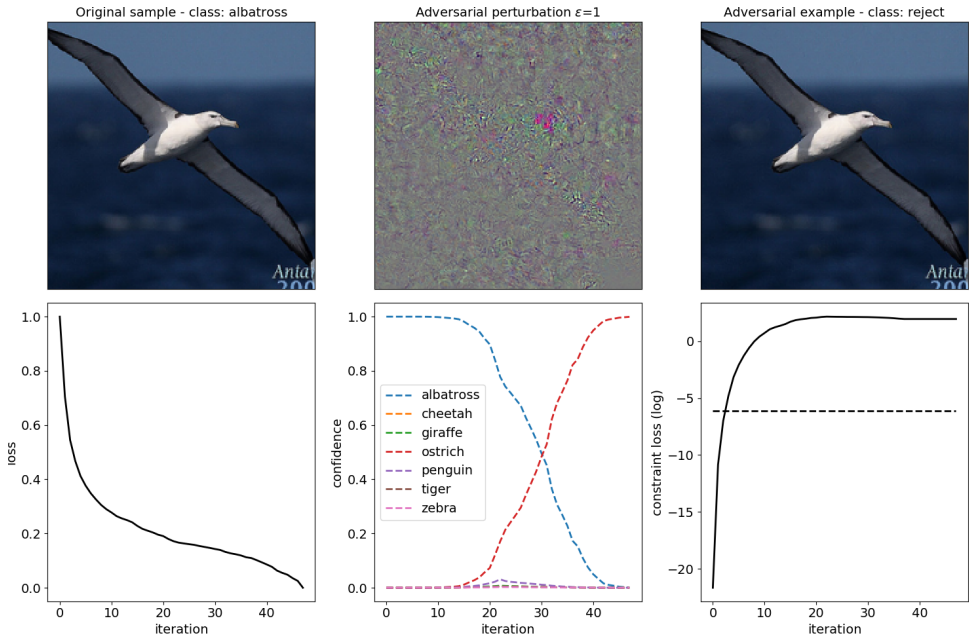
To provide a comprehensive, worst-case evaluation of the adversarial robustness of our approach, we also considered a *white-box* adaptive attacker that knows everything about the target model and exploits knowledge of the defense mechanism to bypass it. Of course, this attack always evades detection if the perturbation size ϵ is sufficiently large. We evaluated multiple values of α of Eq. (5.11), selecting the one that yielded the lowest values of such objective function. In Fig. 5.5 we report the

[9] Recall that our rejection mechanism is completely agnostic to the attack; it neither assumes any knowledge of the attack algorithm nor is retrained on adversarial examples. Nevertheless, it can be used as a complementary defense mechanism.

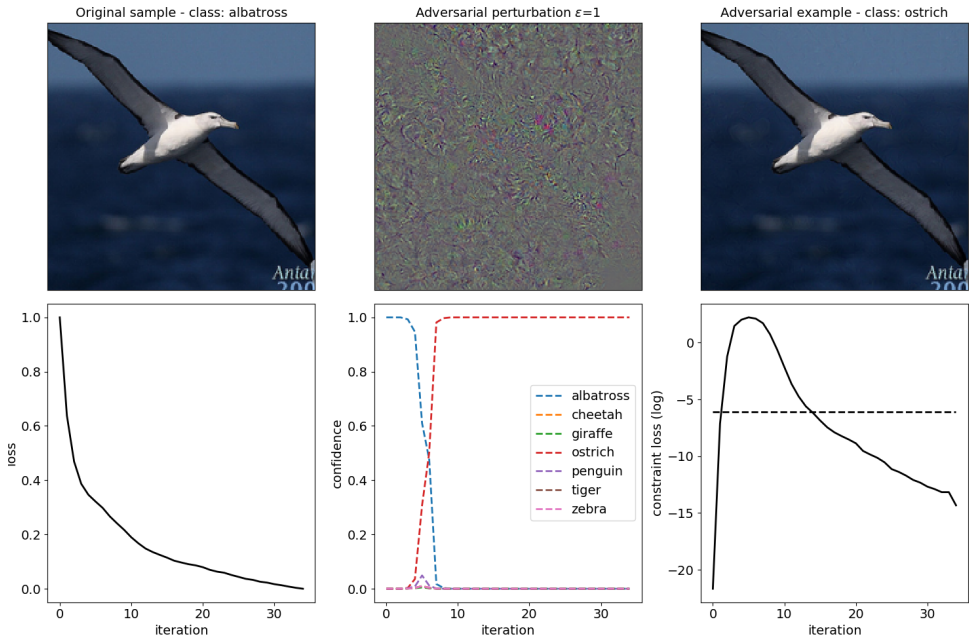
outcome of this analysis for FT models, showing that, even if the accuracy drop is obviously evident for all datasets, in ANIMALS the constrained classifiers require larger perturbations than the unconstrained ones to reduce the performance of the same quantity. A similar behavior is shown in CIFAR-100, even though only at small ϵ values. Accordingly, fooling the proposed detection mechanism is not always as trivial as one might expect, even in this worst-case setting. The impact of the rejection mechanisms is significantly reduced, as expected, but still having a positive impact. Finally, let us point out that the performance drop caused by the white-box attack is much larger than that observed in the black-box case. However, since domain knowledge is not likely to be available to the attacker in many practical settings, it remains an open challenge to develop stronger, practical black-box attacks that are able to infer and exploit such knowledge to bypass our defense mechanism.

Attack Optimization. We report here some additional details on the attack optimization process and on the parameter settings used in our experiments. Our attack optimizes Eq. (5.11) via projected gradient descent. Black-box attacks are non-adaptive, and thus ignore the defense mechanism. For this reason, the constraint loss term φ in our attack is ignored by setting its multiplier $\alpha = 0$ and $\kappa = \infty$. For white-box attacks on ANIMALS and PASCAL-PART, we set $\alpha = 0.1$ and $\alpha = 1$, respectively, while setting $\kappa = 2$. These values are chosen to appropriately scale the values of the constraint loss term φ w.r.t. the logit difference (i.e., the first term in Eq. 5.11, lower bounded by -2κ). This is required to have the sample misclassified while also fulfilling the domain-knowledge constraints. The process is better illustrated in Figs. 5.6a and 5.6b, in which we respectively report the behavior of the black-box and white-box attack optimization on a single image from the ANIMALS dataset, with $\epsilon = 1$. In particular, in each Figure we report the source image, the adversarial perturbation, and the resulting adversarial examples, along with some plots describing how the attack loss of Eq. (5.11) is minimized across iterations, and how the outputs on the main classes and the constraint loss φ change accordingly.

In both the black-box and white-box cases, the attack loss is progressively reduced during the iterations of the optimization procedure. While the *albatross* prediction is progressively transformed into *ostrich*, the constraint loss increases across iterations, exceeding the rejection threshold. Thus, the adversarial example is correctly detected. Similarly, the white-box attack is able to initially flip the prediction from *albatross* to *ostrich*, allowing the constraint loss to increase. However, after this initial phase, the attack correctly reduces the constraint loss after its initial bump, bringing its value below the rejection threshold. The system thus fails to detect the corresponding adversarial example. Finally, it is also worth remarking that, in both cases, the final perturbations do not substantially compromise the source image content, remaining essentially imperceptible to the human eye.



(a) Black-box attack on the ANIMALS dataset. The attack is able to flip the initial prediction from *albatross* to *ostrich*, but it is eventually detected as the constraint loss remains above the rejection threshold (dashed black line).



(b) White-box attack on the ANIMALS dataset. The attack is able to flip the initial prediction from *albatross* to *ostrich*, and then starts reducing the constraint loss which eventually falls below the rejection threshold (dashed black line). The attack sample remains thus undetected.

Figure 5.6: Visual examples of the black-box (a) and white-box (b) attacks.

Ablation Studies on $\mathcal{K}$ and τ. We investigated in more detail the relative impact of the domain information on a target problem, simulating the availability of differently sized knowledge bases, $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, $\mathcal{K}_4$, where each $\mathcal{K}_j \subseteq \mathcal{K}_f$. In particular, we considered the ANIMALS dataset, and we generated $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$ by removing some of the FOL formulas of the original $\mathcal{K}_f$ that was used in the previous experiments while $\mathcal{K}_4 = \mathcal{K}_f$. This means that some information that belongs to $\mathcal{K}_4$ is actually missing in the other knowledge sets. In detail, we created $\mathcal{K}_1$ by removing the rules that either include the 'mammal' or the 'bird' categories, while $\mathcal{K}_2$ is the outcome of discarding from $\mathcal{K}_f$ the rules including the 'mammal' category. Similarly, $\mathcal{K}_3$ is obtained removing the rules of the 'bird' category. We executed a batch of independent experiments, each of them using only one of the generated knowledge bases, and focusing on the same models of Fig. 5.4 (bottom) and Fig. 5.5, that were retrained from scratch. Fig. 5.7 (a,c) shows the classification quality we obtained in the black (a) and white box (c) settings, using the MKA attack and $\epsilon = 0.5$ (almost the mid of the plots in Figs. 5.4-5.5). In the black-box case, focusing on the models that include rejection (Rej), it is evident how larger knowledge bases yield better results. Interestingly, comparing the outcome of such models with the ones without rejection, we can see that our defense makes the classifier more robust to attacks even when using the smallest amount of knowledge ($\mathcal{K}_1$), confirming the versatility of what we propose. In the white-box setting there are still changes in the accuracy when varying $\mathcal{K}_j$, but they are not so evident and they lack a clear trend. This was expected, since, in this case, the attack procedure is aware of the domain knowledge. However, this result confirms the capability of MKA to craft adversarial examples that lead to knowledge-coherent predictions (to a certain extent) even when varying the level of detail of the knowledge sets.

In the same experimental setting we also explored the sensitivity of the system to the rejection threshold τ, using the whole knowledge set $\mathcal{K}$. We compared different τ's that are smaller or greater than the one we selected using validation data (Section 5.3), indicated here with $\tau^\star$. In particular, we evaluated $\tau \in \{10^\kappa \tau^\star,\ \kappa \in [-5, 5], \text{integer}\}$, and we measured both the classification quality on the perturbed data, as we did so far, and the rejection rate on the clean test data, where no samples should be rejected. Fig. 5.7 (b,d) reports these two measures on the y-axis and x-axis, respectively, and each marker is about a a specific τ, considering black (b) and white (d) box settings. The value of $\tau^\star$ has been highlighted with a red circle, and only the significant portions of the plots are shown. Too small thresholds (rightmost areas of each plot) lead to systems that frequently reject also clean examples, while too large τ's (leftmost areas) do not improve the quality of the classifiers, that are fooled by some of the data generated in an adversarial manner within the ϵ-bound. Overall, the $\tau^\star$ we selected represents a pretty appropriate trade-off between the two measures.

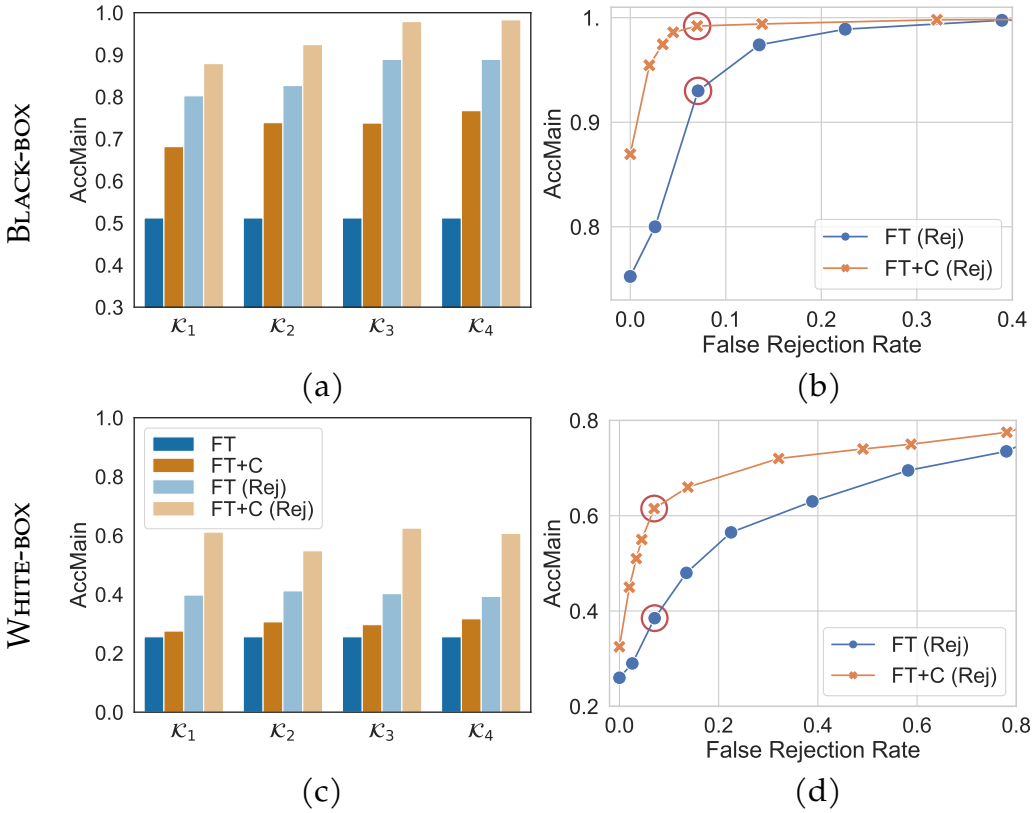


Figure 5.7: Further analysis of the proposed approach in the ANIMALS dataset ($\epsilon = 0.5$). Black-box setting top figures, white-box bottom; (a,c): increasing amounts of domain knowledge $\mathcal{K}_1, \ldots, \mathcal{K}_4$; (b,d): different values of the rejection threshold τ (from larger to smaller values, left-to-right).

5.4.3 Experimental results on single-label classifiers

The focus of this chapter is on multi-class classification paired with domain knowledge. However, as anticipated in Section 5.3 and qualitatively shown in Fig. 5.3, we can consider a special setting in which a single-label classifier internally includes predictors over auxiliary classes that are involved in the knowledge constraints. We experimentally evaluate this setting in the context of the ANIMALS and CIFAR-100 datasets, where the respective main classes (described in Section 5.4.1) are mutually exclusive (which is not the case of PASCAL-Part), thus well suited to simulate the setting of Fig. 5.3. We compared the proposed rejection mechanisms to a concurrent defense mechanism developed under the same assumptions (i.e., without assuming any knowledge of the attack algorithm), known as Neural Rejection (NR) (Melis et al., 2017; Sotgiu et al., 2020), and against the state-of-the-art attacks included in the AutoAttack framework (Croce and Hein, 2020b), developed for single-label classification tasks.

Table 5.5: ANIMALS dataset. Vulnerability analysis of the classifiers against MKA and state-of-art attacks—classification quality is reported, the same of Figs. 5.4-5.5 (first column). For each type of classifier (TL,FT), rows are organized into three groups, that are: models without rejection, with rejection (Rej), classifier equipped with Neural Rejection (NR). For each attack (columns—see (Croce and Hein, 2020b) for a description of the compared attacks), the result of the most robust classifier in the group is highlighted in bold. Models exploiting the proposed rejection (Rej) that overcome NR are marked with *, and vice-versa.

		White-box attacks ($\epsilon = 0.5$)					Black-box transfer attacks ($\epsilon = 0.5$)				
Model	$\epsilon = 0$	MKA	APGD-CE	APGD-T	FAB-T	Square	MKA	APGD-CE	APGD-T	FAB-T	Square
TL	99.0	25.0	17.5	14.9	20.0	96.6	45.3	29.4	29.1	83.1	98.4
TL+C	99.3	**25.0**	**19.0**	15.4	21.5	98.0	47.7	29.8	30.6	87.7	98.9
TL+CC	99.3	24.5	18.1	**15.5**	**22.7**	**98.2**	**48.0**	**30.2**	**32.0**	**89.5**	**99.0**
TL (Rej)	91.8	49.8	43.3	97.0*	100.0*	100.0*	85.6	53.7	98.4	99.9	99.9
TL+C (Rej)	92.3	56.8*	**47.8**	97.7*	100.0*	100.0*	91.2	**56.0**	98.6	99.9	99.9
TL+CC (Rej)	92.7	**57.5***	45.8	**98.1***	100.0*	100.0*	**93.4**	55.4	**98.8**	**100.0***	**100.0***
TL (NR)	99.2	55.5	58.5*	96.8	99.6	99.8	98.0*	71.7*	99.3*	100.0*	100.0*
FT	98.6	25.6	21.9	12.9	20.0	96.3	51.2	47.2	75.6	95.7	98.2
FT+C	99.1	**31.7**	**51.1**	**18.0**	**29.5**	**97.7**	**76.7**	**57.5**	**88.8**	**98.3**	**98.9**
FT (Rej)	92.7	39.3*	36.8	90.0*	99.7*	99.7*	88.9	66.9	99.6*	99.7*	99.8*
FT+C (Rej)	93.2	**60.7***	**66.6***	**97.3***	**99.8***	**99.9***	**98.3***	**82.2***	**99.9***	**99.9***	**99.9***
FT (NR)	98.6	37.3	38.3	87.3	97.0	99.6	91.0	79.2	98.7	99.2	99.5

Compared Defense and Attack Strategies. The NR defense mechanism, proposed in (Melis et al., 2017; Sotgiu et al., 2020), aims to reject inputs that are far from the training data in a given representation space. The rationale is that points with low support from the training set cannot be reliably classified, and should be thus rejected. To this end, the output layer of the deep network is replaced with a Support Vector Machine trained using the RBF kernel (SVM-RBF), which enforces the prediction scores to be proportional to the distance between the input sample and the reference prototypes (i.e., the support vectors) in the representation space. Samples are rejected if the prediction scores do not exceed the rejection threshold. Similarly to our approach, this defense mechanism does not make any assumptions on the attack to be detected, other than assuming an anomalous behavior with respect to the observed training data.

To compare our defense with NR, we have considered four different state-of-the-art evasion attacks: APGD-CE, APGD-T, FAB-T, and Square, implemented within the framework of AutoAttack (Croce and Hein, 2020b). APGD-CE (APGD-T) is an

Table 5.6: CIFAR-100 dataset. Vulnerability analysis of the classifiers against MKA and state-of-art attacks—classification quality is reported, the same of Figs. 5.4-5.5 (second column). Refer to the caption of Table 5.5 for more details (see (Croce and Hein, 2020b) for a description of the compared attacks).

		White-box attacks ($\epsilon = 0.03$)					Black-box transfer attacks ($\epsilon = 0.03$)				
Model	$\epsilon = 0$	MKA	APGD-CE	APGD-T	FAB-T	Square	MKA	APGD-CE	APGD-T	FAB-T	Square
TL	51.0	21.9	22.2	21.6	22.3	51.4	23.1	23.7	23.9	39.5	52.7
TL+C	52.9	**27.4**	24.6	24.2	25.1	**53.3**	32.3	35.5	37.9	**48.4**	**54.5**
TL+CC	50.5	27.1	**25.0**	**24.7**	**25.3**	49.5	**35.5**	**38.6**	**40.4**	46.9	51.5
TL (Rej)	48.1	26.9	33.2*	34.3*	34.4	59.2*	27.6	35.1	36.9	49.1	60.2*
TL+C (Rej)	49.4	**31.8***	**35.0***	**35.6***	**36.2**	**60.6***	40.7	44.8	47.0	**56.0***	**61.0***
TL+CC (Rej)	46.1	30.8*	34.0*	34.7*	35.4	55.7*	**45.4**	**46.3**	**47.6**	53.5*	57.0*
TL (NR)	49.0	30.5	30.1	24.5	39.6*	45.6	49.0*	48.3*	49.0*	51.3	53.3
FT	59.4	29.0	26.4	26.0	26.7	57.2	48.4	49.1	49.7	55.5	59.5
FT+C	60.0	**31.4**	**29.6**	**28.3**	**30.6**	**60.1**	**51.6**	**52.2**	**52.8**	**57.8**	**61.0**
FT (Rej)	57.4	31.1	37.5*	**42.0***	41.1	66.1*	55.1	57.2*	58.4*	62.7*	66.2*
FT+C (Rej)	56.7	**37.6***	**37.8***	41.1*	**44.6**	**67.0***	**60.2***	**59.5***	**60.3***	**64.4***	**67.1***
FT (NR)	59.7	36.5	35.3	30.4	50.9*	55.1	58.0	54.2	55.7	60.0	62.7

indiscriminate (targeted) step-free variant of the famous attack called PGD (Madry et al., 2018). Unlike PGD, the step size reduction is not scheduled a priori but instead governed by the optimization function trend. Moreover, both APGDs attack use momentum. FAB-T is the targeted version of an attack called Fast Adaptive Boundary Attack (FAB) (Croce and Hein, 2020a), which tries to find the minimum distance sample beyond the boundary of the desired class. The Square attack (Andriushchenko et al., 2020), differently from the previously mentioned ones, is a black-box attack; namely, it can query the classifier obtaining the predicted scores without exploiting any knowledge of the model architecture. By default, APGD-CE makes five random restarts, whereas the targeted versions of the attacks, i.e., APGD-T and FAB-T, run the attack nine times, each setting the target class as one of the nine top classes except the true class.

Adversarial Evaluation. In our experiments, we fixed the maximum allowed perturbation ϵ to 0.5 and 0.03 on the ANIMALS and CIFAR-100 datasets, respectively – the values in the middle of x-axis of Figs. 5.4-5.5 – and we used the default value for all the other attacks' parameters. Table 5.5 and Table 5.6 report the results of this analysis, showing the classification quality (same measure of Figs. 5.4-5.5) on the

clean (unmodified) test set $\mathcal{T}$ ($\epsilon = 0$) and on the attacked instances of $\mathcal{T}$ generated in the same white-box and black-box scenarios described in Section 5.4.2. As expected, white-box attacks are more effective than the black-box ones, reducing the model accuracy in a more evident manner. Results confirm that both the domain knowledge introduced at training time (+C and +CC) or exploited to implement the proposed rejection mechanism (Rej) improve the model robustness against all the considered attacks, and jointly using these strategies further improves it. On average, the performances of the unconstrained classifiers (TL and FT) paired with the proposed knowledge-based rejection are comparable with the ones paired with NR, even though they clearly behave in different manners across the datasets/attacks. Differently, when also considering constrained models (+C and +CC), in most of the cases we can find a classifier with rejection (Rej) that outperforms the unconstrained classifiers equipped with NR. On the clean samples ($\epsilon = 0$), the knowledge-based rejection criterion resulted more aggressive than NR.

MKA and APGD-CE/T are more effective than the other attacks. On average, their performances are comparable, and they depend on both the considered model and the dataset. In CIFAR-100, MKA outperforms APGD-CE/T on the black-box transfer scenario and against the model equipped with the proposed rejection mechanism (Rej), whereas on the ANIMALS dataset, APGD-CE usually obtained better results. APGD-CE/T leverages an optimization strategy that is more advanced than the one of MKA that, differently from APGD-CE/T, is designed to be used in multi-label problems too. For example, APGD-CE/T makes several attack restarts and uses a special type of adaptive step size. In the white-box setting, attacks yield a larger reduction of the performances. However, in the case of ANIMALS, the proposed rejection mechanism is still robust to all the attacks, with the exceptions of MKA, that is knowledge aware, and of APGD-CE. The fine-tuned optimization procedure in APGD-CE allows the attack to create samples that are confidently misclassified, and they end-up in belonging to space regions in which the classification functions are paired (Def. 5.3.1). In CIFAR-100, the rejection mechanism still has a positive impact, even if it is less significant than in ANIMALS.

In-depth Analysis. We further analyzed our results, visualizing the behavior of all the compared attacks in terms of value of the constraint loss of Eq. (5.3) and of the supervision loss – first term of Eq. (5.2). Figs. 5.8-5.9 show each generated adversarial example, highlighting them with different markers/colors in function of the corresponding attack procedure, on the ANIMALS and CIFAR-100 datasets, respectively, black-box (i.e., the constraint loss is measured for the purpose of determining whether to reject or not an example). Samples that are rejected are indicated with crosses, while circles represent the non-rejected ones. The dotted line is about the rejection threshold τ from Eq. (5.8).

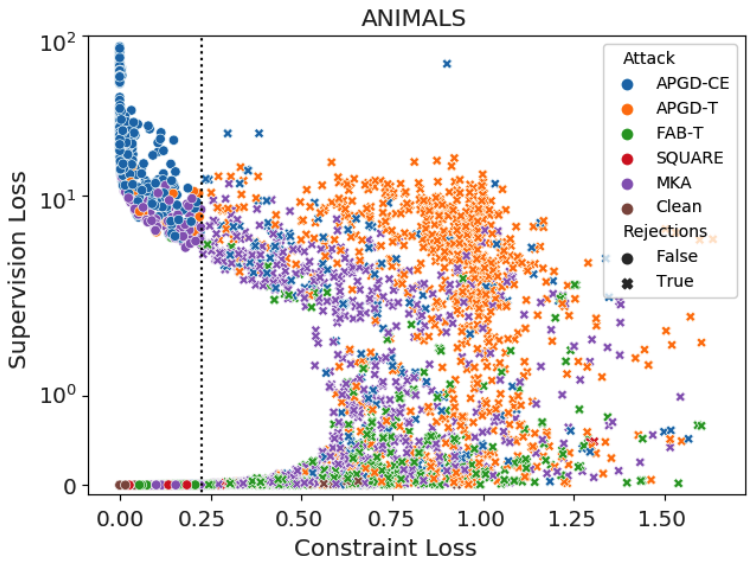


Figure 5.8: Adversarial data generated ($\epsilon = 0.5$) by different attacks – ANIMALS, TL+C(Rej), black-box. Examples that are rejected/not-rejected by the proposed knowledge-based criterion are depicted with crosses/circles ("Clean" indicates unaltered examples from the test set; the vertical line is the reject threshold).

In line with what we observed in the numerical results, in the case of ANIMALS, Fig. 5.8, it is evident how APGD-CE is actually able to craft attacks that strongly increase the supervision loss, still fulfilling the constraints (top-left area). Differently, the other attacks are not able to reach such result, so that their data is localized in high-constraint loss regions, easily rejected by the proposed technique, especially FAB-T, while Square actually fails in generating evident attacks. It is interesting to notice the D-shaped white region over the origin. It is an area in which constraints are almost fulfilled and the loss function can reach significantly non-null values, but no attacks fall there. This suggests that it is not straightforward to increase the supervision loss without violating the constraints. However, there are more extreme APCG-CE configurations with the largest supervision losses that also fulfill the knowledge (Fig. 5.8, top-left area). Of course, this depends on several factors, such as the type of domain knowledge that is available, the way we selected to convert it into polynomial constraints, and the constraint enforcement scheme, thus opening to future improvements. Moving to the CIFAR-100 dataset, Fig. 5.9, we observe different patterns with respect to the case of ANIMALS. This was clearly expected, since the two datasets differ both in terms of the problem they consider, the number of classes and in terms of the known relationships among such classes, described by the dataset-specific domain knowledge and embedded into the constraint loss. However, we can still observe the D-shaped region over the origin, even if in a less significant manner. On this dataset, the rejection rates are generally lower than ANIMALS. This is mostly due to the fact the constraint loss is larger also on the

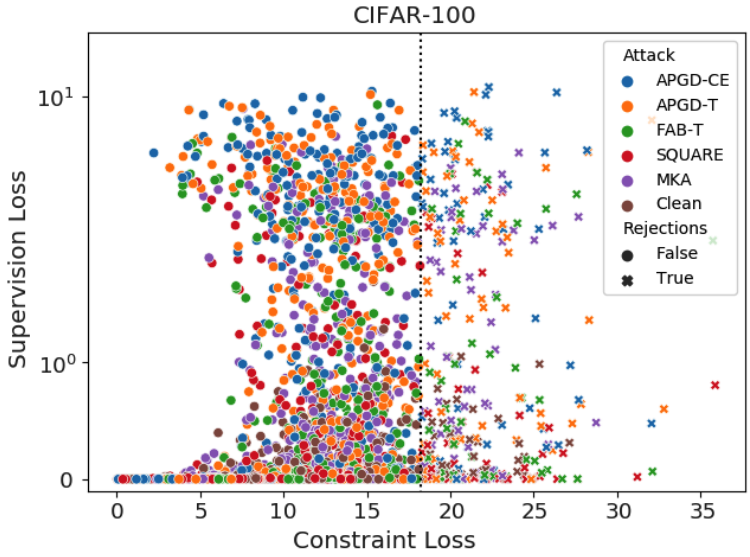


Figure 5.9: Adversarial data generated ($\epsilon = 0.03$) by different attacks – CIFAR-100, TL+C(Rej), black-box. See Fig. 5.8.

unaltered data, due to the already mentioned different problem and different type of domain knowledge. As a matter of fact, we have also a larger reject threshold τ. In this case, the behavior of the different attack strategies is more coherent, remarking previous considerations on the role of knowledge in shaping the attack distribution.

5.5 Related Work

In addition to the literature described in Section 2.3.2, we further emphasize the differences of what we propose with respect to the most strongly related approaches.

Multi-label adversarial perturbations. Most of the work in the adversarial ML area focuses on single-label classification problems. To the best of our knowledge, the first and only study on this problem is the one in (Song et al., 2018), in which the authors focus on targeted multi-label adversarial perturbations defining in advance the set of classes on which the attack is targeted (being them positive or negative) and also introducing another set of classes for which the attack is expected not to change the classifier predictions. The framework described in (Song et al., 2018) is only experimented in a *static/targeted* context, i.e., by selecting in advance the sets mentioned above using custom criteria to simulate the attacking scenario artificially. The multi-label attack that we propose in this chapter is instead *dynamic/untargeted* and without the need of defining in advance what are the classes to be considered. Regarding the defenses, to our best knowledge, none of the previously proposed ones leverage multi-label classification outputs.

Semi-supervised Learning and Adversarial Training. In the context of adversarial machine learning, unlabeled data are usually employed to improve the robustness of the classifier by performing adversarial training. The rationale behind such training scheme is that if the available unlabeled samples are perturbed, then the predicted class should not change. Miyato et al. (Miyato et al., 2016, 2018) and Park et al. (Park et al., 2018) exploit adversarial training (virtual adversarial training and adversarial dropout, respectively) to favor regularity around the supervised and unsupervised training data, and to improve the classifier performance. The work in (Akcay et al., 2018) develops an anomaly detector using adversarial training in the semi-supervised setting. Self-supervised learning is exploited in (Carmon et al., 2019; Najafi et al., 2019) to gain stronger adversarial robustness. Stability criteria are enforced on unlabeled training data in (Zhai et al., 2019), whereas the work in (Alayrac et al., 2019) specifically focuses on an unsupervised adversarial training procedure in the context of semi-supervised classification. Our model neither exploits adversarial training nor any adversary-aware training criteria aimed at gaining intrinsic regularity. We focus on the role of domain knowledge as an indirect means to increase adversarial robustness and, afterward, to detect adversarial examples. Therefore, the proposed apporoach is not attack-dependent, and it is faster at training time as it does not require generating adversarial examples. We believe that using unlabeled data also to simulate attacks and incorporate them into the training process may further improve robustness. All the described methods could also be applied jointly with what we propose.

Rejection-based Approaches for Adversarial Examples. A different line of defenses, complementary with adversarial training, is based on detecting and rejecting samples sufficiently far from the training data in feature space. Our approach differs from other adversarial-example detectors (Carlini and Wagner, 2017a; Ma et al., 2018; Samangouei et al., 2018; Miller et al., 2020) as it has no additional training cost and negligible runtime cost. We are the first to show that domain knowledge can be used to reject adversarial examples and also to propose a detector that exploits unlabeled data.

Domain-Agnostic Methods and Semantic Attacks. Recent work in adversarial attacks considers the role of the learning domain and of additional semantic information, even if with different goals to the ones of this chapter. The way the learning domain is related to the generation of attacks was recently studied in (Naseer et al., 2019), that is based on the idea of developing generative adversarial perturbations easily transferable from the source domain (where the attack function is modeled) to another domain. Differently, we focus on knowledge that is domain specific and used both for defending and creating more informed attacks. The knowledge of a set

semantic attributes is used to implement the threat model of semantic adversarial attacks in (Joshi et al., 2019). A generative network is considered, and the attack procedure focuses on altering the activation of such human-understandable attributes, that, in turn, yield visible changes in the input image (e.g., adding glasses to the input face). Differently, our approach is built on an L_p-norm-bounded perturbation model that does not enforce the input image to change in a human-understandable manner. Our approach considers a more generic notion of knowledge, that includes information also on the relationships within subsets of logic predicates, and that exploits the power of FOL. Predicate activations are modeled by neural networks and not by scalar variables as for the attributes of (Joshi et al., 2019).

Part III

Extracting Logic Constraints to Explain Deep Learning

Chapter 6

Devising Explanations with an Auxiliary Network

In this chapter, we show how it is possible to extract logic constraints from a neural network (as introduced in Section 3.3) with the aim of tackling the explainability problem introduced in Section 2.3.3. Part of the content of this chapter is extracted from the works Ciravegna et al. (2020b) and Ciravegna et al. (2020a), that have been accepted at the AAAI2020 and IJCAI 2020 conferences. These papers have been written with some colleagues from the University of Siena.

The chapter is organized as follows. In Section 6.1 a high-level overview of the problem and of the proposed framework is given. Section 6.2 introduces the use cases covered in this chapter, while the proposed model is described in Section 6.3. Experiments are collected in Section 6.4 showing the validity of the proposed approach in two computer vision problems. Finally, Section 6.5 resumes the related work and concludes the chapter.

6.1 Introduction

In this chapter we propose a general framework to learn explanations with respect to classifier outputs, according to certain criteria that may be specified by the final user. In particular, given a multi-label classification problem, we aim at learning both a set of *classifier* (also referred as *task* or *predicate*) functions and a set of *explanations*, that express some knowledge about the classifier activation. In this scenario it is possible to exploit the acquired explanations to both improve the classifier performances and obtain newly devised knowledge eventually converted into symbolic form. The main advantage of this approach is that it may be used to simultaneously produce different customizable pieces of knowledge about the task functions.

Again, also in this chapter we consider multi-label classification, where each input example belongs to one or more classes, but we focus on the problem of ex-

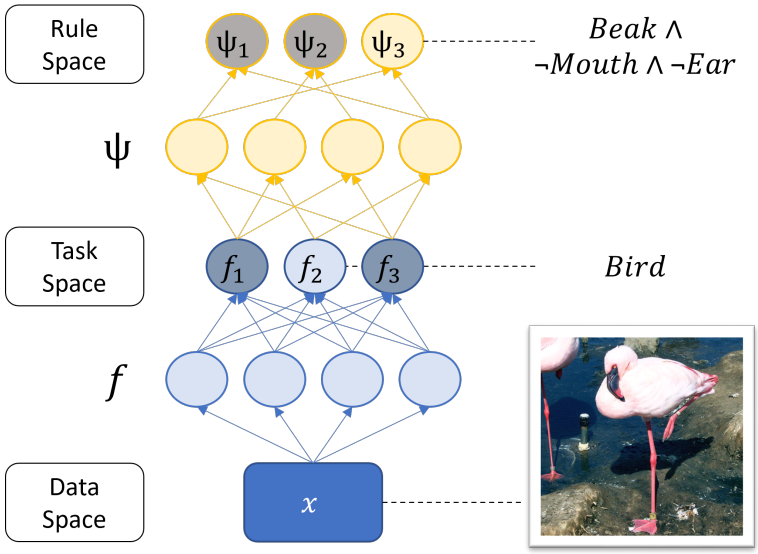


Figure 6.1: Overview of the proposed approach. Input data are x are mapped to one or more classes $f(x)$. The relationships among the classes are uncovered by the ψ neural network, which also provides for FOL explanations. In this case the relationships between the predicted class $Bird(x)$ and the other classes is taken into consideration. Figure from the *PASCAL-Part* dataset (Chen et al., 2014).

tracting First-Order Logic (FOL)-based explanations of the behaviour of a classifier. More precisely, we focus on neural network-based systems, that implicitly learn from supervisions the relationships among the considered classes. We propose to introduce another neural network that operates in the output space of the classifier, also referred to as task space, further projecting the data onto the so-called rule space, where each coordinate represents the satisfaction of a constraint that we describe by a FOL rule, as it is shown in Figure 6.1.

In particular, we propose to progressively prune the connections of the newly introduced network and interpret each of its neurons as a learnable boolean function (an idea related to several methods (Fu, 1991; Towell and Shavlik, 1993; Tsukimoto, 2000; Sato and Tsukimoto, 2001; Zilke et al., 2016)), ending up in a FOL formula for each coordinate of the rule space.

The tasks-to-rules projection can be learned by using different criteria, that bias the type of rules discovered by the system. On one side, when the user does not express any preference on the required explanation, we propose a general unsupervised criterion based on information principles, following Melacci and Gori (2012). However, humans usually have expectations on the kind of explanations they might get. For example, suppose we are training a network to classify digits and also to predict whether they are even numbers. If we do not know what being *even* means, we might be particularly interested in knowing the relationships between the class *even* and the other classes (i.e., that even numbers are 0 or 2 or 4 or 6 or 8). It could

not be so useful to discover that 0 is not 2, even if it is still a valid explanation in the considered multi-label problem. Motivated by this consideration, we propose a generic framework that can discover both unbiased and user-biased explanations.

A key feature of the proposed framework is that learning the classifier and the explanation-related network takes place in a joint process, differently from what could be done, for example, by classic data mining tools (Liu, 2007; Witten and Frank, 2005). This implicitly introduces a latent dependency between the development of the explanation mechanism and the one of the classifiers. When cast into the semi-supervised learning setting, we show that linking the two networks can lead to better quality classifiers, bridging the predictions on the unsupervised portion of the training data by means of the explanation net, that acts as a special regularizer.

6.2 Scenarios

Following the notation introduced in Section 3.1, we consider a multi-label classification problem, in which a multi-output classifier f is learned from data. Each output unit is associated to a function in $[0, 1]$ that predicts how strongly an input example belongs to the considered class. We will also interchangeably refer to these functions as *task functions* (in a more general perspective where each function is related to a different task), or *predicates* (if we interpret each output score as the truth value of a logic predicate).

We also consider a set of logic constraints ψ, that express the *explanations* on the relationships among the task functions, and that are the outcome of the proposed approach. Such knowledge is not known in advance, and it represents a way to explain what the classifier implicitly learned about the task functions. In order to guide the process of building the explanations, the user can specify one or more preferences. In particular, the user can decide if the explanations have to describe local relationships that only hold in sub-portions of the task space or global rules that hold everywhere, or even if they must focus on a user-selected task function (as in the example in the introduction of the chapter). In what follows we report an overview of the specific use cases explored in this chapter.

Local Explanations. In this scenario, the explanations are automatically produced without making any assumptions on which task functions to consider. In order to provide a valid criterion to develop explanations, we enforce them to only hold in sub-portions of the task space and, overall, to cover the whole dataset. The user can provide an example to the trained network and get back the explanation associated to it, that may highlight partial co-occurrences of the task functions. For instance, the system might discover that "*eyes* or *sunglasses*" is a valid rule for some pictures (the ones with faces) but not for others (the ones without faces).

Global Explanations. Local explanations may provide very specific knowledge concerning only small portions of data. In order to describe more general properties that hold on the whole dataset, we may be interested in global explanations. Global explanations may catch general relations among task functions that are valid for all the points of the considered dataset, such as mutual exclusion of two classes or hierarchical relations.

Class-driven Explanations. The user may require explanations about the behaviour of specific task functions. He could also specify if he is looking for necessary conditions ($IF \rightarrow$) or necessary and sufficient explanations ($IFF \leftrightarrow$). For instance, focusing on the driving class *man*, we may discover that a certain pattern is classified as "*man* only if it is also classified as containing *hand*, *body*, *head*", and so on. In the example at the beginning of the chapter, *even* was the class driving a necessary and sufficient explanation. The rules of this scenario are completely tailored around the user-selected target classes.

Combined Explanations. All the scenarios described so far may be arbitrarily combined in case the user is simultaneously interested in multiple explanations according to different criteria. In particular, some explanations might have to specify the behaviour of some task functions, while the remaining ones might have to be automatically acquired in order to describe global or local interactions.

6.3 Model

Following what introduced in Section 3.3, we consider data x belonging to an *input space* $X \subset \mathbb{R}^d$ with y labels living in a *task space* $Y \subseteq \mathbb{R}^r$ each of them associated to a task function $f_i, i = 1 \dots, r$ (that corresponds to an output unit of a neural network).

We also consider a set of learnable constraints $\psi_j, j = 1 \dots, m$, whose inputs lives in the *task space* while it outputs in the *rule space*. Each $\psi_j(f(x))$ expresses the satisfaction of a certain constraint with respect to the output of the task functions on the data sample $x \in X$. To simplify the notation, in this case we denote with 1 the satisfaction of the constraint so that when the output neuron $\psi_j(f(x)) = 1$, the corresponding constraint is satisfied. In addition, we assume $\psi_j(f(x)) \in [0, 1]$ in order to relate the value of ψ_j to the truth-degree of a certain FOL formula which represent an *explanation* of $f(x)$ on the input sample $x \in X$. For this reason, we will frequently make no explicit distinctions between constraints FOL formula and explanations throughout the chapter.

Different criteria are needed to learn the parameters of the functions ψ_j in order to implement the scenarios of Section 6.2, as we will describe in Section 6.3.1. Once the explaining functions are learnt, we will consider their approximation as boolean

functions, and they will be given a description in terms of FOL, as we will discuss in Section 6.3.2. Throughout the chapter, the notation $\hat{\psi}_j$ denotes both the approximating boolean function and its associated logical formula. Finally, as introduced in Section 3.3, $\mathcal{X}_{\psi_j}$ denotes the subset of the input space where the j-th explanation holds true, also named its *support*, i.e., $\mathcal{X}_{\psi_j} = \{x \in X : \hat{\psi}_j(f(x)) = 1\}$.

6.3.1 Learning criteria

We consider a semi-supervised setting in which only a portion of the data $\mathcal{S} \subseteq \mathcal{X} \subset X$ is labelled (Melacci and Belkin, 2011). This is a natural setting of several real-world applications, since getting labelled data is usually costly, and it also allows us to better emphasize the proprieties of the explanation learning mechanisms, that can exploit both labelled and unlabelled training data with no distinctions.

Again, following the notation of Section3.3, a classic given knowledge K_s representing the supervision on the labelled data is considered to train the task functions f_i's, paired with a regularization criterion to favour smooth solutions (weight decay). For the sake of simplicity, in this scenario we consider that only the knowledge about the supervision K_s and no further knowledge (e.g. on the relations of the task function K_f) is available. Therefore, the problem of learning from – and of – constraints can be formalized as,

$$f^\star, \psi^\star = \arg\min_{f,\psi} \{U(f, \mathcal{S}) + D(\psi, f, \mathcal{X})\} \ , \tag{6.1}$$

where $U(f, \cdot)$ represents the problem of learning *from constraints* on a certain set of data and is defined as $U(f, \mathcal{S}) = \varphi(f, \mathcal{K}_s, \mathcal{S})$ (see Eq. 3.6) and $D(\psi, f, \mathcal{X})$ represents the problem of learning *of constraints* (see Eq. 3.8).

In order to implement the scenarios of Section 6.2, we need to properly define some learning criteria $D(\psi, f)$ involving the explaining functions ψ_j's, for all x's, being them labelled or not. In the following, some possible formulations are proposed and are summarized in Fig. 6.2, which also describe their relations with the scenarios of Section 6.2.

Mutual Information-based Criterion. The maximization of the Mutual Information (MI) between the task and rule spaces can be enforced in order to implement the principles behind the *Local Explanations* scenario, and it could also be used as a basic block to implement the *Global Explanations* scenario (Section 6.2). In the latter case, further operations are needed, and they will be described in Section 6.3.2.

Maximizing the transfer of information from the r task functions to the m explaining functions is a fully unsupervised process that leads to configurations of the ψ_j's functions such that, for each $x \in \mathcal{X}$, only one of them is active (close to 1) while all the others are close to zero (see Melacci and Gori (2012)). In order to

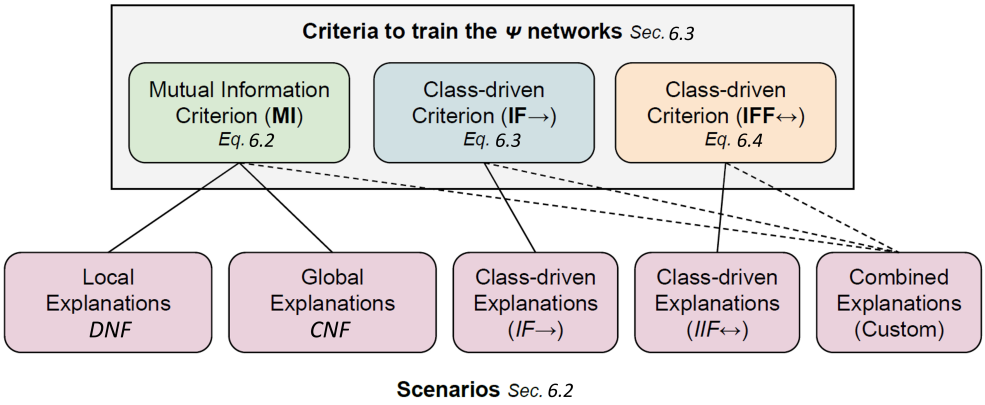


Figure 6.2: The learning criteria of the proposed framework and their relations with the use-cases of Section 6.2.

define the MI index, we introduce the probability distribution $P_{\Psi=j|Y=f(x)}(\psi, f(x))$, for all j, as the probability of ψ_j to be active in $f(x)$. Following the classic notation of discrete MI, Ψ is a discrete random variable associated to the set of constraint functions ψ, while Y is the variable related to the output of the task functions f in the task space. We implemented the probability distribution using the softmax operator, scaling the logits with a constant factor to ensure that when $\psi_j(x) = 1$ all the other $\psi_{z\neq j}0$ are zero. The developmental function minimize is minus the MI index, that is:

$$D_{MI}(\psi, f, \mathcal{X}_\psi) = -H_\Psi(\psi, f, \mathcal{X}) + H_{\Psi|Y}(\psi, f, \mathcal{X}) \,, \tag{6.2}$$

where H_Ψ and $H_{\Psi|Y}$ denote the entropy and conditional entropy functions (respectively) associated to the aforementioned probability distribution and measured over the whole set of input data $\mathcal{X}$. An outcome of the maximization of the MI index is that the supports of the explaining functions will tend to partition the input space $\mathcal{X}$, i.e., $\mathcal{X} = \bigcup_{j=1}^{m} \mathcal{X}_{\psi_j}$ and $\mathcal{X}_{\psi_j} \cap \mathcal{X}_k = \emptyset$, for $j \neq k$ (see Melacci and Gori (2012); Betti et al. (2019) for further details).

Class-driven Criteria. The *Class-driven Explanations* scenario of Section 6.2 aims at providing explanations for user-selected task functions. Let us assume that the user wants the system to learn an explaining function $\psi_{h(i)}$ that is driven by the user-selected f_i, being $h(\cdot)$ an index mapping function. We propose to enforce the support $\mathcal{X}_{\psi_{h(i)}}$ of $\psi_{h(i)}$ to contain ($IF \rightarrow$) or to be equal to ($IFF \leftrightarrow$) the space regions in which f_i is active. Notice that f_i and $\psi_{h(i)}$ have different input domains (perceptual space and task space, respectively), so we are introducing a constraint between two different representations of the data (see e.g. Melacci et al. (2009)). Moreover, since the goal of this scenario is to explain f_i in terms of the other $f_{u\neq i}$'s, we mask the i-th component of $f(x)$ by setting it to 0 for all $x \in X$. This also avoids trivial solutions in which $\psi_{h(i)}$ only depends on f_i.

We denote by $P, S \subseteq \{1, \ldots, n\}$ the disjoint sets of task function indexes selected for class-driven *IF* $\rightarrow$ and *IFF* $\leftrightarrow$ explanations, respectively. The loss terms that implement the described principles are reported in Eq. 6.3 and Eq. 6.4,

$$D_{\rightarrow}(\psi, f, \mathcal{X}_{\psi}) = \sum_{i \in P, x \in \mathcal{X}} \max\{0, f_i(x) - \psi_{h(i)}(f(x))\} \tag{6.3}$$

$$D_{\leftrightarrow}(\psi, f, \mathcal{X}_{\psi}) = \sum_{i \in S, x \in \mathcal{X}} |f_i(x) - \psi_{h(i)}(f(x))| \,. \tag{6.4}$$

While Eq. 6.3 does not penalize those points on which $\psi_{h(i)}(x) > f_i(x)$, Eq. 6.4 specifically enforces the $\psi_{h(i)}$ and f_i to be equivalent. In order to avoid trivial solutions of Eq. 6.3 in which, for instance, $\psi_{h(i)}$ is always 1, we enforce the supervision y_i also on the output of $\psi_{h(i)}$. Notice that these losses never explicitly estimate $\mathcal{X}_{\psi_{h(i)}}$.

Class-driven & Mutual Information-based Criteria. The *Combined Explanations* scenario of Section 6.2 is the most general one, and it can be implemented involving all the penalty terms described so far. The MI index can be enforced only on those ψ_j's for which the user is looking for a local explanation, while other explaining functions can be dedicated to class-driven explanations. Interestingly, we can also nest the MI index inside a class-driven explanation, since the user could ask for multiple local explanations for each selected driving class. In this case, multiple ψ_j's are allocated for each driving class, and the MI index is computed assuming the probability distribution of the discrete samples in the task space to be proportional to the activation of the task function we have to explain. This scenario can be arbitrarily made more complex, and it is out of the scope of this chapter to focus on all the possible combinations of the proposed criteria.

6.3.2 First-order logic formulas

Each explaining function ψ_j is a $[0,1]$-valued function defined in $[0,1]^m$. At the end of the training stage, each ψ_j is converted into a boolean function $\hat{\psi}_j$ (this is also considered with a different goal e.g. in Fu (1991); Towell and Shavlik (1993); Tsukimoto (2000); Sato and Tsukimoto (2001); Zilke et al. (2016)), and then converted into a FOL formula.

The booleanization step is obtained by approximating any neuron output with its closest integer (assuming sigmoids as activation functions, this value can only be 0 or 1) and by repeating this process for each layer, from the output neurons of the task functions up to the output layer of ψ. As a result, for each neuron we get a boolean function, whose truth-table can be easily rewritten as a boolean formula in *Disjunctive Normal Form* (DNF), i.e., a disjunction of minterms (conjunction of literals). By composing the formulas attached to each neuron, accordingly to the

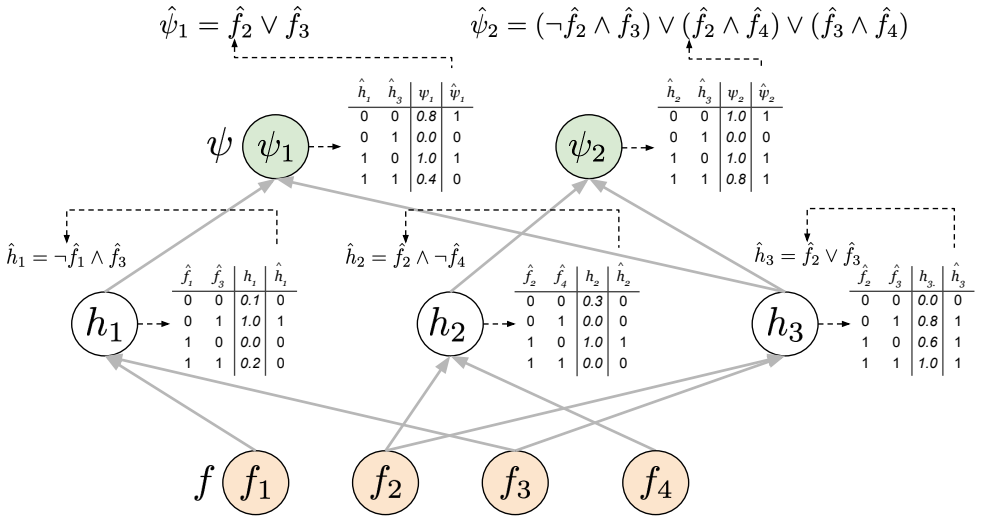


Figure 6.3: Extracting FOL formulas from each ψ_j. Hidden and output neurons are paired truth tables (right) and their corresponding logic description (top), as described in Section 6.3.2. The truth tables include the real-value neuron outputs (third column) and their boolean approximation (last column). The FOL descriptions of ψ_1, ψ_2 are the outcome of composing the truth tables of the hidden neurons.

network structure, we get $\hat{\psi}_j$, that is the boolean formula of the output neuron associated to ψ_j. The whole procedure is illustrated in the example of Figure 6.3. Clearly, this procedure is efficient only if the fan-in of each neuron is small, a condition that we enforce with the procedure described in Section 6.3.3.

In the case of *Local Explanations* (Section 6.2), each ψ_j is close to 1 only in some sup-portions of the space, due to the maximum mutual information criterion, so that the FOL rule $\hat{\psi}_j$ will hold true only on $\mathcal{X}_{\psi_j} \subset X$ (and false otherwise). As a consequence, each explanation is local, $\forall x \in \mathcal{X}_{\psi_j},\ \hat{\psi}_j(f(x))$, for $j = 1, \ldots, m$.

The case of *Global Explanations* (Section 6.2) is still built on the maximum mutual information criterion. A global explanation (i.e., an explanation holding on the whole input space X) can be obtained by a disjunction of $\hat{\psi}_1, \ldots, \hat{\psi}_m$. However, the resulting formula will be generally unclear and quite complex. A possible approach to get a set of global explanations starting from the previous case is then to convert it in *Conjunctive Normal Form* (CNF), i.e., a conjunction of m', $m' \neq m$ disjunctions of literals $\hat{\psi}'_k$, $k = 1, \ldots, m'$,

$$\bigvee_{j=1}^{m} \hat{\psi}_j(f(x)) \equiv \bigwedge_{k=1}^{m'} \hat{\psi}'_k(f(x)) \,. \tag{6.5}$$

In this case, the following global formulas are valid in all $\mathcal{X}, \forall x \in \mathcal{X},\ \hat{\psi}'_k(f(x))$, for $k = 1, \ldots, m'$.

Unfortunately, converting a boolean formula into CNF can lead to an exponential explosion of the formula. However, after having converted each $\hat{\psi}_j$ in CNF, the conversion can be computed in polynomial time with respect to the number of minterms in each $\hat{\psi}_j$ (Russell and Norvig, 2016).

The *Class-driven Explanations* (Section 6.2) naturally generate rules that hold for all $\mathcal{X}$ but that are specific for some set of predicates. In particular, Eq. 6.3 and Eq. 6.4 enforce $1_{f_i} \subseteq \mathcal{X}_{\psi_{h(i)}}$ and $1_{f_i} = \mathcal{X}_{\psi_{h(i)}}$ respectively (for all $i \in P$ and $i \in Q$), being 1_{f_i} the characteristic function associated to regions where f_i is active. From a logic point of view, we get the validity of the following FOL formulas:

$$\forall x \in X,\ \hat{f}_i(x) \rightarrow \hat{\psi}_{h(i)}(x) \quad \text{for } i \in P\,, \tag{6.6}$$

$$\forall x \in X,\ \hat{f}_i(x) \leftrightarrow \hat{\psi}_{h(i)}(x) \quad \text{for } i \in Q\,, \tag{6.7}$$

where $\rightarrow$ and $\leftrightarrow$ are the implication and logical equivalence, respectively, and $\hat{f}_i$ is the boolean approximation of f_i.

6.3.3 Learning strategies

Keeping the fan-in of each neuron in the ψ-networks close to small values is a condition that is needed in order to efficiently devise FOL formulas. L_1-norm-based regularization can be exploited to reduce the number of non-zero-weighed input connections of each neuron. After the training stage, we propose to progressively prune the connections with the smallest absolute values of the associated weights, in order to keep exactly $q \geq 2$ input connections per neuron. This process is performed in an iterative fashion. At each iteration, only one connection per neuron is removed, and a few optimization epochs are performed (using the same loss of the training stage), to let the weights of the ψ functions to re-adapt after the weight removal. We repeat this process until all the neurons are left with q input connections.

Globally training the whole model involves optimizing time the weights of the f- and ψ-networks. However, this might lead to low-quality solutions, since the criteria of Section 6.3.1 might have a dominating role in the optimization of Eq. 6.1. We propose to initially train only the f-networks using the available supervisions and the cross-entropy loss, for E epochs. Then, once the selected criteria of Section 6.3.1 are added to the cost function, both the f and ψ-networks are jointly trained (*Global* Optimization). After a first experimentation, we found to be even more efficient to further specialize the latter training, alternating the optimization of the f and ψ-networks. Therefore, we propose a (*Stage-based* Optimization) procedure (formalized in Algorithm 1) in which we first learn the task functions subject to the given constraints only (stage 1 – only $U(f)$ is involved) for N_f epochs, and then we learn new constraints in the task space, keeping the task functions fixed (stage 2 – only $D(\psi, f)$ is involved), for N_ψ epochs. Afterwards, we apply an iterative process

Algorithm 1 Stage-based Optimization procedure (superscripts indicate the iteration number). In order to simplify the notation, here we do not make explicit the dependence of objective functions U and D on the data.

initialize: $f^{(0)}, \psi^{(0)}$ ▷ initialization of the network weights
$f^{(1)} \leftarrow \arg\min_f U(f^{(0)})$ ▷ learning the task functions (stage 1)
$\psi^{(1)} \leftarrow \arg\min_\psi \overline{D}(\psi^{(0)}, f^{(1)})$ ▷ learning of constraints (stage 2)
for $t = 1$ to $t = T - 1$ **do**
  $f^{(t+1)} \leftarrow \arg\min_f U(f) + \overline{D}(\psi^{(t)}, f^{(t)})$ ▷ refinement of the task functions
  $\psi^{(t+1)} \leftarrow \arg\min_\psi \overline{D}(\psi^{(t)}, f^{(t+1)})$ ▷ refinement of the learned constraints
end for
output: $f^\star = f^{(T)}, \psi^\star = \psi^{(T)}$

that is based on optimizing the whole Eq. 6.1, alternately keeping fixed either the constraints or the task functions. In other words, we further refine the task functions by the learned constraints (emphasizing their relationships) and, in turn, we refine the constraints by means of the updated task functions (in this case, we do not consider the term $U(f, \mathcal{S})$, since it is not function of ψ). This procedure is repeated $T - 1$ times, and it is guaranteed to converge since we keep minimizing the same functional. At the end of Section 6.4.1, we report an experiment comparing the two learning strategies and showing the advantages of the stage-based one.

6.4 Experiments

We considered two different tasks, the joint recognition of objects and objects parts in the PASCAL-Part dataset[1], and the recognition of face attributes in portrait images of the CelebA dataset.[2] In both cases, we compared the quality of the plain classifier (*Baseline*), against the classifiers augmented with the explanation networks.

Experimental Setup. According to Section 6.3.3, we set $E = 25$, and then 3 learning stages ($T = 4$) are performed, each of them composed of $N_f = 20$ epochs for the f-network (stage > 1) and $N_\psi = 5$ epochs for the ψ-network. For a fair comparison, the baseline classifier is trained for 100 epochs. Each dataset was divided into training, validation, test sets, and we report the (macro) F1 scores measured on the test data. All the main hyperparameters (weights of terms composing the learning criteria of Section 6.3.1, initial learning rate (Adam optimizer, mini-batch-based stochastic gradient), contribute of the weight decay) have been chosen through a

[1]PASCAL-Part: `https://www.cs.stanford.edu/~roozbeh/pascal-parts/pascal-parts`.
[2]CelebA: `http://mmlab.ie.cuhk.edu.hk/projects/CelebA`.

grid search procedure, with values ranging in $[10^{-1}, 10^{-4}]$, selecting the best model through cross-validation. Results are averaged over 5 different runs.

Each neuron is forced to keep only $q = 2$ input connections in the ψ-network. Deeper ψ-networks are capable of providing more complex explanations, since the compositional structure of the network can relate multiple predicates. We considered two types of ψ-networks, with one or two hidden layers (10 units each), respectively, with the exception of the case of MI in which we considered no-hidden layers or one hidden layer (10 units). This is due to the unsupervised nature of the MI criterion, that, when implemented in deeper networks might capture more complex regularities that are harder to evaluate for a human.

When class-driven criteria are exploited, we considered an independent neural network to implement each ψ_j associated to a driving class. The input space of each of them is different, due to the masking of the driving task function, as described in Section 6.3.1. When considering the MI criterion only, we used a single ψ-network with a number of output units m (one for each ψ_j) ranging from 10 to 50.

PASCAL-Part. This dataset is the same introduced in Chapter 4, but used in a multilabel classification task as in Chapter 5. We divided the dataset into three splits, composed of 9,092 training images, 505 validation images, 506 test images, respectively (keeping the original class distribution). From each image, we extracted 2048 features using a ResNet50 backbone network pretrained on ImageNet. We used 100 hidden units and r output units to implement the f-network.

We tested two different semi-supervised settings in which 10 and 100 labeled examples per-class are respectively provided. The remaining portion of the training data is left unlabeled (it is exploited by the learning criteria of Section 6.3.1). In the class-driven cases, we considered the main objects as driving classes, so $m = 16$. Results are reported in Table 6.1, in which the F1 scores (upper portion) and a sample of the extracted rules (lower portion) are shown. The proposed learning criteria lead to an improvement of the classifier performance that is more evident when less supervisions are provided, as expected. We further explored this result, distinguishing between the F1 measured (a) on the driving classes, that are more represented, and (b) on the other classes. Fig. 6.4 (top) shows that evident improvements (w.r.t. the baseline) can sometimes be due to only one of the two groups of classes, and there is not a clear trend among the criteria. Notice that class-driven criterion not necessarily leads to better driving-task-functions, while it can also improve the other functions. This is because some driving classes might also participate in explaining other driving classes.

The explanations in Table 6.1 show that deeper ψ networks usually lead to more complex formulas, as expected. *Local Explanations* depend on the regions covered by the $\mathcal{X}_{\psi_j}$, and they sometimes involve semantically related classes, that might be

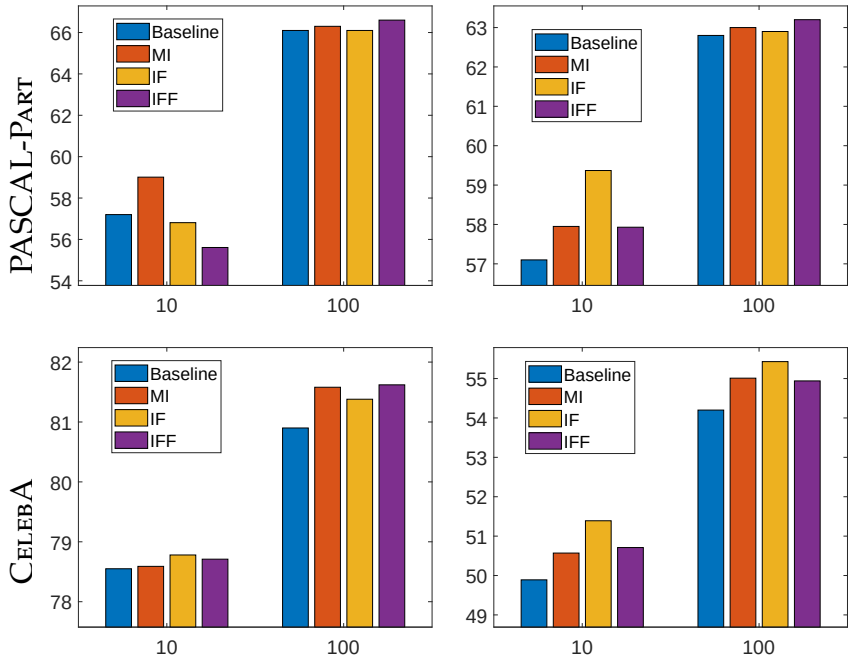


Figure 6.4: F1 score % on (a) the driving classes and (b) on the other classes, in function of the number of labeled examples per class.

simultaneously active on the same region. *Global Explanations* show possible coverings of the whole classifier output space. We only show 2 sample $\hat{\psi}_j'$'s from Eq. 6.5. They might be harder to follow, since they merge multiple local explanations. In the deeper case we get more compact terms, that, however, are more numerous, i.e., larger K. *Class-driven Explanations* $IF\rightarrow$ and $IFF\leftrightarrow$ provide a semantically coherent description of objects and their parts. Interestingly, these rules usually implement reasonable expectations on this task, with a few exceptions. The $IFF\leftrightarrow$ case is more restrictive than $IF\rightarrow$ (compare *Car, Bicycle* in the two cases).

CelebA. This dataset is composed of over 200k images of celebrity faces, out of which 45% are used as training data, 5% as validation data and $\approx$ 100k are used for testing. The dataset is composed of 40 annotated attributes (classes) per image (*BlondHair, Sideburns, GrayHair, WavyHair, etc.*), that we extended by adding the attributes *NotAttractive, NotBald, Female, Beard, Old*, as opposite of the already existing *Attractive, Bald, Male, NoBeard, Young*. In the class-driven criteria, these two sets of attributes are the ones we require to explain ($m = 10$). We exploit the same preprocessing and neural architectures of the previous experiment, evaluating semi-supervised settings with 25 and 100 labeled examples per class. Results are reported in Table 6.2 and Fig. 6.4 (bottom).

We obtained a slightly less evident improvement of the performance with re-

Table 6.1: PASCAL-Part dataset. Top: macro F1 scores % ($\pm$standard deviation), different learning settings and number of labeled points *per-class*. Bottom: explanations yielded in different scenarios (two types of ψ-network). Functions $\hat{f}_i$'s are indicated with their class-names.

# Labeled	**Baseline**	**MI**	**IF$\rightarrow$**	**IFF$\leftrightarrow$**
10	57.0 ± 0.3	58.1 ± 0.2	$\mathbf{58.5 \pm 0.2}$	57.1 ± 0.1
100	63.5 ± 0.2	63.7 ± 0.2	63.6 ± 0.2	$\mathbf{63.9 \pm 0.1}$

Scenario	**Explanations**	**Explanations** (DEEPER ψ)
LOCAL	$\forall x \in X_i$, *Beak* $\vee$ *Bird* $\forall x \in X_j$, *Headlight* $\vee$ *Plate* $\forall x \in X_k$, *Cat* $\vee$ *Horse*	$\forall x \in X_i$, *Bottle* $\vee$ *Table* $\forall x \in X_j$, *Arm* $\wedge \neg$*Bottle* $\wedge \neg$*Horn* $\wedge \neg$*Table* $\forall x \in X_k$, $\neg$*Bottle* $\wedge \neg$*Table* $\wedge$ (*Car* $\vee$ *Motorbike*)
GLOBAL	$\forall x$, *AeroplaneBody* $\vee$ *Beak* $\vee$ *Bird* $\vee$ *Table* $\vee$*Car* $\vee$ *Headlight* $\vee$ *Motorbike* $\vee$ *Muzzle* $\vee$*Train* $\vee$ *Chainwheel* $\vee \neg$*Aeroplane* $\vee$ *Plant* $\forall x, \neg$*Horse* $\vee$ *AeroplaneBody* $\vee$ *Beak* $\vee$ *Bird* $\vee$*Car* $\vee$ *Chainwheel* $\vee$ *Headlight* $\vee$ *Muzzle* $\vee$*Table* $\vee$ *Motorbike* $\vee$ *Plant* $\vee$ *Train*	$\forall x$, *Bird* $\vee$ *Coach* $\vee$ *Hand* $\vee$ *Nose* $\vee$*Sheep* $\vee$ *Stern* $\vee$ *Wheel* $\vee \neg$*Roofside* $\forall x$, $\neg$*Saddle* $\vee$ *Bird* $\vee$ *Coach* $\vee$ *Hand* $\vee$ *Nose* $\vee$*Sheep* $\vee$ *Stern* $\vee$ *Wheel*
CLASS-DRIVEN $IF \rightarrow$	$\forall x$, *Car* $\rightarrow$ *Backside* $\vee$ *Mirror* $\vee$(*Window* $\wedge \neg$*Coach*) $\forall x$, *Bicycle* $\rightarrow$ *Saddle* $\vee$ *Handlebar* $\forall x$, *Train* $\rightarrow$ *Coach* $\vee$ *TrainHead*	$\forall x$, *Aeroplane* $\rightarrow$ *Engine* $\vee$ *Stern* $\forall x$, *Chair* $\rightarrow$ (*Table* $\wedge$ *Sofa*) $\vee$ (*Table* $\wedge \neg$*Door*) $\forall x$, *Boat* $\rightarrow \neg$*Bottle* $\wedge \neg$*Cat* $\wedge \neg$*Coach* $\wedge \neg$*Leftside* $\wedge \neg$*Paw* $\wedge \neg$*Wheel* $\wedge \neg$*Wing*
CLASS-DRIVEN $IFF \leftrightarrow$	$\forall x$, *Horse* $\leftrightarrow$ (*Hoof* $\wedge$ *Ear*) $\vee$ (*Hoof* $\wedge$ *Neck*) $\forall x$, *Bird* $\leftrightarrow$ *Beak* $\wedge \neg$*Horn* $\forall x$, *Bicycle* $\leftrightarrow$ (*Chainwheel* $\wedge \neg$*Cow* $\wedge$ *Handlebar*) $\vee$(*Chainwheel* $\wedge \neg$*Cow* $\wedge$ *Saddle*)	$\forall x$, *Aeroplane* $\leftrightarrow$ *AeroplaneBody* $\wedge \neg$*Horn* $\forall x$, *Car* $\leftrightarrow$ *Door* $\vee$ *Mirror* $\forall x$, *Dog* $\leftrightarrow$ *Muzzle* $\wedge$ *Paw* $\wedge \neg$*Table* $\wedge \neg$*TrainHead*

spect to the baseline, especially in the less-supervised case. This is mostly due to the fact that some classes are associated to high-level attributes (such as *Attractive*) that might be not easy to generalize from a few supervisions. When distinguishing among the results on driving and not-driving classes (Fig. 6.4), improvements are more evident. From the *Local Explanations* in the lower portion of Table 6.2, we can appreciate that some rules are able to capture in a fully unsupervised way the relationships between, for example, being *Attractive* and *Young*, or being *Old* and with *GrayHair*. *Global Explanations* show more differentiated coverings of the classifier output space. *Class-driven Explanations IF*$\rightarrow$ and *IFF*$\leftrightarrow$ yield descriptions that, again, are usually in line with common expectations (see *Beard*, *Bald*, *Male*).

Table 6.2: CelebA dataset. Top: macro F1 scores % ($\pm$standard deviation), different learning settings and number of labeled points *per-class*. Bottom: explanations yielded in different scenarios (two types of ψ-network). Functions $\hat{f}_i$'s are indicated with their class-names.

# Labeled	**Baseline**	**MI**	**IF$\rightarrow$**	**IFF$\leftrightarrow$**
25	54.7 ± 0.4	55.0 ± 0.3	$\mathbf{56.1} \pm \mathbf{0.1}$	55.1 ± 0.2
100	60.0 ± 0.1	60.4 ± 0.2	$\mathbf{60.9} \pm 0.2$	60.5 ± 0.2

Scenario	**Explanations**	**Explanations** (Deeper ψ)
Local	$\forall x \in X_i$, $Bangs \wedge \neg Bald$ $\forall x \in X_j$, $StraightHair \wedge BushyEyebrows$ $\forall x \in X_k$, $Female \wedge Attractive$	$\forall x \in X_i$, $BlackHair \wedge Attractive \wedge Young$ $\forall x \in X_j$, $(Old \wedge GrayHair) \vee (Old \wedge \neg Young)$ $\forall x \in X_k$, $NoBeard \wedge Female \wedge \neg WearNecktie$
Global	$\forall x$, $Bangs \vee BlondHair \vee Blurry \vee Goatee \vee StraightHair \vee WearHat \vee \neg Attractive \vee \neg Female \vee \neg Male$ $\forall x$, $Blurry \vee Goatee \vee WearHat \vee \neg Attractive \vee \neg BlackHair \vee \neg BlondHair \vee \neg Female \vee \neg StraightHair$	$\forall x$, $Beard \vee BlackHair \vee BrownHair \vee Goatee \vee HeavyMakeup \vee Mustache \vee Old \vee WearNecktie \vee \neg Beard \vee \neg Young$ $\forall x$, $HeavyMakeup \vee Mustache \vee WearNecktie \vee Young \vee \neg Beard \vee \neg WearLipstick$
Class-driven $IF \rightarrow$	$\forall x$, $Attractive \rightarrow PaleSkin \vee RosyCheeks \vee (\neg Blurry \wedge \neg Chubby)$ $\forall x$, $Beard \rightarrow Goatee \vee Sideburns$ $\forall x$, $Old \rightarrow GrayHair \vee \neg Attractive$	$\forall x$, $Male \rightarrow Beard \vee FiveOClockShadow \vee DoubleChin \vee \neg WearLipstick$ $\forall x$, $Bald \rightarrow RecedingHairline \wedge \neg Bangs \wedge \neg RosyCheeks \wedge \neg WavyHair$ $\forall x$, $Female \rightarrow HeavyMakeup \vee WearLipstick \vee (\neg DoubleChin \wedge \neg WearNecktie)$
Class-driven $IFF \leftrightarrow$	$\forall x$, $Bald \leftrightarrow \neg BlackHair \wedge \neg BrownHair \wedge \neg StraightHair \wedge \neg WavyHair$ $\forall x$, $NotBald \leftrightarrow Bangs \vee BrownHair \vee WavyHair$ $\forall x$, $Male \leftrightarrow \neg WearLipstick \wedge \neg WearNecklace$	$\forall x$, $Beard \leftrightarrow (Goatee \wedge Mustache) \vee (Goatee \wedge Sideburns)$ $\forall x$, $Bald \leftrightarrow \neg Bangs \wedge \neg StraightHair \wedge \neg WavyHair$ $\forall x$, $Young \leftrightarrow (\neg GrayHair \wedge BigLips) \vee (\neg GrayHair \wedge \neg WearNecklace)$

6.4.1 Learning comparison

We here analyse the training performances of the learning configurations described in Section 6.3.3, the *Stage-based* Optimization and the *Global* Optimization, together with a Baseline represented by learning the f function only. These strategies have been compared under the same conditions, i.e., same hyperparameters, labelled data and network initialization and extracting *Local Explanations* (therefore maximizing the MI as defined in Eq. 6.2). Fig. 6.5 reports the F1 growth in the case of the PASCAL-Part experiment when 100 labelled samples per class are given as supervision. For both strategies, we can observe an improvement of the performances over the Baseline. The stage-based performance evolution is particularly interesting to inspect: after 20 epochs, the learning *of constraints* stage begins (the one that involves

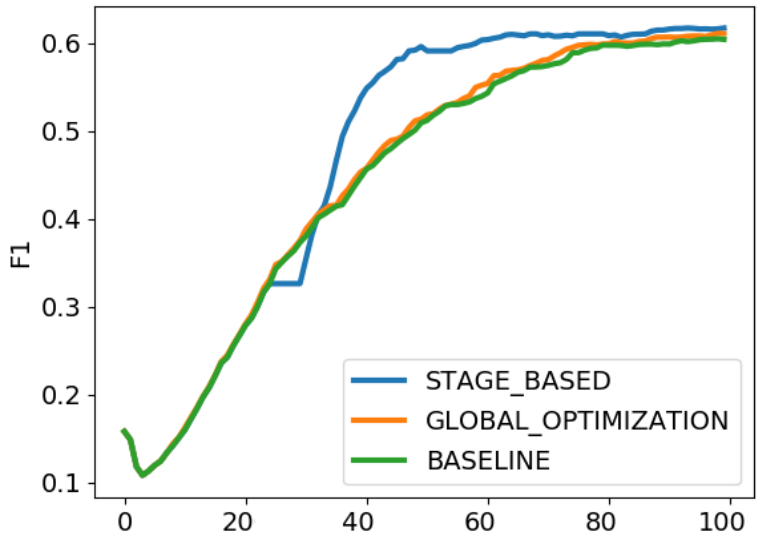


Figure 6.5: Evolution of the training performances when increasing the number of epochs, comparing the different learning strategies on the PASCAL-Part problem with 100 labelled samples per class and over a single seed. The *Stage-based* optimization boost the performances due to the regularization effect of D_{MI} in Eq. 6.1.

the MI), and for the following 5 epochs the F1 score does not improve as the task functions are not modified. Nonetheless, in a few epochs the regularization effects imposed by the D_{MI} term in Eq. 6.1 boost the performances of the network, overcoming those of the other two networks. This process is repeated every 25 epochs with smaller effects, leading to higher final F1 score for the *Stage-based* optimization with respect to the others.

6.5 Related Work

In the last few years, the scientific community devoted a lot of effort to the proposal of approaches that yield explanations to the decisions of machine learning-based systems (Bibal and Frénay, 2016; Doshi-Velez and Kim, 2017; Došilović et al., 2018; Guidotti et al., 2018b; Teso and Kersting, 2019). In particular, several Explainable Artificial Intelligence (XAI) (Gunning, 2017) techniques have been developed, with different properties and output formats. They generally rely on existing interpretable models, such as decision trees, rules, linear models (Freitas, 2014; Huysmans et al., 2011), that are considered easily understandable by humans. On the other hand, in order to provide an explanation for black-box predictors, such as (deep) neural networks and support vector machines, a new interpretable model that is as faithful as possible to the original predictor is considered, sometimes acting on localized regions of the space (Guidotti et al., 2018b). Then, the explanation

problem consists in finding the best interpretable model approximating the black-box predictor. In the context of the XAI literature, there is no clear agreement on what an explanation should be, nor on what are the suitable methodologies to quantitatively evaluate its quality (Carvalho et al., 2019; Molnar, 2020). There is also a strong dependence on the target of the explanation, e.g., a common user, an expert, or an artificial intelligence researcher. In this regard, the framework proposed in this chapter allows the final user to choose among different kinds of explanations, even if assuming a first-order logic language.

Symbolic interpretation of neural networks (rule extraction) has been the subject of many researches by several authors, especially in the nineties. Some approaches are about Fuzzy Logic Kasabov (1996); Huang and Xing (2002); Castro and Trillas (1998); Di Nola et al. (2013), but are generally less straightforward in terms of explainability than the ones based on Boolean Logic Fu (1991); Towell and Shavlik (1993); Tsukimoto (2000); Sato and Tsukimoto (2001). In the latter case, it is pretty common to rely on a discretization of the input and output values of the neurons, pruning the network to keep it simple. The interpretation we propose in this chapter follows this approach, mostly in the spirit of Zilke et al. (2016) where Boolean interpretation is applied neuron-by-neuron to a deep neural network that is trained to solve a specific task, decomposing the interpretation of the whole network to its sub-constituents. However, in this chapter we followed a different paradigm, both developing a set of task functions and another network that, in turn, learns how the tasks are related (constrained), with the aim of extracting logic explanations.

Chapter 7

An Explainable-by-Design Network

In this chapter we propose a different solution to the explainability issue described in Section 2.3.3, by introducing a neural network which both solves the learning problem and provide explanations of its predictions. Part of the content of this chapter is extracted from the works Ciravegna et al. (2021) and Barbiero et al. (2021b) that we submitted to the Artificial Intelligence journal and to the AAAI 2022 conference respectively, together with some colleagues from the University of Siena and Cambridge.

In this chapter, we first propose an entropy-based layer (Section 7.2.1) that enables the implementation of *concept-based* neural networks, providing First-Order Logic explanations (Fig. 7.1). Second, we describe how to interpret the predictions of the proposed neural model to distil logic explanations for individual observations and for a whole target class (Section 7.2.3). We therefore define our approach as *explainable by design*, since the proposed architecture allows neural networks to automatically provide logic explanations of their predictions. Finally, we demonstrate how our approach provides high-quality explanations according to six *quantitative* metrics while outperforming some state-of-the-art white-box models (later described in Section 7.4) in terms of classification accuracy on four case studies (Section 7.3).

7.1 Introduction

Following the notation introduced in Section 3.1, we consider again the classification problem $X \rightarrow Y$ and to extract logic constraints from a classifier f. However, in this chapter we aim at providing explanations of the prediction for a certain class in terms of the input features rather than other classes. Therefore, input samples $x \in X$ are first mapped to a concept space C that represent symbolic concepts. We then consider the family of Concept-based classifiers f, machine learning models predicting class memberships from the activation scores of h human-understandable

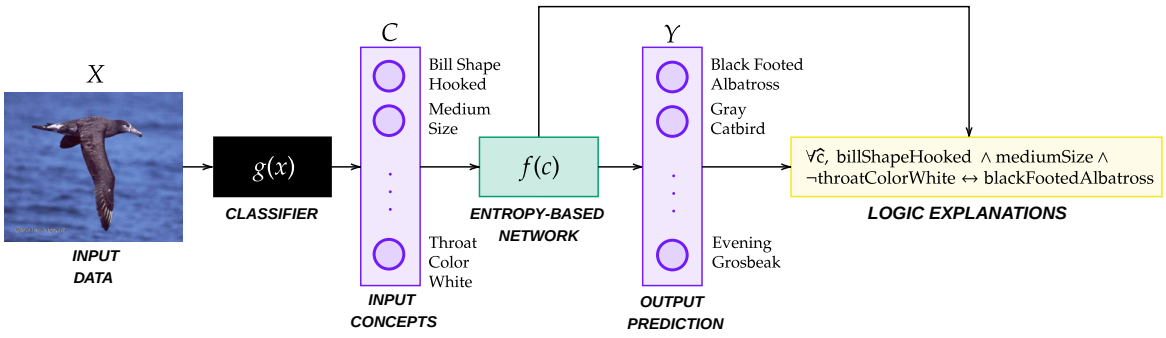


Figure 7.1: The proposed pipeline on one example from the CUB dataset. The proposed neural network maps concepts onto target classes $f: C \mapsto Y$ and provide concise logic explanations (yellow—we dropped the arguments in the logic predicates, for simplicity) of its own decision process. When the input features are non-interpretable (as pixels intensities), a classifier $g: X \mapsto C$ maps inputs to concepts.

categories, $f : C \mapsto Y$, where $C \subset [0,1]^h$ (see Fig. 7.1). When observations are represented in terms of non-interpretable input features belonging to $X \subset \mathbb{R}^d$ (such as pixels intensities), a "concept decoder" g (similarly to what proposed in (Koh et al., 2020) is used to map the input into a concept-based space, $g : X \mapsto C$ (see Fig. 7.1). Otherwise, they are simply rescaled from the unbounded space $\mathbb{R}^d$ into the unit interval $[0,1]^h$, such that input features can be treated as logic predicates.

A similar method related to the proposed approach is the ψ network[1] introduced in Chapter 6 (Ciravegna et al., 2020a,b), an end-to-end differentiable model explaining the predictions of another network. However, we here aim at creating an *explainable-by design* classifier f per se rather than explaining another classifier. Since the decision process of the ψ is interpretable, we directly could use them to classify, optimizing both terms of Eq. 6.1 with the same network and asking for *Class-driven* explanations of its own predictions. However, the ψ network applies an $L1$-regularization and a strong pruning strategy to each layer of the network to allow the computation of logic formulas representing the activation of each node. Also, it requires employing a sequence of fully connected layers with sigmoid activations only. Such constraints limit the learning capacity of the model when used as a standard classifier and impair its classification accuracy (as we will see in Section 7.3). Therefore, in this chapter, we propose here a variant of the ψ network that leverages the intermediate symbolic layer C to distil First-Order Logic formulas, representing the learned mapping from C to Y. More precisely, we propose an entropy-based approach which allows the extraction of concise FOL explanations without severely impairing the learning capability of the network. The approach does not impose constraints on the form of activation functions nor weights' pruning. As we will

[1] In this chapter we indicate with ψ the model introduced in Chapter 6, *not* the generic function mapping $Y \to Z$ defined in Chapter 3.

better see in the following, the method only encourages the classifier f to focus on a limited subset of concepts (Section 7.2.1) for each task Y^i for which it provides an explanation. The proposed method provides state-of-the art classification accuracy and high-quality explanations according to several metrics when compared to standard interpretable-by-design machine learning models.

7.2 Entropy-based Logic Explanations of Neural Networks

The key contribution of this chapter is a novel linear layer enabling entropy-based logic explanations of neural networks (see Fig. 7.2). This layer receives in input samples from the concept space C, while the outcomes of the layer computations are: (i) the embeddings h^i (as any linear layer), (ii) a truth table $\mathcal{T}^i$ explaining how the network leveraged concepts to make predictions for the i-th target class. Each class of the problem requires an independent entropy-based layer, as emphasized by the superscript i. For ease of reading and without loss of generality, all the following descriptions concern inference for a single observation (corresponding to the concept tuple $c \in C$) and a neural network f^i predicting the class memberships for the i-th class of the problem. For multi-class problems, multiple "heads" of this layer are instantiated, with one "head" per target class (see Section 7.3), and the hidden layers of the class-specific networks could be eventually shared.

7.2.1 Entropy-based linear layer

When humans compare a set of hypotheses outlining the same outcomes, they tend to have an implicit bias towards the simplest ones as outlined in philosophy (Aristotle, 0 BC; Hoffmann et al., 1996; Soklakov, 2002; Rathmanner and Hutter, 2011), psychology (Miller, 1956; Cowan, 2001), and decision making (Simon, 1956, 1957, 1979). The proposed entropy-based approach encodes this inductive bias in an end-to-end differentiable model. The purpose of the entropy-based linear layer is to encourage the neural model to pick a limited subset of input concepts, allowing it to provide concise explanations of its predictions. The learnable parameters of the layer are the usual weight matrix W and bias vector b. In the following, the forward pass is described by the operations going from Eq. 7.1 to Eq. 7.4 while the generation of the truth tables from which explanations are extracted is formalized by Eq. 7.5 and Eq. 7.6.

The relevance of each input concept can be summarized in a first approximation by a measure that depends on the values of the weights connecting such concept to the upper network. In the case of network f^i (i.e. predicting the i-th class) and of the j-th input concept, we indicate with W_j^i the vector of weights departing from the

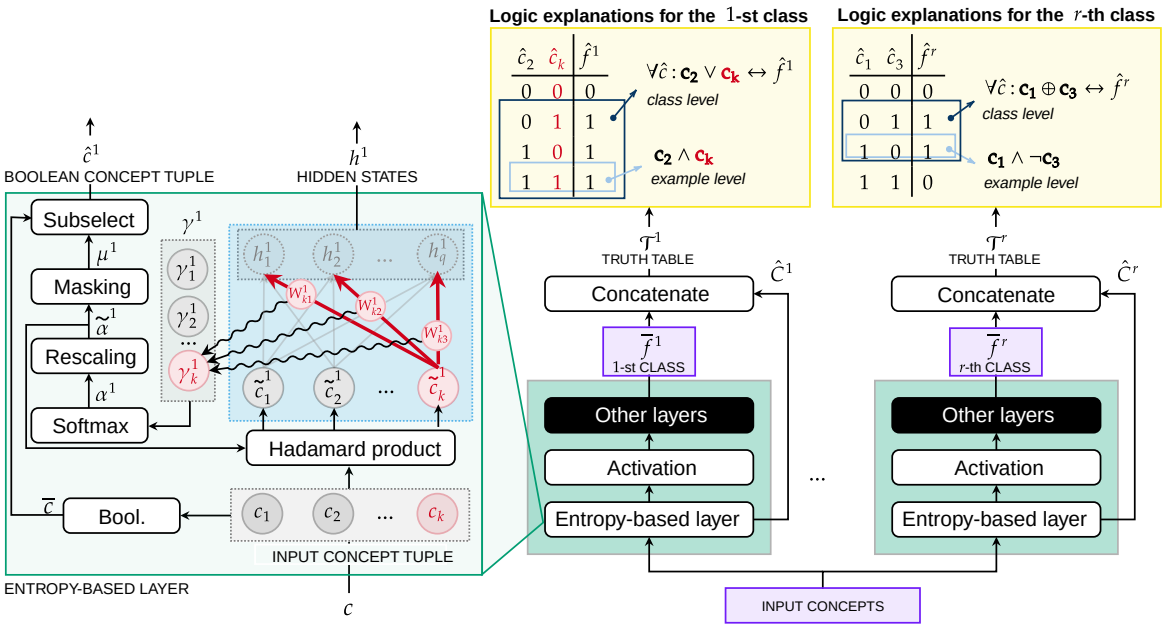


Figure 7.2: On the right, the proposed neural network learns the function $f : C \mapsto Y$. For each class, the network leverages one "head" of the entropy-based linear layer (green) as first layer. For each target class i, the network provides: the class membership predictions f^i and the truth table $\mathcal{T}^i$ (Eq. 7.6) to distil FOL explanations (yellows, top). On the left, a detailed view on the entropy-based linear layer for the 1-st class, emphasizing the role of the k-th input concept as example: (i) the scalar γ_k^1 (Eq. 7.1) is computed from the set of weights connecting the k-th input concept to the output neurons of the entropy-based layer; (ii) the relative importance of each concept is summarized by the categorical distribution α^1 (Eq. 7.2); (iii) rescaled relevance scores $\tilde{\alpha}^1$ drop irrelevant input concepts out (Eq. 7.3); (iv) hidden states h^1 (Eq. 7.4) and Boolean-like concepts $\hat{c}^1$ (Eq. 7.5) are provided as outputs of the entropy-based layer.

j-th input (see Fig 7.2), and we introduce

$$\gamma_j^i = ||W_j^i||_1 \,. \tag{7.1}$$

The higher γ_j^i, the higher the relevance of the concept j for the network f^i. In the limit case, $\gamma_j^i \to 0$, the model f^i drops the j-th concept out. To select only few relevant concepts for each target class, concepts are set up to compete against each other. To this aim, the relative importance of each concept to the i-th class is summarized in the categorical distribution α^i, composed of coefficients $\alpha_j^i \in [0,1]$ (with $\sum_j \alpha_j^i = 1$), modeled by the softmax function:

$$\alpha_j^i = \frac{e^{\gamma_j^i/\tau}}{\sum_{l=1}^{k} e^{\gamma_l^i/\tau}} \tag{7.2}$$

where $\tau \in \mathbb{R}^+$ is a user-defined temperature parameter to tune the intrinsic tendency of the softmax function. For a given set of γ_j^i, when using high temperature values ($\tau \to \infty$) all concepts have nearly the same relevance. For low temperatures values ($\tau \to 0$), the probability of the most relevant concept tends to ≈ 1, while it becomes ≈ 0 for all other concepts. As the probability distribution α^i highlights the most relevant concepts, this information is directly fed back to the input, weighting concepts by the estimated importance. To avoid numerical cancellation due to values in α^i close to zero, especially when the input dimensionality is large, we replace α^i with its normalized instance $\tilde{\alpha}^i$, still $\in [0,1]^h$, and each input sample $c \in C$ is modulated by the (normalized) estimated importance,

$$\tilde{c}^i = c \odot \tilde{\alpha}^i \qquad \text{with} \qquad \tilde{\alpha}_j^i = \frac{\alpha_j^i}{\max_u \alpha_u^i}, \tag{7.3}$$

where $\odot$ denotes the Hadamard (element-wise) product. The highest value in $\tilde{\alpha}^i$ is always 1 (i.e. $\max_j \tilde{\alpha}_j^i = 1$) and it corresponds to the most relevant concept. The embeddings h^i are computed as in any linear layer by means of the affine transformation:

$$h^i = W^i \tilde{c}^i + b^i. \tag{7.4}$$

Whenever $\tilde{\alpha}_j^i \to 0$, the input $\tilde{c}_j \to 0$. This means that the corresponding concept tends to be dropped out and the network f^i will learn to predict the i-th class without relying on the j-th concept.

In order to get logic explanations, the proposed linear layer generates the truth table $\mathcal{T}^i$ formally representing the behaviour of the neural network in terms of Boolean-like representations of the input concepts. In detail, we indicate with $\bar{c}$ the Boolean interpretation of the input tuple $c \in C$, while $\mu^i \in \{0,1\}^h$ is the binary mask associated to $\tilde{\alpha}^i$. To encode the inductive human bias towards simple explanations (Miller, 1956; Cowan, 2001; Ma et al., 2014), the mask μ^i is used to generate the binary concept tuple $\hat{c}^i$, dropping the least relevant concepts out of c,

$$\hat{c}^i = \xi(\bar{c}, \mu^i) \qquad \text{with} \qquad \mu^i = \mathbb{I}_{\tilde{\alpha}^i \geq \epsilon} \qquad \text{and} \qquad \bar{c} = \mathbb{I}_{c \geq \epsilon}, \tag{7.5}$$

where $\mathbb{I}_{z \geq \epsilon}$ denotes the indicator function that is 1 for all the components of vector z being $\geq \epsilon$ and 0 otherwise (considering the unbiased case, we set $\epsilon = 0.5$). The function ξ returns the vector with the components of $\bar{c}$ that correspond to 1's in μ^i (i.e. it sub-selects the data in $\bar{c}$). As a results, $\hat{c}^i$ belongs to a space $\hat{C}^i$ of m_i Boolean features, with $m_i < h$ due to the effects of the subselection procedure.

The truth table $\mathcal{T}^i$ is a particular way of representing the behaviour of network f^i based on the outcomes of processing multiple input samples collected in a generic dataset $\mathcal{C}$. As the truth table involves Boolean data, we denote with $\hat{\mathcal{C}}^i$ the set with the Boolean-like representations of the samples in $\mathcal{C}$ computed by ξ, Eq. 7.5. We also

introduce $\bar{f}^i(c)$ as the Boolean-like representation of the network output, $\bar{f}^i(c) = \mathbb{I}_{f^i(c) \geq \epsilon}$. From an operational perspective, the contents $\mathbf{T}^i$ of the truth table $\mathcal{T}^i$ are obtained by stacking data of $\hat{\mathcal{C}}^i$ into a 2D matrix $\hat{\mathbf{C}}^i$ (row-wise), and concatenating the result with the column vector $\bar{\mathbf{f}}^i$ whose elements are $\bar{f}^i(c)$, $c \in \mathcal{C}$:

$$\mathbf{T}^i = \left(\hat{\mathbf{C}}^i \,\middle|\middle|\, \bar{\mathbf{f}}^i \right). \tag{7.6}$$

$\mathcal{T}^i$ is used to generate logic explanations, as we will explain in Section 7.2.3.

7.2.2 Loss function

The entropy of the probability distribution α^i (Eq. 7.2),

$$\mathcal{H}(\alpha^i) = -\sum_{j=1}^{k} \alpha_j^i \log \alpha_j^i \tag{7.7}$$

is minimized when a single α_j^i is one, thus representing the extreme case in which only one concept matters, while it is maximum when all concepts are equally important. We jointly minimize $\mathcal{H}$ with the usual loss function about the knowledge on the supervision $\varphi(f, \mathcal{K}_{\int}, \mathcal{S})$ (being $\mathcal{S} = \mathcal{C}$, since we are in a fully supervised learning scenario). This allows the model to find a trade-off between fitting quality and a parsimonious activation of the concepts, allowing each network f^i to predict i-th class memberships using few relevant concepts only. Overall, the loss function to train the network f is defined as,

$$\mathcal{L}(f, y, \alpha_1, \ldots, \alpha_r) = \varphi(f, \mathcal{K}_{\int}, \mathcal{S}) + \lambda \sum_{i=1}^{r} \mathcal{H}(\alpha^i), \tag{7.8}$$

where $\lambda > 0$ is the hyperparameter used to balance the relative importance of low-entropy solutions in the loss function. Higher values of λ lead to sparser configuration of α, constraining the network to focus on a smaller set of concepts for each classification task (and vice versa), thus encoding the inductive human bias towards simple explanations (Miller, 1956; Cowan, 2001; Ma et al., 2014).

7.2.3 First-order logic explanations

Any Boolean function can be converted into a logic formula in Disjunctive Normal Form (DNF) by means of its truth-table (Mendelson, 2009). We indicate with $\hat{f}^i$ the Boolean function represented by the truth table $\mathcal{T}^i$, $\hat{f}^i : \hat{C}^i \mapsto Y^i$, being Y^i the i-th component of Y. Converting a truth table into a DNF formula provides an effective mechanism to extract logic rules of increasing complexity from individual observations to a whole class of samples. The following rule extraction mechanism is considered for each task i.

FOL extraction. Each row of the truth table $\mathcal{T}^i$ can be partitioned into two parts that are a binary tuple of concept activations, $\hat{q} \in \hat{C}^i$, and the outcome of $\hat{f}^i(\hat{q}) \in \{0,1\}$. An *example-level* logic formula, consisting in a single minterm, can be trivially extracted from each row for which $\hat{f}^i(\hat{q}) = 1$, by simply connecting with the logic AND $\wedge$ the true concepts and negated instances of the false ones. The logic formula becomes human understandable whenever concepts appearing in such a formula are replaced with human-interpretable strings that represent their name (similar consideration holds for $\hat{f}^i$, in what follows). For example, the following logic formula φ_t^i,

$$\varphi_t^i = \mathbf{c}_1 \wedge \neg \mathbf{c}_2 \wedge \ldots \wedge \mathbf{c}_{m_i}, \tag{7.9}$$

is the *example-level* formula extracted from the t-th row of the table where, in the considered example, only the second concept is false, being $\mathbf{c}_z$ the name of the z-th concept. Example-level formulas can be aggregated with the logic OR $\vee$ to provide a *class-level* formula,

$$\bigvee_{t \in S_i} \varphi_t^i, \tag{7.10}$$

being S_i the set of rows indices of the truth table for which $\hat{f}^i(\hat{q}) = 1$, i.e. it is the support of $\hat{f}^i$. We define with $\phi^i(\hat{c})$ the function that holds true whenever Eq. 7.10, evaluated on a given Boolean tuple $\hat{c}$, is true. Due to the aforementioned definition of support, we get the following class-level First-Order Logic (FOL) explanation for all the concept tuples,

$$\forall \hat{c} \in \hat{C}^i : \ \phi^i(\hat{c}) \leftrightarrow \hat{f}^i(\hat{c}). \tag{7.11}$$

We note that in case of non-concept-like input features, we may still derive the FOL formula through the "concept decoder" function g (see Section 7.1),

$$\forall x \in X : \ \phi^i \left(\xi(\overline{g(x)}, \mu^i) \right) \leftrightarrow \hat{f}^i \left(\xi(\overline{g(x)}, \mu^i) \right) , \tag{7.12}$$

where $\xi(\overline{g(x)\mu^i}) = \hat{c}$ and is the result of the selection and the boolenization process over the output of g. An example of the above scheme for both example and class-level explanations is depicted on the top-right of Fig. 7.2.

Remarks. The aggregation of many example-level explanations may increase the length and the complexity of the FOL formula being extracted for a whole class. However, existing techniques as the Quine–McCluskey algorithm can be used to get compact and simplified equivalent FOL expressions (McColl, 1878; Quine, 1952; McCluskey, 1956). For instance, the explanation $(person \wedge nose) \vee (\neg person \wedge nose)$ can be formally simplified in *nose*. Moreover, the Boolean interpretation of concept tuples may generate colliding representations for different samples. For instance, the Boolean representation of the two samples $\{(0.1, 0.7), (0.2, 0.9)\}$ is the tuple $\bar{c} = (0,1)$ for both of them. This means that their example-level explanations

match as well. However, a concept can be eventually split into multiple finer grain concepts to avoid collisions. Finally, we mention that the number of samples for which any example-level formula holds (i.e. the support of the formula) is used as a measure of the explanation importance. In practice, example-level formulas are ranked by support and iteratively aggregated to extract class-level explanations, until the aggregation improves the support of the formula on a validation set.

7.3 Experiments

The quality of the explanations and the classification performance of the proposed approach are quantitatively assessed. The proposed approach is also compared with state-of-the-art white-box methods providing logic-based, global explanations (see Section 7.4). In particular, we selected one representative approach from different families of methods: Decision Trees[2] (white-box machine learning), BRL[3] (rule mining) and ψ Networks[4] (interpretable neural models).

A visual sketch of each classification problem (described in detail in the following together with all the experimental details) and a selection of the logic formulas found by the proposed approach is reported in Fig. 7.3. Six quantitative metrics are defined and used to compare the proposed approach with state-of-the-art methods. Finally, we summarizes the main findings. A python package[5] and a GitHub repository[6] implementing the proposed approach are publicly available. In appendix A.4.1 a snippet of the code extracted from the library is reported.

7.3.1 Classification tasks and datasets

For the experimental analysis, four classification problems ranging from computer vision to medicine are considered. Computer vision datasets (e.g. CUB) are annotated with high-level concepts (e.g. bird attributes) used to train concept bottleneck pipelines (Koh et al., 2020). In the other datasets, the input data is rescaled into a categorical space ($\mathbb{R}^k \to C$) suitable for concept-based networks.

Will we recover from ICU? (MIMIC-II). The Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II, (Saeed et al., 2011; Goldberger et al., 2000))[7] is a public-access intensive care unit (ICU) database consisting of 32,536 subjects

[2]Decision Trees: `https://scikit-learn.org/stable/modules/tree.html`, BSD-3 Clause License.
[3]BRL: `https://github.com/tmadl/sklearn-expertsys`, MIT license.
[4]ψ Networks: `https://github.com/pietrobarbiero/logic_explainer_networks`, Apache 2.0 License.
[5]Entropy Net PIP: `https://pypi.org/project/pytorch_explain/`
[6]Entropy Net GitHub: `https://github.com/pietrobarbiero/pytorch_explain`
[7]MIMIC-II: `https://archive.physionet.org/mimic2`.

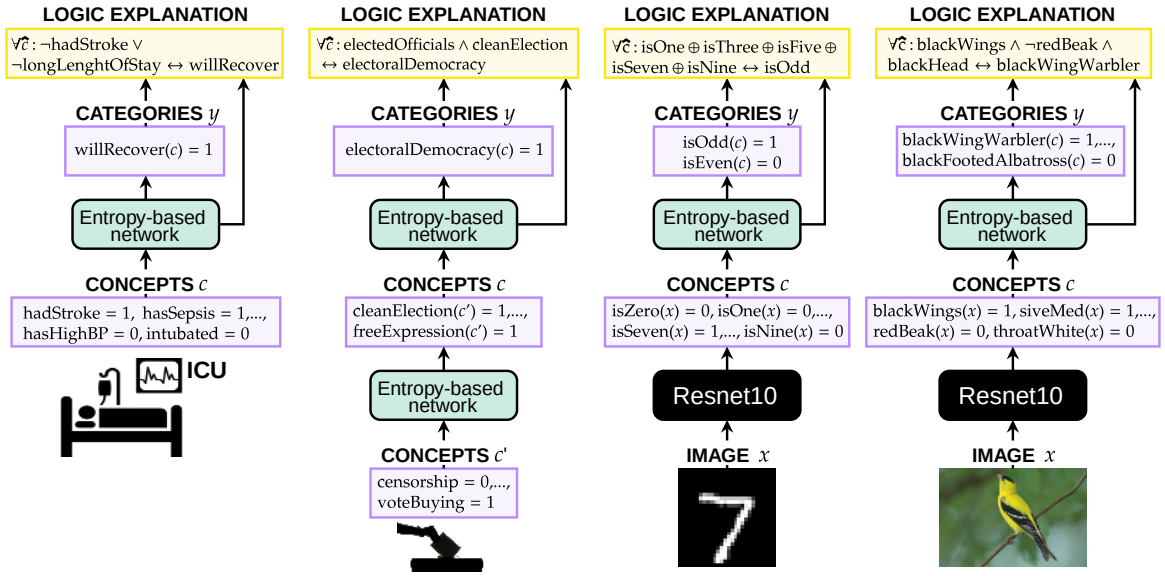


Figure 7.3: The four case studies show how the proposed Entropy-based networks (green) provide concise logic explanations (yellow—we dropped the arguments in the logic predicates, for simplicity) of their own decision process in different real-world contexts. When input features are non-interpretable, as pixel intensities (MNIST and CUB), a "concept decoder" (ResNet10) is employed to map images into concepts. Entropy-based networks then map concepts into target classes.

(with 40,426 ICU admissions) admitted to different ICUs. The dataset contains detailed descriptions of a variety of clinical data classes: general, physiological, results of clinical laboratory tests, records of medications, fluid balance, and text reports of imaging studies (e.g. x-ray, CT, MRI, etc). In our experiments, we removed non-anonymous information, text-based features, time series inputs, and observations with missing data. We discretize continuous features into one-hot encoded categories. After such preprocessing step, we obtained an input space C composed of $h = 90$ key features. The task consists in identifying recovering or dying patients after ICU admission.

What kind of democracy are we living in? (V-Dem). Varieties of Democracy (V-Dem, (Pemstein et al., 2018; Coppedge et al., 2021))[8] is a dataset containing a collection of indicators of latent regime characteristics over 202 countries from 1789 to 2020. The database includes $h_1 = 483$ low-level indicators (e.g. media bias, party ban, high-court independence, etc.), $h_2 = 82$ mid-level indices (e.g. freedom of expression, freedom of association, equality before the law, etc), and 5 high-level indices of democracy principles (i.e. electoral, liberal, participatory, deliberative, and egalitarian). In the experiments, a binary classification problem is considered

[8]V-Dem: `https://www.v-dem.net/en/data/data/v-dem-dataset-v111`.

to identify electoral democracies from non-electoral democracies. We indicate with C_1 and C_2 the spaces associated to the activations of the aforementioned two levels of concepts. Two classifiers f_1 and f_2 are trained to learn the map $C_1 \rightarrow C_2 \rightarrow Y$. Explanations are given for classifier f_2 in terms of concepts $c_2 \in C_2$.

What does parity mean? (MNIST Even/Odd). The Modified National Institute of Standards and Technology database (MNIST, (LeCun, 1998))[9] contains a large collection of images representing handwritten digits. The input space $X \subset \mathbb{R}^{28\times 28}$ is composed of 28x28 pixel images while the concept space C with $h = 10$ is represented by the label indicator for digits from 0 to 9. However, the task we consider here is slightly different from the common digit-classification. Assuming $Y \subset \{0,1\}^2$, we are interested in determining if a digit is either odd or even, and explaining the assignment to one of these classes in terms of the digit labels (concepts in C). Notice how, for this classification problem, we trivially have ground-truth first-order logic formulas: $\forall x, \text{isOdd}(x) \leftrightarrow \text{isOne}(x) \oplus \text{isThree}(x) \oplus \text{isFive}(x) \oplus \text{isSeven}(x) \oplus \text{isNine}(x)$ and $\forall x, \text{isEven}(x) \leftrightarrow \text{isZero}(x) \oplus \text{isTwo}(x) \oplus \text{isfour}(x) \oplus \text{isSix}(x) \oplus \text{isEight}(x)$, being $\oplus$ the exclusive OR. The mapping $X \rightarrow C$ is provided by a ResNet10 classifier g (He et al., 2016) trained from scratch. while the classifier f is used to learn both the final mapping and the explanation as a function $C \rightarrow Y$.

What kind of bird is that? (CUB). The Caltech-UCSD Birds-200-2011 dataset (CUB, (Wah et al., 2011b))[10] is a fine-grained classification dataset. It includes 11,788 images representing $c = 200$ ($Y = \{0,1\}^{200}$) different bird species. 312 binary attributes describe visual characteristics (color, pattern, shape) of particular parts (beak, wings, tail, etc.) for each bird image. Attribute annotations, however, is quite noisy. For this reason, attributes are denoised by considering class-level annotations (Koh et al., 2020)[11]. In the end, a total of 108 attributes (i.e. concepts with binary activations belonging to C) have been retained. The mapping $X \rightarrow C$ from images to attribute concepts is performed again with a ResNet10 model g trained from scratch while the classifier f learns the final function $C \rightarrow Y$.

7.3.2 Experimental details

Batch gradient-descent and the Adam optimizer with decoupled weight decay (Loshchilov and Hutter, 2017) and learning rate set to 10^{-2} are used for the optimization of all neural models' parameters (Entropy-based Network and ψ Network). An early

[9] MNIST: `http://yann.lecun.com/exdb/mnist`.

[10] CUB: `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`.

[11] A certain attribute is set as present only if it is also present in at least 50% of the images of the same class. Furthermore we only considered attributes present in at least 10 classes after this refinement.

Table 7.1: Hyperparameters selected through cross-validation for the Entropy-based network in each learning task.

	λ	τ	Epochs	Hidden neurons
MIMIC-II	10^{-3}	0.7	200	20
V-Dem	10^{-5}	5	200	20, 20
MNIST	10^{-7}	5	200	10
CUB	10^{-4}	0.7	500	10

stopping strategy is also applied: the model with the highest accuracy on the validation set is saved and restored before evaluating the test set.

With regard to the Entropy-based Network, Tab. 7.1 reports the hyperparameters employed to train the network in all experiments. A grid search cross-validation strategy has been employed to select hyperparameter values. The objective was to maximize at the same time both model and explanation accuracy. λ represents the trade-off parameter in Eq. 7.8 while τ is the temperature of Eq. 7.2.

Concerning the ψ network, in all experiments one network per class has been trained. They are composed of two hidden layers of 10 and 5 hidden neurons, respectively. As indicated in the original paper, an l_1 weight regularization has been applied to all layers of the network. As in this work, the contribute in the overall loss of the l_1 regularization is weighted by a hyperparameter $\lambda = 10^{-}4$. The maximum number of non-zero input weight (fan-in) is set to 3 in MIMIC and V-Dem while for MNIST and CUB200 it is set to 4. In Chapter 6, ψ networks were devised to provide explanations of existing models; in this chapter, however, we show how they can directly solve also a classification problem.

Decision Trees have been limited in their maximum height in all experiments to maintain the complexity of the rules at a comparable level w.r.t the other methods. More precisely, the maximum height has been set to 5 in all binary classification tasks (MIMIC-II, V-Dem, MNIST) while we allowed a maximum height of 30 in the CUB experiment due to the high number of classes to predict (200).

BRL algorithms requires to first run the FP-growth algorithm (Han et al., 2000) (an enhanced version of Apriori) to mine a first set of frequent rules. The hyperparameter used by FP-growth are: the minimum support in percentage of training samples for each rule (set to 10%), the minimum and the maximum number of features considered by each rule (respectively set to 1 and 2). Regarding the Bayesian selection of the best rules, the number of Markov chain Monte Carlo used for inference is set to 3, while 50000 iterations maximum are allowed. At last, the expected length and width of the extracted rule list is set respectively to 3 and 1. These are the default values indicated in the BRL repository. Due to the computational complexity

Table 7.2: Classification accuracy (%) of the compared models.

	Entropy net	Tree	BRL	ψ net	Black box
MIMIC-II	$\mathbf{79.05 \pm 1.35}$	77.53 ± 1.45	76.40 ± 1.22	77.19 ± 1.64	77.81 ± 2.45
V-Dem	$\mathbf{94.51 \pm 0.48}$	85.61 ± 0.57	91.23 ± 0.75	89.77 ± 2.07	94.53 ± 1.17
MNIST	$\mathbf{99.81 \pm 0.02}$	99.75 ± 0.01	99.80 ± 0.02	99.79 ± 0.03	99.81 ± 0.08
CUB	$\mathbf{92.95 \pm 0.20}$	81.62 ± 1.17	90.79 ± 0.34	91.92 ± 0.27	93.32 ± 0.35

and the high number of hyperparameters, they have not been cross validated.

All the experiments have been run on the same machine: Intel® Core™ i7-10750H 6-Core Processor at 2.60 GHz equipped with 16 GiB RAM and NVIDIA GeForce RTX 2060 GPU. All datasets employed can be downloaded for free (only MIMIC-II requires an online registration).

7.3.3 Quantitative metrics

Measuring the classification quality is of crucial importance for models that are going to be applied in real-world environments. On the other hand, assessing the quality of the explanations is required for their lawful deployment. In contrast with other kind of explanations, logic-based formulas can be evaluated quantitatively. Given a classification problem, first a set of rules are extracted from the entropy-based network for each target category and then each explanation is tested on an unseen set of test samples. The results for each metric are reported in terms of mean and standard error, computed over a 5-fold cross validation (Krzywinski and Altman, 2013). For each experiment and for each explainer (i.e. the model $f : C \to Y$ mapping concepts to target categories) six quantitative metrics are measured. (i) The MODEL ACCURACY measures how well the explainer identifies the target classes on unseen data (see Table 7.2). (ii) The EXPLANATION ACCURACY measures how well the extracted logic formulas identifies the target classes (Fig. 7.4). This metric is obtained as the average of the F1 scores computed for each class explanation. (iii) The COMPLEXITY OF AN EXPLANATION estimates how hard to understand the logic formula is for a human being (see Fig. 7.4). This metric is computed by standardizing the explanations in DNF and then by counting the number of terms of the standardized formula (Fig. 7.4). The longer the formula, the harder the interpretation for a human being. (iv) The FIDELITY OF AN EXPLANATION measures how well the extracted explanation matches the predictions obtained using the explainer (Table 7.3). reports out-of-distribution fidelity, i.e. computed on unseen test data. (v) The RULE EXTRACTION TIME measures the time required to obtain an explanation from scratch (see Fig. 7.5). It is computed as the sum of the time required to train the model and the time required to

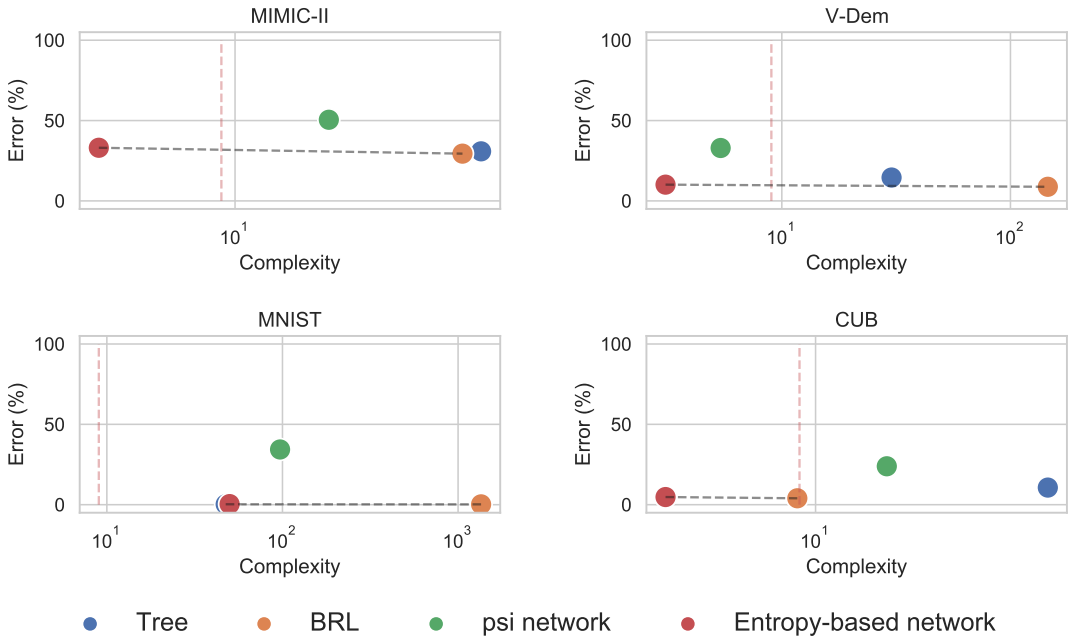


Figure 7.4: Non-dominated solutions (Marler and Arora, 2004) (dotted black line) in terms of average classification test error and their average complexity of the explanations. The vertical dotted red line marks the maximum explanation complexity laypeople can handle (i.e. complexity ≈ 9, see (Miller, 1956; Cowan, 2001; Ma et al., 2014)). When humans compare a set of hypotheses outlining the same outcomes, they tend to have an implicit bias towards the simplest ones, making explanations from entropy-based networks the best choice.

extract the formula from a trained explainer. (vi) The CONSISTENCY OF AN EXPLANATION measures the average similarity of the extracted explanations over the 5-fold cross validation runs (see Table 7.4). It is computed by counting how many times the same concepts appear in the logic formulas over different iterations.

7.3.4 Results analysis

Experiments show how entropy-based network outperforms state-of-the-art white box models such as BRL and decision trees and interpretable neural models such as the ψ networks on challenging classification tasks (Table 7.2). Moreover, the entropy-based regularization has minor effects on the classification accuracy of the explainer as shown in Table 7.2 when compared to a standard black box neural model. This is a neural network having the same architecture and hyperparameters of the entropy-based network, with the only exception of the Lagrangian multiplier in the loss function (see Eq. 7.8) which is set to $\lambda = 0$. This setting makes the network free from any constraint related to explainability. At the same time, the logic explanations provided by entropy-based networks are better than ψ networks

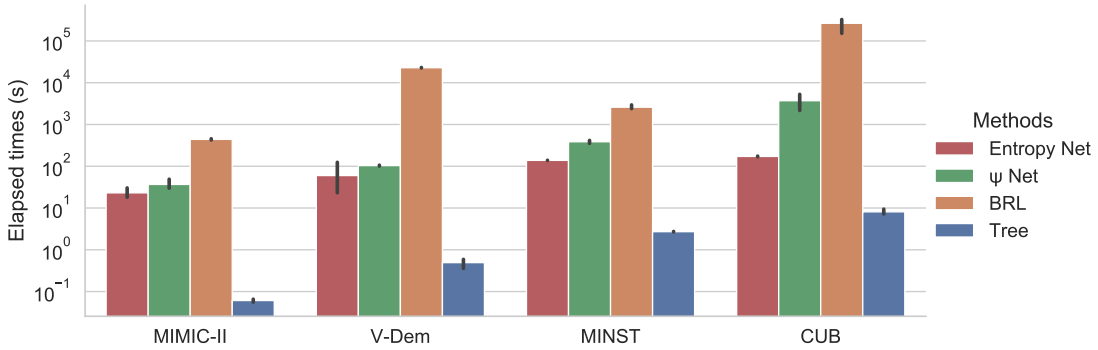


Figure 7.5: Time required to train models and to extract the explanations. Our model compares favourably with the competitors, with the exception of Decision Trees. BRL is by one to three order of magnitude slower than our approach. Error bars show the 95% confidence interval of the mean.

and almost as accurate as the rules found by decision trees and BRL, while being far more concise, as demonstrated in Fig. 7.4. More precisely, logic explanations generated by the proposed approach represent non-dominated solutions (Marler and Arora, 2004) *quantitatively* measured in terms of complexity and classification error (i.e. 100 minus the classification accuracy of the explanation). The complexity of decision tree formulas is never below 100 terms, making them useless as explanations. Furthermore, the time required to train entropy-based networks is only slightly higher with respect to Decision Trees but is lower that ψ Networks and BRL by one to three orders of magnitude (Fig. 7.5), making it feasible for explaining also complex tasks. The fidelity (Table 7.3)[12] of the formulas extracted by the entropy-based network is always higher than 90% with the only exception of MIMIC. This means that almost any prediction made using the logic explanation matches the corresponding prediction made by the model, making the proposed approach very close to a white box model. The combination of these results empirically shows that our method represents a viable solution for the lawful deployment of *explainable* cutting-edge models.

The reason why the proposed approach consistently outperform ψ networks across all the key metrics (i.e. classification accuracy, explanation accuracy, and fidelity) can be explained observing how entropy-based networks are far less constrained than ψ networks, both in the architecture (our approach does not apply weight pruning) and in the loss function (our approach applies a regularization on the distributions α^i and not on all weight matrices). Likewise, the main reason why the proposed approach provides a higher classification accuracy with respect to BRL and decision trees may lie in the smoothness of the decision functions of neural networks which tend to generalize better than rule-based methods, as already observed

[12]We did not compute the fidelity of decision trees and BRL as they are trivially rule-based models.

Table 7.3: Out-of-distribution fidelity (%)

	Entropy net	ψ net
MIMIC-II	$\mathbf{79.11 \pm 2.02}$	51.63 ± 6.67
V-Dem	$\mathbf{90.90 \pm 1.23}$	69.67 ± 10.43
MNIST	$\mathbf{99.63 \pm 0.00}$	65.68 ± 5.05
CUB	$\mathbf{99.86 \pm 0.01}$	77.34 ± 0.52

Table 7.4: Consistency (%)

	Entropy net	Tree	BRL	ψ net
MIMIC-II	28.75	**40.49**	30.48	27.62
V-Dem	46.25	72.00	**73.33**	38.00
MNIST	**100.00**	41.67	**100.00**	96.00
CUB	35.52	21.47	**42.86**	41.43

by Tavares et al. (Tavares et al., 2020). For each dataset, we report in the supplemental material (Appendix A.4.2) a few examples of logic explanations extracted by each method, as well as in Fig. 7.3. We mention that the proposed approach is the only matching the logically correct ground-truth explanation for the MNIST even/odd experiment, i.e. $\forall x, \text{isOdd}(\text{x}) \leftrightarrow \text{isOne}(\text{x}) \oplus \text{isThree}(\text{x}) \oplus \text{isFive}(\text{x}) \oplus \text{isSeven}(\text{x}) \oplus \text{isNine}(\text{x})$ and $\forall x, \text{isEven}(\text{x}) \leftrightarrow \text{isZero}(\text{x}) \oplus \text{isTwo}(\text{x}) \oplus \text{isfour}(\text{x}) \oplus \text{isSix}(\text{x}) \oplus \text{isEight}(\text{x})$, being $\oplus$ the exclusive OR. In terms of formula consistency, we observe how BRL is the most consistent rule extractor, closely followed by the proposed approach (Table 7.4).

The combination of these results empirically shows that our method represents a viable solution for the lawful deployment of *explainable* cutting-edge models.

7.4 Related work

In the last few years, the demand for human-comprehensible models has significantly increased in safety-critical and data-sensible contexts. In order to provide explanations for a given black-box model, most methods focus on identifying or scoring the most relevant input features (Erhan et al., 2010; Simonyan et al., 2013; Zeiler and Fergus, 2014; Ribeiro et al., 2016; Lundberg and Lee, 2017; Selvaraju et al., 2017). Feature scores are usually computed sample by sample (i.e. providing *local explanations*) analyzing the activation patterns in the hidden layers of neural networks (Erhan et al., 2010; Simonyan et al., 2013; Zeiler and Fergus, 2014; Selvaraju et al., 2017) or by following a model-agnostic approach (Ribeiro et al., 2016; Lundberg and Lee, 2017). Either way, feature-scoring methods are not able to explain *how* neural networks compose features to make predictions (Kindermans et al., 2019; Kim et al., 2018b; Alvarez-Melis and Jaakkola, 2018) and only a few of these approaches have been efficiently extended to provide explanations for a whole class (i.e. providing *global explanations*) (Simonyan et al., 2013; Ribeiro et al., 2016). From a different prospective, some recent approaches attempted to identify common activations patterns in the last nodes of a neural network which could be associated to symbolic concepts (Kim et al., 2018a; Kazhdan et al., 2020), or to directly force the

network to extract such concepts (Chen et al., 2020; Koh et al., 2020). Also in this case, even though these methods may help in understanding the model behaviour, they can not be employed to support human decisions.

Differently, rule-based explanations can be employed in such a scenario since they usually rely on a formal language, such as FOL. Logic rules are used to explain how black boxes predict class memberships for indivudal samples (Guidotti et al., 2018a; Ribeiro et al., 2018), or for a whole class (Sato and Tsukimoto, 2001; Zilke et al., 2016; Ciravegna et al., 2020a,b).

Distilling explanations from an existing model, however, is not the only way to achieve explainability. Historically, standard machine-learning such as Logistic Regression (McKelvey and Zavoina, 1975), Generalized Additive Models (Hastie and Tibshirani, 1987; Lou et al., 2012; Caruana et al., 2015) Decision Trees (Breiman et al., 1984; Quinlan, 1986, 2014) and Decision Lists (Rivest, 1987; Letham et al., 2015; Angelino et al., 2018) were devised to be intrinsically interpretable. However, most of them struggle in solving complex classification problems. Logistic Regression, for instance, in its vanilla definition, can only recognize linear patterns, e.g., it cannot to solve the XOR problem (Minsky and Papert, 2017). Further, only Decision Trees and Decision Lists provide explanations in the form of logic rules. Considering decision trees, each path may be seen as a human comprehensible decision rule when the height of the tree is reasonably contained. Another family of concept-based XAI methods is represented by rule-mining algorithms, which became popular at the end of the last century (Holte, 1993; Cohen, 1995). Recent research has led to powerful rule-mining approaches as Bayesian Rule Lists (BRL) (Letham et al., 2015), where a set of rules is "pre-mined" using the frequent-pattern tree mining algorithm (Han et al., 2000) and then the best rule set is identified with Bayesian statistics. As we have seen in this chapter, however, the learning capability of a neural network is not matched by any of the compared white-box model. At the same time, the proposed approach has been able to extract rule — i.e., logic explanations — of comparable quality in terms of both accuracy and conciseness.

Chapter 8

Conclusions

This Chapter concludes this work, resuming and analysing the overall ideas presented in this thesis. Going into more details, in Sec. 8.1 we will analyse in general the strengths and the limits of the proposed methods; in Sec. 8.2, instead, we will resume the contribution of each chapter, and we will outline possible future work in each field of study.

8.1 Learning with logic constraints: strengths and limits

In this thesis, we proposed two different learning with constraints approaches to tackle a few important problems of DNNs. First, we considered the learning *from constraints* framework in which we aim at learning task functions subject to a set of constraints coming from different types of knowledge. This setting allows enhancing DNNs with previously available knowledge and to tackle the data-hungry and the fragility against adversarial attack issues. Second, we considered the learning *of constraints* problem in which we aim at learning function expressing the existing (but unknown) relations among the task functions or between symbolic input data and the task functions. This allows explaining the prediction of a network, either in terms of the other predictions or in terms of the same input data. In both learning problems, we rely on First-Order-Logic to represent a given knowledge or the extracted constraints.

The proposed methods, however, have some limitations, and they require, in general, a higher amount of work from human experts. The learning from constraints approach exploits the domain knowledge, which, however, may not be always available. When this is the case, we can still extract the knowledge from a neural network following the learning of constraint approach, but the quality of the knowledge extracted may be lower than the one provided by a human expert -— e.g., it may suffer from biases present in the dataset. Furthermore, both approaches

rely on the presence of concepts. As defined in Chapter 7, we can regard as concepts the same input data when facing structured data problems. When facing unstructured data problems (e.g., image recognition), however, we can employ as concepts only low-level annotations about the final classes (e.g., person, animal or object attributes). In the latter case, a human expert needs to manually label also the secondary classes, increasing the overall annotation effort. Recent works on automatic concept extraction may alleviate the related costs, leading to more cost-effective concept annotations (Ghorbani et al., 2019; Kazhdan et al., 2020).

8.2 Future Work

Active Learning by Logic Constraints In Chapter 4, we proposed a Knowledge-driven Active Learning strategy (KAL) driven by knowledge consistency principles. The KAL strategy inspects model predictions on unseen data to detect those violating the logic constraints, extracted from an available domain knowledge by means of the T-Norm. The performance of a model equipped with such a strategy outperforms the standard uncertainty-based approach.

As future work, KAL paves the way for novel online learning methods in which the model keeps learning only on those points where the knowledge is violated. Also, the proposed approach could be refined by asking for supervision only for the predicates involved in the violated rules, reducing further the number of required labelled data. At last, in case no knowledge is available on a certain problem, a first idea could be to pair the KAL strategy with the method presented in Chapter 7, where FOL explanations of network predictions are extracted from training data, to continuously check whether the knowledge learnt on the training distribution is also valid on unseen data.

Detecting Adversarial Attacks with Logic Constraints The adversarial example rejection scheme proposed in Chapter 5 is based on the idea of forcing classifiers to fulfil the knowledge-related constraints only over the training space regions. Indeed, the constraints may not be satisfied on the other regions, which may allow detecting malicious data. This has been experimentally evaluated to be a key ingredient to profitably build a rejection strategy. Also, we showed that advanced optimization strategies can fool the defence by injecting, however, a stronger perturbation.

In future work, we will consider intermixing adversarial training with knowledge constraints, to strengthen the violation of the constraints out of the distribution of the real data. We also plan to design a learnable model that decides whether to reject or not the examples in function of the fulfilment of each specific logic formula, going beyond a simple-but-effective threshold on the cumulative constraint loss.

Devising Explanations with an Auxiliary Network In Chapter 6, we presented an approach that yields First-Order Logic-based explanations of a multi-label neural classifier, using another neural network that learns to explain the classifier itself. In multiple use-cases that include user preferences, the model extracts qualitatively sound rules that can also partially improve the classifier.

The main drawback of the proposed approach, however, is the important reduction of the learning capacity of the ψ network due to the pruning procedure in order to make them interpretable. For this reason, in Chapter 6 they have always been used to explain another network, and never as a stand-alone classifier. The method proposed in Chapter 7, is the natural extension of this work since it combines the explainability of ψ network with the learning capacity of standard FFNNs. In further future work, we propose to analyse the effect of learning new constraints with respect to the robustness of the task functions, as in the case of adversarial attacks.

An Explainable-by-Design Network Finally, in chapter 7, we proposed a novel end-to-end differentiable approach enabling the extraction of logic explanations from (almost) standard neural networks. The method, indeed, relies on an entropy-based first neural layer that automatically identifies the most relevant concepts. This entropy layer enables the distillation of concise logic explanations from the neural network, and it contributes to a lawful and safer adoption of deep neural networks.

However, the employment of a Concept-bottleneck model slightly decreases the performance w.r.t. an end-to-end neural network. This is mainly due to the boolean or fuzzy encoding of the concepts, which limit the representation of the final classes. In future work, we will study whether employing a distributed encoding of the concepts may reduce this issue. Also, even though logic explanations are richer than feature importance lists, they are not easy to interpret from a non-expert user. Converting FOL into natural language (following, e.g., (Mpagouli and Hatzilygeroudis, 2007)) could be an interesting way to solve this issue. Finally, a user study comparing different types of explanations could attest which explanations are better from a human point of view.

Appendix A

Experiment Appendix

A.1 Knowledge-driven Active Learning Appendix

In the following, we report some additional results related to the active learning approach presented in Chapter 4.

A.2 Experimental details and further results

A.2.1 The Dog-vs-Person Dataset

The *DOGvsPERSON* dataset is a publicly available dataset that we have created for showing the potentiality of the proposed method on a well-defined object-recognition problem. It has been extracted from the *PASCAL-Part* dataset by considering only the Dog and Person main classes and their corresponding parts. Very specific parts have been merged into a single class following the approach of Serafini and Garcez (2016) (e.g., **Left_Lower_Arm**, **Left_Upper_Arm**, **Right_Lower_Arm**, **Right_Upper_Arm** $\rightarrow$ **Arm**). Differently from the standard *PASCAL-Part* dataset, parts in common to different objects were considered as different classes (**head** $\rightarrow$ **Dog_Head**, **Person_Head**). Furthermore, we only considered as valid label masks having areas $\geq$ 1% of the whole image areas. At last, only classes being displayed at least 100 times have been retained. This led to a total of 20 classes with 2 main classes (**Dog** and **Person**) and 18 parts (**Dog_Ear**, **Dog_Head**, **Dog_Leg**, **Dog_Muzzle**, **Dog_Neck**, **Dog_Nose**, **Dog_Paw**, **Dog_Tail**, **Dog_Torso**, **Person_Arm**, **Person_Foot**, **Person_Hair**, **Person_Hand**, **Person_Head**, **Person_Leg**, **Person_Neck**, **Person_Nose**, **Person_Torso**) displayed in a total of 4304 samples. Final classes are distributed in the samples as shown in Figure A.1.

A.2.2 Experimental Details

XOR-like The problem of inferring the *XOR-like* operation has been already introduced in Section 4.2.1: it is an artificial dataset consisting of 10000 samples $x \in X \subset R^2$, mapped to the corresponding label $y \in Y \subset [0,1]$ as in Eq. 4.2. A Multi-Layer Perceptron (MLP) $f: X \rightarrow Y$ is used to solve the task. It is equipped with a single hidden layer of 100 neurons and Rectified Linear Unit (ReLU) activation, and a single output neuron with sigmoid activation. It has been trained with an AdamW optimizer Loshchilov and Hutter (2017) for 100 epochs at each iteration, with a learning rate $\eta = 10^{-2}$. Standard cross-entropy loss has been used to enforce f to learn the available supervisions. By starting from $n = 10$ samples, we added $p = 5$ labelled samples at each iteration for a total of $b = 100$ iterations, resulting in 1010 supervisions at the last iteration. The accuracies reported in Figure 4.3a are averaged over 10 different seed initializations. As anticipated, in the *XOR* problem, the rule employed for the KAL strategy is $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \Leftrightarrow f$, with the addition of an uncertainty-like constraint $f \oplus \neg f$. In the original *PASCAL-Part* dataset, labels are given in the form of segmentation masks. We extracted a bounding box from each mask by considering the leftmost and highest pixel as the first coordinate and the rightmost and lowest pixel as the second one.

For more real-life style problems, we have considered five more datasets where the domain-knowledge is partial or only related to the class functions.

IRIS The *IRIS*[1] dataset is the standard iris-species classification problem. More precisely, the task consists in classifying $c = 3$ Iris species (Iris Setosa, Iris Versicolour, Iris Virginica) starting from $d = 4$ features (sepal length, sepal width, petal length, petal width). To solve the learning problem, an MLP $f: X^d \rightarrow Y^c$ is employed, with one hidden layer composed of 100 neurons equipped with ReLu activation functions. It has been trained again with AdamW optimizer for 1000 epochs at each iteration and learning rate $\eta = 3 * 10^{-3}$. A Binary Cross-Entropy (BCE) loss is employed to enforce the supervisions. By starting from $n = 5$ points and by adding $p = 5$ labelled samples at each iteration for $b = 16$ iterations. Also in this case, the accuracies reported in Figure 4.3a are averaged over 10 different seed initializations. The knowledge employed in this case consists of 3 very simple rules (one per class) based on the two predicates **Long_Petal** and **Wide_Petal**, with the addition, in this case, of a mutually exclusive rule on the classes **Setosa** $\oplus$ **Versicolour** $\oplus$ **Virginica** and of the uncertainty like constraint for each class.

CUB200 The Caltech-UCSD Birds-200-2011[2] dataset Wah et al. (2011a) is a collection of 11,788 images of birds. The task consists in the classification of 200 birds

[1] *Iris*: `https://archive.ics.uci.edu/ml/datasets/iris`

[2] *CUB200*:`http://www.vision.caltech.edu/visipedia/CUB-200-2011`

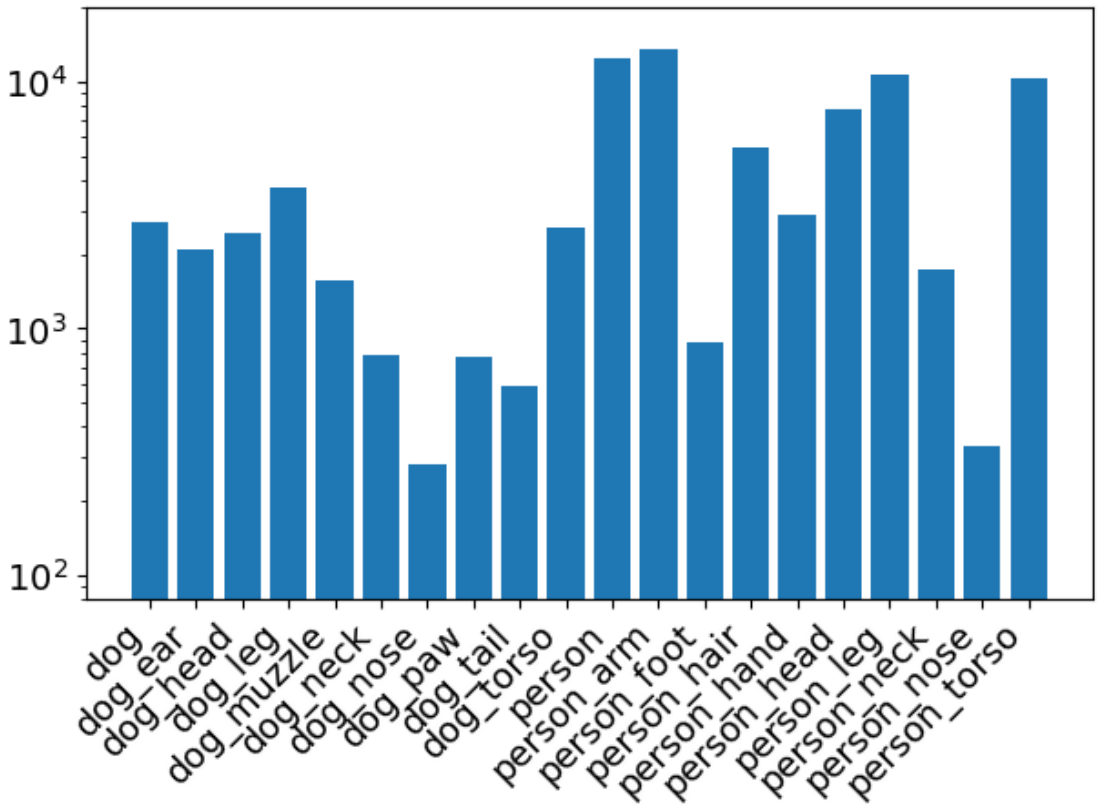


Figure A.1: Class distribution in the *DOGvsPERSON* dataset.

species (e.g., **Black_foooted_Albatross**) and birds attributes (e.g., **White_Throat**, **Medium_Size**). Attribute annotation, however, is quite noisy. For this reason, attributes are denoised by considering class-level annotations similarly to Koh et al. (2020). A certain attribute is set as present only if it is also present in at least 50 images of the same class. Furthermore, we only considered attributes present in at least 10 classes after this refinement. In the end, 108 attributes have been retained, for a total of $c = 308$ classes. Images have been resized to a dimension $d = 256 \times 256$ pixels. A Resnet50 CNN has been employed to solve the task $f\colon X^d \rightarrow Y^c$. Going into more details, a transfer learning strategy has been employed: the network f was pretrained on the ImageNet dataset and only the last fully connected layer has been trained (from scratch) on the *CUB200* dataset. Again, an AdamW optimizer is considered with a learning rate $\eta = 10^{-3}$ for 2000 epochs of training. Owing to the increased difficulty of the problem, we started with $n = 2000$ labelled samples, and we added $p = 100$ samples for $b = 50$ iterations, for a total of 7000 samples. For diversity sampling, at each iteration, we pick a maximum of $r = 20$ samples violating the same rule. Since *CUB200* is a multi-label classification problem, in Figure 4.3c we reported the F1 score of the model when vary-

ing the number of supervised training samples The knowledge employed in this case consider the relation between the classes and their attributes both from the class to the attributes (e.g., **White_Pelican** $\Rightarrow$ **Black_Eye** $\vee$ **Solid_Belly_Pattern** $\vee$**Solid_Wing_Pattern**), and the vice-versa (e.g., **Striped_Breast_Pattern** $\Rightarrow$ **Parakeet_Auklet** $\vee$**Black_throated_Sparrow** $\vee \dots$) Furthermore, a disjunction of the main classes and one on the attributes are considered Again also in this case an uncertainty-like constraint is considered (e.g., **Black_throated_Sparrow**$\oplus$ $\neg$**Black_throated_Sparrow**).

ANIMALS The *Animals'* dataset is a collection of 8287 images of animals, taken from the ImageNet database[3]. The task consists in the classification of 7 main classes (**Albatross**, **Giraffe**, **Cheetah**, **Ostrich**, **Penguin**, **Tiger**, **Zebra**) and 26 animal attributes (e.g., **Mammal**, **Fly** or **Lay_Eggs**), for a total of $c = 33$ classes. The same network as in the previous case has been employed to solve the learning problem. Training was carried on for 1000 epochs at each iteration with standard cross-entropy loss and the same optimizer as in the previous case. We started with $n = 100$ labelled samples, and we added $p = 50$ samples each time for $b = 98$ iterations, for a total of 5000 labelled samples. For diversity sampling, at each iteration, we pick a maximum of $r = 20$ samples violating the same rule. Also in this case, we employed the F1 score as a metric in Figure 4.3d. In the case of *Animals*, the employed knowledge is a simple collection of 16 FOL formulas, defined by Winston and Horn (1986) as a benchmark. They involve relationships between animals and their attributes, such as $\forall x \, \text{Fly}(x) \wedge \text{Lay\_Eggs}(x) \Rightarrow \text{Bird}(x)$ (mostly *type b*) rules, following the notation introduced in Section 4.2.2). To this collection of rules, we have also added a mutual exclusive disjunction among the animal classes (only one animal is present in each image) and a standard disjunction over the animal attributes (each animal has more than one attribute.

DOGvsPERSON This dataset has been already introduced in Appendix A.2.1. Since we filtered out very small object masks in this dataset, we have been able to employ a YOLOv3 model Redmon and Farhadi (2018) to solve the object-recognition problem. The model has been trained for 100 epochs at each iteration with an AdamW optimizer, with a learning rate $\eta = 3 * 10^{-4}$ decreasing by 1/3 every 33 epochs. For both training and evaluation, the Input Over Union (IOU) threshold has been set to 0.5, the confidence threshold to 0.01 and the Non-Maximum Suppression (NMS) threshold to 0.5. We started training with $n = 1000$ labelled examples and by adding $p = 100$ samples for $b = 10$ iterations for a total of 2000 labelled examples. In Figure 4.3e we reported the growth of the mean Average Precision (mAP) of the model averaged 10 times with Intersection over Union (IoU) ranging from 0.5 to

[3] *Animals*: `http://www.image-net.org/`

0.95. On *DOGvsPERSON* , we have a set of rules listing the parts belonging to the dog or the person, (e.g., **Person** $\Rightarrow$ **Person_Arm** $\vee$ **Person_Foot** $\vee$ **Person_Hair** $\vee$ **Person_Hand** $\vee \ldots$), the opposite rules implying the presence of the main object given the part (e.g., **Person_Foot** $\Rightarrow$ **Person**). Also, we have a disjunction of all the main classes and a disjunction of all the object-parts, for a total of 22 rules employed.

PASCAL-Part The *PASCAL-Part* dataset[4] is composed of 10103 images of varying size, depicting objects (**Person**, **Aeroplane**, etc.) and object-parts (**Head**, **Muzzle**, **Tail**, etc.). We preprocessed the dataset following the approach of Serafini and Garcez (2016), merging specific parts into unique labels. Furthermore, we have divided original part classes describing very different objects into different classes (e.g., **Body** $\rightarrow$ **Bottle_Body**, **aeroplane_body**), leading to $c = 66$ classes, out of which 16 are main objects. In this dataset, labels are given in the form of segmentation masks. A Faster R-CNN network Ren et al. (2016) is trained on the bounding boxes extracted from each mask, the leftmost and highest pixels are used for the first coordinate and the opposites for the second one. Owing to the computational complexity of the task, the model has been trained for 50 epochs at each iteration, with an SGD optimizer with momentum and a learning rate schedule with an initial value $\eta = 3 \cdot 10^{-3}$ decreasing by 0.3 every 20 epochs. In this case, we started with $n = 500$ labelled examples and by adding $p = 50$ samples for $b = 10$ iterations for a total of 1000 supervisions. At each iteration, we pick a maximum of $r = 5$ samples violating the same rule. In Figure 4.3f the reported mAP is calculated as in the previous case. On *PASCAL-Part*, we have a set of rules listing the parts belonging to a certain object, (e.g., **Motorbike** $\Rightarrow$ **Wheel** $\vee$ **Headlight** $\vee$ **Handlebar** $\vee$ **Saddle**), and listing all the objects in which a part can be found (e.g., **Handlebar** $\Rightarrow$ **Bicycle** $\vee$ **Motorbike**). Also, we have a disjunction of all the main classes and a disjunction of all the object-parts, for a total of 62 rules employed.

For all experiments, the results reported in Figure 4.3a corresponds to the model classification performances over the whole pool of data, i.e., over both the labelled and the unlabelled samples.

Diversity-based strategies In the UNCERTAIN+ strategy, the K-Means clustering has been set to always search for 8 clusters. Instead, the maximum number of points selected from the same cluster has been set through a hyperparameters search to 2 in the *XOR-like* and *IRIS* problems, while it has been set to 20 in the computer vision problems. In the KAL strategy, instead, a diverse selection is always obtained by heuristically requiring that a maximum of $r = p/2$ of the selected samples can mostly violate the same rule.

[4] *PASCAL-Part*: `http://roozbehm.info/pascal-parts/pascal-parts`.

A.2.3 Training evolutions on the *XOR-like* problem

In Figure A.2 we report further snapshots of the training process. They depict for each of the compared method the model predictions, similarly to what it has been shown in Figure 4.4, but for different iterations. As it has already been noticed, UNCERTAIN is the only method unable to discover the data distribution in the right-bottom angle (for which no samples have been drawn during the initial random sampling) even after 100 iterations. When coupled with diversity-based strategy, UNCERTAIN+ is instead capable to eventually cover all data distribution; however, after 25 iterations the right-bottom distribution was still wrongly predicted. Also, it is interesting to notice how the sampling selection performed by the proposed approach resemble that made by the SUPERVISED one. For the complete animations showing the evolution of the training at each iteration for each method, please check the public GitHub repository.

A.3 Software

The Python code and the scripts used for the experiments, including parameter values and documentation, is freely available under Apache 2.0 Public Licence from a GitHub repository[5]. The proposed approach requires only a few lines of code to train a model following the KAL strategy, as we sketch in the following code example (Listing A.1).

[5] `https://github.com/gabrieleciravegna/Constrained-Active-Learning`

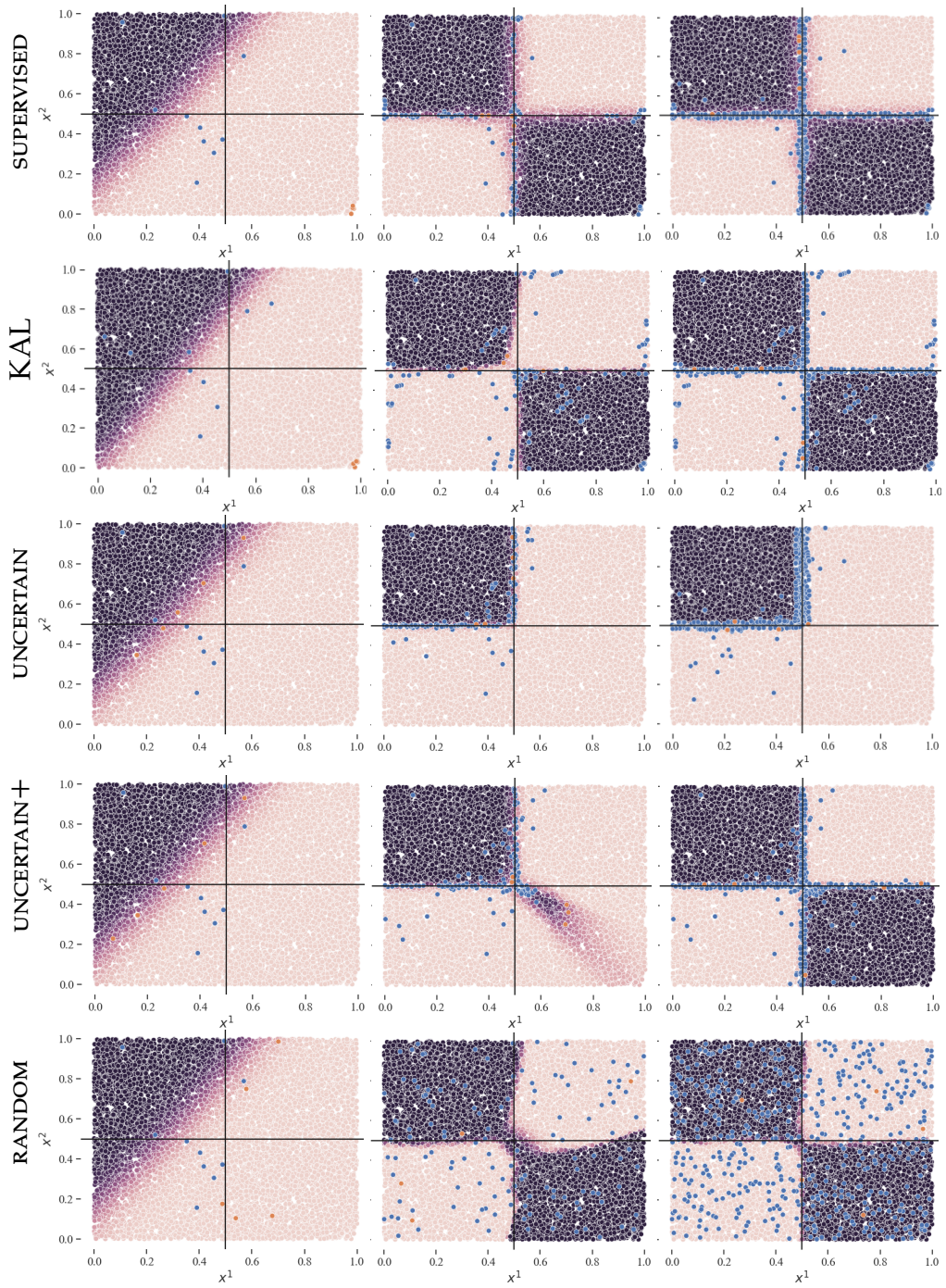


Figure A.2: A visual example on the *XOR-like* problem, showing how the training evolves in each of the compared strategy. We depict network predictions with different colour degrees (light colours negative predictions, dark colours positive prediction). In blue, we depict the points selected in previous iterations, in orange those selected at the current iteration. Black lines at $x^1 = 0.5$ and $x^2 = 0.5$ are reported only for visualization purposes. From left to right, the situation at the 1^{st}, 25^{th} and 100^{th} iteration.

```
tot_points = 10000
first_points = 10
n_points = 5
n_iterations = 98

# Generating data for the xor problem
x = np.random.uniform(size=(tot_points, 2))
y = np.ndarray.astype(((x[:, 0] > 0.5) & (x[:, 1] < 0.5)) |
                      ((x[:, 1] > 0.5) & (x[:, 0] < 0.5)), float)
x_t = torch.as_tensor(x, dtype=torch.float)
y_t = torch.as_tensor(y, dtype=torch.float)

# Defining constraints as product t-norm of the FOL rule expressing the XOR (x1 & ~x2) | (x2 & ~x1) <=> f
def calculate_constraint_loss(x_continue: torch.Tensor, f: torch.Tensor):
    discrete_x = steep_sigmoid(x_continue).float()
    x1 = discrete_x[:, 0]
    x2 = discrete_x[:, 1]
    cons_loss1 = f * ((1 - (x1 * (1 - x2))) * (1 - (x2 * (1 - x1))))
    cons_loss2 = (1 - f) * (1 - ((1 - (x1 * (1 - x2))) * (1 - (x2 * (1 - x1)))))
    return cons_loss1 + cons_loss2

# Constrained Active learning strategy
# We take the p samples that most violate the constraints and are among available idx
def cal_selection(not_avail_idx: list, c_loss: torch.Tensor, p: int):
    c_loss[torch.as_tensor(not_avail_idx)] = -1
    cal_idx = torch.argsort(c_loss, descending=True).tolist()[:p]
    return cal_idx

# Few epochs with n randomly selected data
net = MLP(2, 100)
first_idx = np.random.randint(0, x.shape[0] - 1, first_points).tolist()
train_loop(net, x_t, y_t, first_idx)

preds_t = net(x_t)

cons_loss = calculate_constraint_loss(x_t, preds_t)
available_idx = [*range(tot_points)]
used_idx = first_idx

# Active Learning with KAL strategy for n_iterations
for n in range(1, n_iterations + 1):
    available_idx = list(set(available_idx) - set(used_idx))
    active_idx = cal_selection(used_idx, cons_loss, n_points)
    used_idx += active_idx

    # train for 50 epochs the MLP on the used idx
    train_loop(net, x_t, y_t, used_idx, epochs=50)

    preds_t = net(x_t).squeeze()
    accuracy = (preds_t > 0.5).eq(y_t).sum().item() / y_t.shape[0] * 100
    cons_loss = calculate_constraint_loss(x_t, preds_t)
```

Listing A.1: KAL code - Example on the XOR problem.

A.4 Entropy-based Logic Explanation of Neural Networks Appendix

In the following, we report some additional results related to the XAI method presented in Chapter 7.

A.4.1 Software

In order to make the proposed approach accessible to the whole community, we released "PyTorch, Explain!" Barbiero et al. (2021a), a Python package [6] with an extensive documentation on methods and unit tests. The Python code and the scripts used for the experiments, including parameter values and documentation, is freely available under Apache 2.0 Public License from a GitHub repository [7]. The code library is designed with intuitive APIs requiring only a few lines of code to train and get explanations from the neural network as shown in the following code snippet A.2 showing how to train an Entropy-based network on the XOR problem.

A.4.2 Extracted Rules

In Table A.1, we report a selection of the rule extracted by each method in all experiments is shown. For all methods, we report only the explanations of the first class for the first split of the Cross-validation. For the Entropy-based method only, Tables A.2, A.3, A.4, A.5 resume the explanations of all classes in all experiments.

[6] `https://pypi.org/project/torch_explain/`
[7] `https://github.com/pietrobarbiero/pytorch_explain`

```
import torch_explain as te
from torch_explain.logic import test_explanation
from torch_explain.logic.nn import explain_class

# XOR problem with additional features
x0 = torch.zeros((4, 100))
x = torch.tensor([
    [0, 0, 0],
    [0, 1, 0],
    [1, 0, 0],
    [1, 1, 0],
], dtype=torch.float)
x = torch.cat([x, x0], dim=1)
y = torch.tensor([0, 1, 1, 0], dtype=torch.long)

# network architecture
layers = [
    te.nn.EntropyLogicLayer(x.shape[1], 10, n_classes=2),
    torch.nn.LeakyReLU(),
    te.nn.LinearIndependent(10, 10, n_classes=2),
    torch.nn.LeakyReLU(),
    te.nn.LinearIndependent(10, 1, n_classes=2, top=True)
]
model = torch.nn.Sequential(*layers)

# train loop
optimizer = torch.optim.AdamW(model.parameters(), lr=0.01)
loss_form = torch.nn.CrossEntropyLoss()
model.train()
for epoch in range(1001):
    optimizer.zero_grad()
    y_pred = model(x)
    loss = loss_form(y_pred, y) + \
           0.00001 * te.nn.functional.entropy_logic_loss(model)
    loss.backward()
    optimizer.step()

# logic explanations
y1h = one_hot(y)
_, class_explanations, _ = explain_class(model, x, y1h, x, y1h)
```

Listing A.2: Example on how to use the APIs to implement the proposed approach on the simple XOR problem.

Table A.1: Comparison of the formulas obtained in the first run of each experiment for all methods. Only the formula explaining the first class has been reported. Ellipses are used to truncate overly long formulas.

Dataset	Method	Formulas
MIMIC-II	Entropy	non_recover ↔ ¬liver_flg ∧ ¬stroke_flg ∧ ¬mal_flg
	DTree	non_recover ↔ (age_high < 0.5 ∧ mal_flg < 0.5 ∧ stroke_flg < 0.5 ∧ age_normal < 0.5 ∧ iv_day_1_normal < 0.5) ∨ (age_high < 0.5 ∧ mal_flg < 0.5 ∧ stroke_flg < 0.5 ∧ age_normal < 0.5 ∧ iv_day_1_normal > 0.5) ∨ (age_high < 0.5 ∧ ...
	BRL	non_recover ↔ (age_low ∧ sofa_first_low ∧ ¬(mal_flg ∧ ¬weight_first_normal)) ∨ (age_high ∧ ¬service_num_normal ∧ ¬(age_low ∧ sofa_first_low) ∧ ¬(chf_flg ∧ ¬day_icu_intime_num_high) ∧ ¬(mal_flg ∧ ¬weight_first_normal) ∧ ¬(stroke_flg ∧ ...
	ψ Net	non_recover ↔ (iv_day_1_normal ∧ ¬age_high ∧ ¬hour_icu_intime_normal ∧ ¬sofa_first_normal) ∨ (mal_flg ∧ ¬age_high ∧ ¬hour_icu_intime_normal ∧ ¬sofa_first_normal) ∨ ...
V-Dem	Entropy	non_electoral_democracy ↔ ¬v2xel_frefair ∨ ¬v2x_elecoff ∨ ¬v2x_cspart ∨ ¬v2xeg_eqaccess ∨ ¬v2xeg_eqdr
	DTree	non_electoral_democracy ↔ (v2xel_frefair < 0.5 ∧ v2xdl_delib < 0.5 ∧ v2x_frassoc_thick < 0.5) ∨ (v2xel_frefair < 0.5 ∧ v2xdl_delib < 0.5 ∧ v2x_frassoc_thick > 0.5 ∧ v2x_freexp_altinf < 0.5 ∧ v2xeg_eqprotec < 0.5) ∨ ...
	BRL	non_electoral_democracy ↔ ¬v2x_cspart ∨ ¬v2x_elecoff ∨ ¬v2x_frassoc_thick ∨ ¬v2x_freexp_altinf ∨ ¬v2xcl_rol ∨ (¬v2x_mpi ∧ ¬v2xel_frefair)
	ψ Net	non_electoral_democracy ↔ ¬v2xeg_eqaccess ∨ (v2x_egal ∧ ¬v2x_frassoc_thick) ∨ (v2xeg_eqdr ∧ ¬v2x_egal) ∨ (v2xel_frefair ∧ ¬v2x_frassoc_thick) ∨ (¬v2x_cspart ∧ ¬v2x_suffr) ∨ (¬v2x_frassoc_thick ∧ ¬v2x_suffr) ∨ ...
MNIST	Entropy	even ↔ (zero ∧ ¬one ∧ ¬two ∧ ¬three ∧ ¬four ∧ ¬five ∧ ¬six ∧ ¬seven ∧ ¬eight ∧ ¬nine) ∨ (two ∧ ¬zero ∧ ¬one ∧ ¬three ∧ ¬four ∧ ¬five ∧ ¬six ∧ ¬seven ∧ ¬eight ∧ ¬nine) ∨ (four ∧ ¬zero ∧ ...
	DTree	even ↔ (one < 0.54 ∧ nine < $1.97{\cdot}10^{-}5$ ∧ three < 0.00 ∧ five < 0.09 ∧ seven < 0.20) ∨ (one < 0.54 ∧ nine < $1.97{\cdot}10^{-}5$ ∧ three < 0.00 ∧ five > 0.09 ∧ two > 0.97) ∨ (one < 0.54 ∧ nine < $1.97{\cdot}10^{-}5$ ∧ three > 0.00 ∧ two < 0.99 ∧ eight > 1.00) ∨ ...
	BRL	even ↔ (two ∧ ¬one ∧ ¬seven ∧ ¬three ∧ ¬(seven ∧ ¬two)) ∨ (four ∧ ¬five ∧ ¬nine ∧ ¬seven ∧ ¬three ∧ ¬(seven ∧ ¬two) ∧ ¬(two ∧ ¬one)) ∨ (four ∧ ¬five ∧ ¬seven ∧ ¬three ∧ ¬(four ∧ ¬nine) ∧ ¬(seven ∧ ...
	ψ Net	even ↔ (four ∧ nine ∧ six ∧ three ∧ zero ∧ ¬eight ∧ ¬one ∧ ¬seven) ∨ (four ∧ nine ∧ six ∧ two ∧ zero ∧ ¬eight ∧ ¬one ∧ ¬seven) ∨ (eight ∧ six ∧ ¬four ∧ ¬nine ∧ ¬seven ∧ ¬three ∧ ¬two) ∨ (eight ∧ six ∧...
CUB	Entropy	black_footed_albatross ↔ has_bill_length_about_the_same_as_head ∧ has_wing_pattern_solid ∧ ¬has_upper_tail_color_grey ∧ ¬has_belly_color_white ∧ ¬has_wing_shape_roundedwings ∧ ¬has_bill_color_black
	DTree	black_footed_albatross ↔ (has_back_pattern_striped < 0.46 ∧ has_back_color_buff < 0.69 ∧ has_upper_tail_color_white < 0.59 ∧ has_under_tail_color_buff < 0.82 ∧ has_shape_perchinglike < 0.66 ∧ ...
	BRL	black_footed_albatross ↔ (has_back_pattern_striped ∧ has_belly_color_black ∧ has_bill_shape_hooked_seabird ∧ ¬has_belly_color_white) ∨ (has_back_pattern_striped ∧ ...
	ψ Net	black_footed_albatross ↔ (has_bill_shape_hooked_seabird ∧ ¬has_breast_color_white ∧ ¬has_size_small_5__9_in ∧ ¬has_wing_color_grey) ∨ (has_bill_shape_hooked_seabird ∧ ...

Table A.2: Formulas extracted from the MIMIC-II dataset.

Formulas
non_recover $\leftrightarrow \neg$liver_flg $\wedge \neg$stroke_flg $\wedge \neg$mal_flg
recover $\leftrightarrow$ mal_flg $\vee$ (age_HIGH $\wedge \neg$iv_day_1_NORMAL)

Table A.3: Formulas extracted from the V-Dem dataset.

Formulas
non_electoral_democracy $\leftrightarrow \neg$v2xel_frefair $\vee \neg$v2x_elecoff $\vee \neg$v2x_cspart $\vee \neg$v2xeg_eqaccess $\vee \neg$v2xeg_eqdr
electoral_democracy $\leftrightarrow$ v2xel_frefair $\wedge$ v2x_elecoff $\wedge$ v2x_cspart

Table A.4: Formulas extracted from the MNIST dataset.

Formulas
even $\leftrightarrow$ (zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (two $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (four $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (six $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (eight $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$nine)
odd $\leftrightarrow$ (one $\wedge \neg$zero $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (three $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (five $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (seven $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (nine $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight)

Table A.5: Formulas extracted from the CUB dataset. For the sake of brevity only the rules extracted for the first three classes have been reported.

Formulas
Black_footed_Albatross $\leftrightarrow$ has_bill_length_about_the_same_as_head $\wedge$ has_wing_pattern_solid $\wedge \neg$has_upper_tail_color_grey $\wedge \neg$has_belly_color_white $\wedge \neg$has_wing_shape_roundedwings $\wedge \neg$has_bill_color_black
Laysan_Albatross $\leftrightarrow$ has_crown_color_white $\wedge$ has_wing_pattern_solid $\wedge \neg$has_under_tail_color_-white
Sooty_Albatross $\leftrightarrow$ has_upper_tail_color_grey $\wedge$ has_size_medium_9__16_in $\wedge$ has_bill_color_-black $\wedge$
...

Appendix B

Publications

Journal papers

1. **Gabriele Ciravegna** Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Lio', Marco Maggini, Stefano Melacci, "Logic Explained Networks", *Artificial Intelligence Journal*, 2023.
   Candidate's contributions: developed parts of the algorithm, carried out most of the experimental analysis, wrote most of the paper.

2. Stefano Melacci, **Gabriele Ciravegna**, Angelo Sotgiu, Ambra Demontis, Battista Biggio, Marco Gori, and Fabio Roli. "Domain Knowledge Alleviates Adversarial Attacks in Multi-Label Classifiers", *IEEE Transaction on Pattern Analysis and Machine Intelligence* (*TPAMI*) journal, 2022.
   Candidate's contributions: developed part of the algorithm, carried out most of the experimental analysis.

3. Matteo Tiezzi, **Gabriele Ciravegna** Marco Gori "Graph Neural Networks for Graph Drawing", submitted to the *IEEE Transactions on Neural Networks and Learning Systems* (*TNNLS*) *journal*, 2022.
   Candidate's contributions: developed part of the algorithm, carried out some experimental analysis, wrote part of the paper.

4. Giansalvo Cirrincione, **Gabriele Ciravegna**, Pietro Barbiero, Vincenzo Randazzo, and Eros Pasero. "The GH-EXIN neural network for hierarchical clustering". *Neural Networks*, pages: 57-73, 2020.
   Candidate's contributions: developed part of the algorithm, carried out most of the experimental analysis, wrote part of the paper.

5. Marta Lovino, Vincenzo Randazzo, **Gabriele Ciravegna** Pietro Barbiero, Elisa Ficarra and Giansalvo Cirrincione, "A survey on data integration for multi-omics sample clustering", *Neurocomputing* journal, 2022 (to appear).

Candidate's contributions: carried out some of the experimental analysis, wrote part of the paper.

Peer reviewed conference papers

1. **Gabriele Ciravegna**, Francesco Giannini, Stefano Melacci, Marco Maggini, and Marco Gori. "A constraint-based approach to learning and explanation". *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3658–3665, 2020.
   Candidate's contributions: carried out all the experimental analysis, wrote part of the paper.

2. **Gabriele Ciravegna**, Francesco Giannini, Marco Gori, Marco Maggini, and Stefano Melacci. "Human-driven fol explanations of deep learning". *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2234–2240, 2020.
   Candidate's contributions: developed part of the algorithm, carried out all the experimental analysis, wrote part of the paper.

3. Pietro Barbiero, **Gabriele Ciravegna**, Francesco Giannini, Marco Gori, Pietro Lio, Stefano Melacci, "Entropy-based Logic Explanations of Neural Networks", accepted at the *AAAI Conference on Artificial Intelligence 2022*.
   Candidate's contributions: developed part of the algorithm, carried out some experimental analysis, wrote part of the paper.

4. Pietro Barbiero, **Gabriele Ciravegna**, Vincenzo Randazzo, and Giansalvo Cirrincione. "Topological gradient-based competitive learning", 2021 International Joint Conference on Neural Networks (IJCNN), pages 1-8, 2021.
   Candidate's contributions: developed part of the algorithm, carried out some experimental analysis, wrote part of the paper.

5. Barbiero, Pietro, Lovino, Marta, Siviero, Mattia, Ciravegna, Gabriele, Randazzo, Vincenzo, Ficarra, Elisa, and Cirrincione, Giansalvo "Unsupervised Multi-omic Data Fusion: The Neural Graph Learning Network". *International Conference on Intelligent Computing* (pp. 172-182), Springer, Cham, 2020.
   Candidate's contributions: carried out some experimental analysis, wrote part of the paper.

6. **Gabriele Ciravegna** Frederic Precioso, Marco Gori, "Knowledge-driven Active Learning", submitted to the *European Conference on Machine Learning (ECML)*, 2023.
   Candidate's contributions: developed most of the algorithm, carried out all the experimental analysis, wrote most of the paper.

Workshop Papers

1. **Gabriele Ciravegna**, Pietro Barbiero, Giansalvo Cirrincione, Giovanni Squillero, Alberto Tonda, "Discovering Hierarchical Neural Archetype Sets", *Workshop Italiano Reti Neurali (WIRN)*, published as chapter of *Progresses in Artificial Intelligence and Neural Systems, Springer*, pages 255-267, 2019.
   Candidate's contributions: developed part of the algorithm, carried out part of the experimental analysis, wrote part of the paper.

Bibliography

Agrawal, R. (2019). Introduction to deep learning. `https://medium.com/@rochak.agrawal/introduction-to-deep-learning-5ffd8b625b00`. Accessed: 2021-10-18.

Akcay, S., Atapour-Abarghouei, A., and Breckon, T. P. (2018). Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*, pages 622–637. Springer.

Alayrac, J.-B., Uesato, J., Huang, P.-S., Fawzi, A., Stanforth, R., and Kohli, P. (2019). Are labels required for improving adversarial robustness? In *Neural Information Processing Systems*, pages 12214–12223.

Alvarez-Melis, D. and Jaakkola, T. S. (2018). Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020). Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer Vision – ECCV 2020*, pages 484–501, Cham. Springer International Publishing.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., and Rudin, C. (2018). Learning certifiably optimal rule lists for categorical data.

Araujo, A., Meunier, L., Pinot, R., and Negrevergne, B. (2020). Robust Neural Networks using Randomized Adversarial Training. *arXiv:1903.10219* [*cs, stat*]. arXiv: 1903.10219.

Aristotle (350 B.C.). Posterior analytics.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2019). Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*.

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

Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR.

Babbar, R. and Schölkopf, B. (2018). Adversarial extreme multi-label classification. *arXiv preprint arXiv:1803.01570*.

Barbiero, P., Ciravegna, G., Georgiev, D., and Giannini, F. (2021a). Pytorch, explain! a python library for logic explained networks. *arXiv preprint arXiv:2105.11697*.

Barbiero, P., Ciravegna, G., Giannini, F., Lió, P., Gori, M., and Melacci, S. (2021b). Entropy-based logic explanations of neural networks. *arXiv preprint arXiv:2106.06804*.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Betti, A., Gori, M., and Melacci, S. (2019). Cognitive action laws: The case of visual features. *IEEE transactions on neural networks and learning systems*.

Bibal, A. and Frénay, B. (2016). Interpretability of machine learning models and representations: an introduction. In *ESANN*.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In Blockeel, H., Kersting, K., Nijssen, S., and Železný, F., editors, *Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Part III*, volume 8190 of *LNCS*, pages 387–402. Springer Berlin Heidelberg.

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 59–66.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Burbidge, R., Rowland, J. J., and King, R. D. (2007). Active learning for regression based on query by committee. In *International conference on intelligent data engineering and automated learning*, pages 209–218. Springer.

Cao, X. and Tsang, I. W. (2021). Bayesian active learning by disagreements: A geometric perspective.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

Carlini, N. and Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 3–14.

Carlini, N. and Wagner, D. A. (2017b). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57. IEEE Computer Society.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. (2019). Unlabeled data improves adversarial robustness. In *Neural Information Processing Systems*, pages 11190–11201.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (2015). 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*, pages 1721–1730.

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

Castro, J. L. and Trillas, E. (1998). The logic of neural networks. *Mathware and Soft Computing*, 5:23–37.

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts.

Chen, Z., Bei, Y., and Rudin, C. (2020). Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782.

Choi, J., Elezi, I., Lee, H.-J., Farabet, C., and Alvarez, J. M. (2021). Active learning for deep object detection via probabilistic modeling.

Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., and Melacci, S. (2021). Logic explained networks. *arXiv preprint arXiv:2108.05149*.

Ciravegna, G., Giannini, F., Gori, M., Maggini, M., and Melacci, S. (2020a). Human-driven fol explanations of deep learning. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}*, pages 2234–2240. International Joint Conferences on Artificial Intelligence Organization.

Ciravegna, G., Giannini, F., Melacci, S., Maggini, M., and Gori, M. (2020b). A constraint-based approach to learning and explanation. In *AAAI*, pages 3658–3665.

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier.

Coppedge, M., Gerring, J., Knutsen, C. H., Lindberg, S. I., Teorell, J., Altman, D., Bernhard, M., Cornell, A., Fish, M. S., Gastaldi, L., et al. (2021). V-dem codebook v11.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences*, 24(1):87–114.

Croce, F. and Hein, M. (2020a). Minimally distorted adversarial examples with a fast adaptive boundary attack. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2196–2205, Virtual. PMLR.

Croce, F. and Hein, M. (2020b). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 1–12.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

d'Avila Garcez, A. S., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., and Tran, S. N. (2019). Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics - IfCoLog Journal*, 6(4):611–632.

De Raedt, L. and Kimmig, A. (2015). Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47.

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., and Roli, F. (2019). Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association.

Di Nola, A., Gerla, B., and Leustean, I. (2013). Adding real coefficients to łukasiewicz logic: An application to neural networks. In *International Workshop on Fuzzy Logic and Applications*, pages 77–85. Springer.

Diligenti, M., Gori, M., and Sacca, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165.

Donadello, I., Serafini, L., and Garcez, A. D. (2017). Logic tensor networks for semantic image interpretation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1596–1602. AAAI Press.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Došilović, F. K., Brčić, M., and Hlupić, N. (2018). Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE.

Ducoffe, M. and Precioso, F. (2017). Active learning strategy for cnn combining batchwise dropout and query-by-committee. In *ESANN*.

Ducoffe, M. and Precioso, F. (2018). Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*.

Erhan, D., Courville, A., and Bengio, Y. (2010). Understanding representations learned in deep architectures. *Department dInformatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep*, 1355(1).

EUGDPR (2017). Gdpr. general data protection regulation.

Freitas, A. A. (2014). Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10.

Fu, L. (1991). Rule learning by searching on adapted nets. In *AAAI*, volume 91, pages 590–595.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data.

Ghorbani, A., Wexler, J., Zou, J., and Kim, B. (2019). Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*.

Gnecco, G., Gori, M., Melacci, S., and Sanguineti, M. (2015). Foundations of support constraint machines. *Neural computation*, 27(2):388–480.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220.

Gong, T., Liu, B., Chu, Q., and Yu, N. (2019). Using multi-label classification to improve object detection. *Neurocomputing*, 370:174–185.

Goodfellow, I., McDaniel, P., and Papernot, N. (2018). Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *preprint arXiv:1412.6572*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Gori, M. and Melacci, S. (2013). Constraint verification with kernel machines. *IEEE Trans. Neural Networks Learn. Syst.*, 24(5):825–831.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J. P., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., and Giannotti, F. (2018a). Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018b). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2(2).

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12.

Harmon, S. A., Sanford, T. H., Xu, S., Turkbey, E. B., Roth, H., Xu, Z., Yang, D., Myronenko, A., Anderson, V., Amalou, A., et al. (2020). Artificial intelligence

for the detection of covid-19 pneumonia on chest ct using multinational datasets. *Nature communications*, 11(1):1–7.

Hastie, T. and Tibshirani, R. (1987). Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386.

Haussmann, E., Fenzi, M., Chitta, K., Ivanecky, J., Xu, H., Roy, D., Mittel, A., Koumchatzky, N., Farabet, C., and Alvarez, J. M. (2020). Scalable active learning for object detection.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hein, M., Andriushchenko, M., and Bitterwolf, J. (2019). Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50.

Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*.

Hendrycks, D. and Gimpel, K. (2017). Early methods for detecting adversarial images. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Hoffmann, R., Minkin, V. I., and Carpenter, B. K. (1996). Ockham's razor and chemistry. *Bulletin de la Société chimique de France*, 2(133):117–130.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning.

Huang, S. H. and Xing, H. (2002). Extract intelligible and concise fuzzy rules from neural networks. *Fuzzy Sets and Systems*, 132(2):233–243.

Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., and Baesens, B. (2011). An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154.

Ichnowski, J., Avigal, Y., Satish, V., and Goldberg, K. (2020). Deep learning can accelerate grasp-optimized motion planning. *Science Robotics*, 5(48).

Joshi, A., Mukherjee, A., Sarkar, S., and Hegde, C. (2019). Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4773–4783.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Kasabov, N. K. (1996). Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy sets and Systems*, 82(2):135–149.

Kazhdan, D., Dimanov, B., Jamnik, M., Liò, P., and Weller, A. (2020). Now you see me (cme): Concept-based model extraction. *arXiv preprint arXiv:2010.13233*.

Kim, B., Gilmer, J., Wattenberg, M., and Viégas, F. (2018a). Tcav: Relative concept importance testing with linear concept activation vectors.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018b). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer.

Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning.

Klement, E. P., Mesiar, R., and Pap, E. (2013). *Triangular norms*, volume 8. Springer Science & Business Media.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. (2020). Concept bottleneck models.

Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.-F., Meek, C., Neville, J., et al. (2007). *Introduction to statistical relational learning*. MIT press.

Krzywinski, M. and Altman, N. (2013). Error bars: the meaning of error bars is often misinterpreted, as is the statistical significance of their overlap. *Nature methods*, 10(10):921–923.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Letham, B., Rudin, C., McCormick, T. H., Madigan, D., et al. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371.

Liu, B. (2007). *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lou, Y., Caruana, R., and Gehrke, J. (2012). Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Ma, W. J., Husain, M., and Bays, P. M. (2014). Changing concepts of working memory. *Nature neuroscience*, 17(3):347.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Houle, M. E., Song, D., and Bailey, J. (2018). Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

Marra, G., Giannini, F., Diligenti, M., and Gori, M. (2019). Lyrics: A general interface layer to integrate logic inference and deep learning. *arXiv preprint arXiv:1903.07534*.

McCallumzy, A. K. and Nigamy, K. (1998). Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer.

McCluskey, E. J. (1956). Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444.

McColl, H. (1878). The calculus of equivalent statements (third paper). *Proceedings of the London Mathematical Society*, 1(1):16–28.

McKelvey, R. D. and Zavoina, W. (1975). A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology*, 4(1):103–120.

Melacci, S. and Belkin, M. (2011). Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12(Mar):1149–1184.

Melacci, S., Ciravegna, G., Sotgiu, A., Demontis, A., Biggio, B., Gori, M., and Roli, F. (2020). Domain knowledge alleviates adversarial attacks in multi-label classifiers. *arXiv preprint arXiv:2006.03833*.

Melacci, S., Globo, A., and Rigutini, L. (2018). Enhancing modern supervised word sense disambiguation models by semantic lexical resources. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Melacci, S. and Gori, M. (2012). Unsupervised learning by minimal entropy encoding. *IEEE transactions on neural networks and learning systems*, 23(12):1849–1861.

Melacci, S., Maggini, M., and Gori, M. (2009). Semi–supervised learning with constraints for multi–view object recognition. In *International Conference on Artificial Neural Networks*, pages 653–662. Springer.

Melis, M., Demontis, A., Biggio, B., Brown, G., Fumera, G., and Roli, F. (2017). Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid. In *ICCVW Vision in Practice on Autonomous Robots (ViPAR)*, pages 751–759. IEEE.

Mendelson, E. (2009). *Introduction to mathematical logic*. CRC press.

Miller, D. J., Xiang, Z., and Kesidis, G. (2020). Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108(3):402–433.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63:81–97.

Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. (2016). Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representation*.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Morgado, P. and Vasconcelos, N. (2017). Semantically consistent regularization for zero-shot recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6060–6069.

Mpagouli, A. and Hatzilygeroudis, I. (2007). Converting first order logic into natural language: A first level approach. In *Current Trends in Informatics: 11th Panhellenic Conference on Informatics, PCI*, pages 517–526.

Najafi, A., Maeda, S.-i., Koyama, M., and Miyato, T. (2019). Robustness to adversarial perturbations in learning from incomplete data. In *Neural Information Processing Systems*, pages 5542–5552.

Naseer, M. M., Khan, S. H., Khan, M. H., Shahbaz Khan, F., and Porikli, F. (2019). Cross-domain transferability of adversarial perturbations. *Advances in Neural Information Processing Systems*, 32:12905–12915.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016b). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE.

Park, S., Park, J., Shin, S.-J., and Moon, I.-C. (2018). Adversarial dropout for supervised and semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pemstein, D., Marquardt, K. L., Tzelgov, E., Wang, Y.-t., Krusell, J., and Miri, F. (2018). The v-dem measurement model: latent variable analysis for cross-national and cross-temporal expert-coded data. *V-Dem Working Paper*, 21.

Pi, T., Li, X., and Zhang, Z. M. (2017). Boosted zero-shot learning with semantic correlation regularization. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2599–2605.

Pop, R. and Fulop, P. (2018). Deep ensemble bayesian active learning : Addressing the mode collapse issue in monte carlo dropout via ensembles.

Quine, W. V. (1952). The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Ras, G., van Gerven, M., and Haselager, P. (2018). Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 19–36. Springer.

Rathmanner, S. and Hutter, M. (2011). A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136.

Redmon, J. and Farhadi, A. (2018). Yolo v3: An incremental improvement. *CoRR*, abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " 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*, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Rivest, R. L. (1987). Learning decision lists. *Machine learning*, 2(3):229–246.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*.

Saeed, M., Villarroel, M., Reisner, A. T., Clifford, G., Lehman, L.-W., Moody, G., Heldt, T., Kyaw, T. H., Moody, B., and Mark, R. G. (2011). Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952.

Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. (2020). Toward interpretable machine learning: Transparent deep neural networks and beyond. *arXiv preprint arXiv:2003.07631*.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning.

Sato, M. and Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1870–1875. IEEE.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. *Machine Learning-International Workshop then Conference*.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach.

Serafini, L. and Garcez, A. d. (2016). Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., and Goldstein, T. (2019). Are adversarial examples inevitable? In *International Conference on Learning Representations*.

Sheatsley, R., Hoak, B., Pauley, E., Beugin, Y., Weisman, M. J., and McDaniel, P. (2021). On the robustness of domain constraints. *arXiv preprint arXiv:2105.08619*.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological review*, 63(2):129.

Simon, H. A. (1957). *Models of man; social and rational.* New York: John Wiley and Sons, Inc.

Simon, H. A. (1979). Rational decision making in business organizations. *The American economic review*, 69(4):493–513.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Soklakov, A. N. (2002). Occam's razor as a formal basis for a physical theory. *Foundations of Physics Letters*, 15(2):107–135.

Solso, R. L., MacLin, M. K., and MacLin, O. H. (2005). *Cognitive psychology*. Pearson Education New Zealand.

Song, Q., Jin, H., Huang, X., and Hu, X. (2018). Multi-label adversarial perturbations. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1242–1247.

Sotgiu, A., Demontis, A., Melis, M., Biggio, B., Fumera, G., Feng, X., and Roli, F. (2020). Deep neural rejection against adversarial examples. *EURASIP J. Information Security*, 2020(5).

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014a). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014b). Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*.

Tavares, A. R., Avelar, P., Flach, J. M., Nicolau, M., Lamb, L. C., and Vardi, M. (2020). Understanding boolean function learnability on deep neural networks. *arXiv preprint arXiv:2009.05908*.

Teso, S. (2019). Does symbolic knowledge prevent adversarial fooling? *arXiv preprint arXiv:1912.10834*.

Teso, S. and Kersting, K. (2019). Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Towell, G. G. and Shavlik, J. W. (1993). Extracting refined rules from knowledge-based neural networks. *Machine learning*, 13(1):71–101.

Tsukimoto, H. (2000). Extracting rules from trained neural networks. *IEEE Transactions on Neural networks*, 11(2):377–389.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011a). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011b). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Winston, P. H. and Horn, B. K. (1986). *Lisp*. Addison Wesley Pub., Reading, MA.

Witten, I. H. and Frank, E. (2005). Data mining: Practical machine learning tools and techniques 2nd edition. *Morgan Kaufmann, San Francisco*.

Wu, Y., Bamman, D., and Russell, S. (2017). Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783.

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. (2020). On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33.

Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240.

Yoo, D. and Kweon, I. S. (2019). Learning loss for active learning.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yu, M. and Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhai, R., Cai, T., He, D., Dan, C., He, K., Hopcroft, J., and Wang, L. (2019). Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*.

Zhao, Z., Guo, Y., Shen, H., and Ye, J. (2020). Adaptive object detection with dual multi-label prediction. In *European Conference on Computer Vision*, pages 54–69. Springer.

Zhdanov, F. (2019). Diverse mini-batch active learning.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Zilke, J. R., Loza Mencía, E., and Janssen, F. (2016). Deepred – rule extraction from deep neural networks. In Calders, T., Ceci, M., and Malerba, D., editors, *Discovery Science*, pages 457–473, Cham. Springer International Publishing.