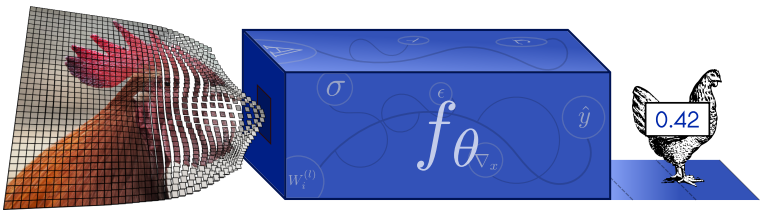
Dissertation

Towards Interpretable Models for Computer Vision



Thesis approved by the Department of Computer Science
University of Kaiserslautern-Landau for the award of
the Doctoral Degree

Doctor of Engineering
(Dr.-Ing.)

to

Sebastian Palacio Bustamante

Reviewers	Prof. Dr. Prof. h.c. Andreas Dengel
	Prof. Dr. Didier Stricker
	Prof. Dr. Marius Kloft
Date of Defense	15.02.2023
Dean of the Dept.	Prof. Dr. Christoph Garth

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

DE-386

Sebastian Palacio Bustamante
Towards Interpretable Models for Computer Vision
Dissertation. Date Viva: 15.02.2023
Reviewers: Prof. Dr. Prof. h.c. Andreas Dengel, Prof. Dr. Didier Stricker and Prof. Dr. Marius Kloft

University of Kaiserslautern-Landau
Department of Computer Science
Gottlieb-Daimler-Straße
67663 Kaiserslautern

Contact Information
`https://spalaciob.github.io`

Abstract

The rising demand for machine learning (ML) models has become a growing concern for stakeholders who depend on automatic decisions. In today's world, black-box solutions (in particular deep neural networks) are being continuously implemented for more and more high-stake scenarios like medical diagnosis or autonomous vehicles. Unfortunately, when these opaque models make predictions that do not align with our expectations, finding a valid justification is simply not possible.

Explainable Artificial Intelligence (XAI) has emerged in response to our need for finding reasons that justify what a machine sees, but we don't. However, contributions in this field are mostly centered around local structures such as individual neurons or single input samples. Global characteristics that govern the behavior of a model are still poorly understood or have not been explored yet. An aggravating factor is the lack of a standard terminology to contextualize and compare contributions in this field. Such lack of consensus is depriving the ML community from ultimately moving away from black-boxes, and start creating systematic methods to design models that are interpretable by design.

So, what are the global patterns that govern the behavior of modern neural networks, and what can we do to make these models more interpretable from the start?

This thesis delves into both issues, unveiling patterns about existing models, and establishing strategies that lead to more interpretable architectures. These include biases coming from imbalanced datasets, quantification of model capacity, and robustness against adversarial attacks. When looking for new models that are interpretable by design, this work proposes a strategy to add more structure to neural networks, based on auxiliary tasks that are semantically related to the main objective. This strategy is the result of applying a novel theoretical framework proposed as part of this work. The XAI framework is meant to contextualize and compare contributions in XAI by providing actionable definitions for terms like "*explanation*" and "*interpretation*".

Altogether, these contributions address dire demands for understanding more about the global behavior of modern deep neural networks. More importantly, they can be used as a blueprint for designing novel, and more interpretable architectures. By tackling issues from the present and the future of XAI, results from this work are a firm step towards more interpretable models for computer vision.

Preface

From the time I first embarked on my academic journey, all I wanted was to understand. There is this magnificent feeling of awe and enlightenment once a seemingly complex process has been turned apart and conceptualized into atomic pieces. I will never forget that moment as an eager freshman in computer science, when I first understood the phrase "a computer is nothing but zeros and ones." I was sitting in a dusty old lab, just large enough to accommodate about twenty students, while the professor explained how mechanical relays were first used to control the flow of electricity. There was no clear sign of where he was heading, but after a few minutes he had arrived at the development of the transistor, and the way they can cleverly be wired to represent logic gates. At that point, the slow cadence of his voice was no longer an issue, and I felt drawn to the story that all the symbols and notes in the board were already telling me. I needed to overcome my morning fatigue and stay focused because, soon enough, those logic gates were going to be elegantly arranged to form ALUs—the heart of the central processing unit—and finally, we were going to witness how gated latches turned into the firsts Random-Access-Memory components.

That lecture ended without any major eventualities, but once outside the classroom, as I was heading to the next appointment, I felt like the whole world was suddenly different. There was nothing in particular that I could point to, but it felt as if a thick layer of fog had abruptly lifted, giving me the chance to see further and more clearly. It took me just a few seconds to realize that the clarity I was experiencing came from rejoicing in the beauty of those emerging patterns we have just been taught; those same patterns that were discovered years, or perhaps decades ago by the brilliant minds I would later have to come back to for inspiration.

During the course of my bachelor studies at the "Escuela de Administración y Finanzas Técnicas" (EAFIT), opportunities to dig deep and discuss even the smallest technicalities opened up, conferring me a growing appreciation for all the ingenuity behind highly complex ideas such as programming languages, computer networks, operative systems and large databases. It was only when I arrived in Germany to specialize in artificial intelligence, that I was confronted with a seemingly unsurmountable amount of new information: multimedia retrieval algorithms, distance measures I have never heard before and theories for doing document analysis or higher-order logics. Being over nine

thousand kilometers away from home, mumbling just enough German to go grocery shopping, I felt overwhelmed. A growing sense of insecurity had exacerbated my impostor syndrome, to the point where I felt compelled to update my *curriculum vitae* and start job hunting before my impending expulsion from academia. Luckily, none of that was necessary. My early education as a software engineer allowed me to navigate some of the most complicated topics by treating them as black boxes. These pieces of mysterious and intricate mechanisms granted me gradual access to elegant solutions for problems that I was just learning to love. However, as my involvement and passion for the field grew, I quickly realized that I needed to understand more, to understand better, and to brush up on concepts I was already supposed to know. The dynamic of uncovering layers of complexity one at the time, served me well during the rest of my master studies, which I proudly completed in the unusual warmth of October 2013.

Once employed as a full-time researcher at the German Center for AI (DFKI), and motivated by the rewarding sense of conquering new problems, I found myself just a few months later timidly agreeing to develop a course, from scratch, about the hottest research topic at the time: *Deep Learning*. The catchy name had already grabbed the attention of many, even though there was little overlap between Deep Learning and anything considered state-of-the-art at that time. So there I was, barely getting acquainted with the specifics of this miracle cure that was poised to reinvigorate the otherwise sluggish advancements in multimedia retrieval. Luckily, the material I was preparing allowed me to dissect all its moving parts, studying each piece individually, and assembling them back into a coherent whole that students could assimilate. This was a wonderful opportunity to appreciate the extent of the phrase "you don't realize whether you completely understand a topic until you are tasked to explain it to someone else." [1] I would use this maxim as the standard to craft every single graphic, every text, and every table that ended up in the course material. The effort paid off quickly, not only because of the flattering feedback or the five subsequent times I was asked to give the lecture, but also because of the invaluable insights it granted me.

It came as no surprise that the topic of my PhD materialized shortly after, as a byproduct of my continuous quest for further understanding. Despite being more comfortable with the principles of Deep Learning, I would often find myself looking at a murky horizon where changes in this or that hyperparameter lead to better results without any convincing reason that could support it. The computer-vision community was elated by the fast and steady progress that Deep Learning had brought to the scene. However, programming a classifier that can reliably distinguish between images of Chihuahuas and blueberry muffins was a borderline circus act. Claiming success simply because we

[1] This version came from neurologist Dr. Steven Novella but phrases with a similar sentiment are often attributed to Albert Einstein or Ernest Rutherford.

"show the model enough samples," didn't spark the same awe-inspiring feeling that got me so infatuated when I first started my academic journey. Naturally, I felt compelled to ask more. The ambition was growing with each new paper that crossed my desk, and I savored the small victories from experiments that uncovered existing but unknown properties. Soon enough I found myself drilling down layers of dense theories and running experiments that would test my own hypotheses. I was finally working towards answering my own questions!

These are the waters that I've navigated during my academic journey. Some of them have been turbulent, like the time I set a one-year deadline to get a paper published or start looking for a position in industry. Some of them have been surreal, like the time at CVPR when I received the NVIDIA Pioneer Award from the CEO himself, Jensen Huang. In retrospective, this whole adventure has been worthwhile, and as they say, the rest is history. As I prepare to come ashore and mark the end of my doctoral adventure, the one step remaining is to document the outcomes of this wonderful voyage through questions, ideas, discussions, conferences, and lots of experiments.

Acknowledgements

Some graduate students think of their thesis as the culmination of a long journey. To me however, it is more of a milestone in the life of a curious person. As such, I want these pages to preserve some of the elements of discovery and ponder that I have encountered while developing each idea. My goal is to invite you, the reader, to tag along the path to discovery rather than just reading passively about it. I hold on to this premise to avoid soporific passages, in an attempt to produce a more writerly text that incites you to imagine and interpret. This is also why I want to introduce and rely on a discursive "we" when referring to the contributions of this work.

The use of the plural form is also a constant remainder that the outcomes described here would not be possible without the invaluable contributions of a large group of people. I want to spend a few lines to thank my first co-authors Joachim, Adriano and Shailza for their support and commitment to the ideas that pushed the boundaries of what we knew. To Fede, Tushar, Benjamin, Patrick, Fatemeh, Stanislav, Andrej and the rest of my colleagues from the Multimedia Analysis and Data Mining (MADM) group at DFKI. A special mention goes to Jörn for his candor, and for always going out of his way to procure an environment that promotes research and personal development. To the amazing scientists who selflessly shared their knowledge, and introduced me to the nuts and bolts of their trade: Adrian Ulges, Christian Schulze, Tom Breuel, Marcus Liwicki, Klaus Madlener, Xavi Giró and Didier Stricker. To Stella Yu from UC Berkeley for giving

me the magnificent opportunity to visit her lab at ICSI and infusing more passion in me for pursuing newer and bigger ideas. I thank the undergraduate students that worked tirelessly along with me, shaping the story that is now being told here: Victor, Tobias, Mariem and Philipp. To my supervisor Andreas Dengel, to whom I am truly grateful for his mentorship, support and the priceless opportunity to forge myself as a researcher and scientist. I want to extend my gratitude to Jhon, Daniel, Tillo, Carlos, Fede, Irina and all the marvelous people who have given me the honor of calling them friends. To Caro for her continuous support and encouragement whenever I needed it. To the Palacio and the Bustamante families for showing me what selfless love looks like, for always cheering for me, and for being the kind of support that can be felt across oceans. Finally, to my parents Luis and Angela, to whom I owe not only the life they gave me, but most importantly the joy, curiosity and courage to push beyond what I ever thought possible.

Publications

Parts of this thesis (including tables, figures, and algorithms) have been published in the following scientific peer-reviewed conferences:

- **Chapter 3**: Palacio, Sebastian, Adriano Lucieri, Mohsin Munir, Sheraz Ahmed, Jörn Hees, and Andreas Dengel. "XAI Handbook: Towards a unified framework for explainable AI." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3766–3775. 2021.
- **Chapter 4**: Palacio, Sebastian, Shailza, Jolly, Joachim Folz, Federico Raue, Jörn Hees, and Andreas Dengel. "P $\approx$ NP, at least in Visual Question Answering." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 2748–2754. IEEE, 2021.
- **Chapter 5**: Palacio, Sebastian, Joachim Folz, Jörn Hees, Federico Raue, Damian Borth, and Andreas Dengel. "What do deep networks like to see?" In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3108–3117. 2018.
- **Chapter 6**: Palacio, Sebastian, Joachim, Folz, Jörn Hees, and Andreas Dengel. "Adversarial defense based on structure-to-signal autoencoders." In 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 3568–3577. IEEE, 2020.
- **Chapter 7**: Palacio, Sebastian, Philipp Engler, Jörn Hees, and Andreas Dengel. "Contextual Classification Using Self-Supervised Auxiliary Models for Deep Neural Networks." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 8937–8944. IEEE, 2021.

Contents

1 Introduction

> *The most exciting phrase to hear in science, the one that heralds the most discoveries, is not "Eureka!" (I found it!) but "That's funny..."*
>
> **— Isaac Asimov**

On the brisk night of June 15th 2019, a select group of renown computer scientists left their screens and keyboards to gather at the exclusive San Francisco Palace Hotel. That night, now as attendees to the ACM Awards Banquet, they would witness the moment when Yoshua Bengio, Geoffrey Hinton, and Yann LeCun received the 2018 Alan M. Turing Award—the most prestigious distinction in the field of computer science. The Association for Computer Machinery had chosen the trio "for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing."[1] These three professors had officially joined the likes of E. Dijkstra, D. Knuth, J. McCarthy and R. Hamming in the Olympus of computer science. Through their research, deep neural networks (DNNs) were now considered as fundamental to the field as other technologies like computer networks, databases, and cryptography.

Despite being based on foundations that matured during the early 1990s, the impact of "Deep Learning" has mostly been felt during the course of the last decade. However, the growing enthusiasm around this technology has caused not only massive leaps in scientific advancements, but it also precipitated its rapid adoption for all kinds of problems. This branch of machine learning (ML) has shown to be more effective than previous state-of-the-art technology, which used to rely on *innate* methods *i.e.*, algorithms that don't require training.

Access to more powerful computers—especially to graphic processing units (GPUs)—made it possible to train bigger and deeper networks. The particulars of the problem became almost irrelevant as long as the right amount of data was available. In addition, ingenious extensions to the backbone of neural networks facilitated their implementation for problems where the number of operations were still prohibitively large. Prominent

[1] https://www.acm.org/media-center/2019/march/turing-award-2018

contributions include depth-wise convolutions [32], residual connections [59] or inception layers [142]. Notably, these technologies have been primarily showcased for solving computer vision problems such as image similarity, scene classification or object detection.

The overarching success of neural networks, propelled their usage for new domains and under different constraints, leaping from niche academic circles to mainstream industrial applications. Higher standards were required, as these models were gradually used for more high-stake scenarios like credit assignment [171], criminal recidivism [1], or medical diagnosis [56]. For some of these, privacy constraints or difficulties procuring expert annotations results in data not being readily available. For example, the International Skin Imaging Collaboration (ISIC) hosts one of the largest datasets for melanoma prediction. For every single sample, they have to ensure that the patient's privacy is preserved, and confirm that the labels for classification are provided by medical experts—arguably a scarce and expensive resource! Moreover, policymakers are passing regulations to hold models (or rather, the organizations behind them) accountable for their decisions [166]. The main goal is to enforce transparency for users whose lives are affected by predictions coming from an automatic model.

The question that naturally arises is: Are there systematic ways to improve both performance and transparency at the same time? In response to this question, eXplainable AI (XAI) came into being. To understand why this is the case, we have to look back at state-of-the-art AI from the previous decade. In the time before Deep Learning, extracting information and issuing predictions mostly followed a traceable pipeline, with intermediate steps that had a direct relation to high-level semantics. Back then, the use of innate methods such as blob detectors [12] or local texture descriptors [160] was predominant in computer vision. Blobs were meant to select salient and stable points of the image, while descriptors provided a compact representation of the most relevant characteristics within each region. These set of features were aggregated into global descriptors (one per sample), and used by powerful but straightforward classifiers like decision trees (DT) or support vector machines (SVMs).

As Deep Networks started dethroning the most advanced models of the time, the field of AI accepted the trade-off between traceability and performance. These networks were now the ones deciding what to detect and what characteristics to represent. Their complex web of interconnected parameters allowed them to find highly non-linear patterns that proved effective even on the most challenging benchmarks. Unfortunately, these patterns were no longer controlled by human experts, but were rather data-driven. Even though traces of semantic relations can be drawn—after all, their components are loosely inspired by the human brain—they are still somewhat accidental. Alex

Kyrzhevsky's seminal paper already noted that their neural network converged to color-, frequency- and orientation-selective filters [76]; three mechanisms that have been widely used in the past to detect and extract visual features.

XAI emerges as a conglomerate of efforts to claim back the traceability that got lost with Deep Learning. Even without a formal definition, an eager community has recently blossomed around the idea of developing AI that can be explained. In 2019, XAI established itself as a field of its own after appearing at the top of Gartner's AI Hype Cycle [112, 20]. Advances in this area focus on three pillars for next-generation AI: high-stake scenarios, legal requirements and scientific advancement. The most important milestones for this community include the development of tools, theories, modifications, strategies and experiments to bridge the gap between the opaque high-performance of Deep Learning and traceable methods.

Somewhat unexpectedly, the field has faced a few obstacles that have slowed down progress, taming expectations on what can be achieved. A predominant issue is the lack of a unified, precise definition for what actually counts as "explainable." Not having one is especially troublesome for a field of research that is expected to provide standards for governance, and across industries. We address this issue in Chapter 3, arguing why standard definitions are urgently needed before any specific solution can be pushed forward.

These are the origins and current state of XAI. But before we start talking about the contributions that this thesis makes to the field, we first need to understand two important properties that have been attributed to XAI. They will allow us to contextualize the gap that every contribution in the field (including those in this thesis) addresses.

1.1 Unveiling the XAI Spectrum

Finding structure in the sea of contributions for XAI has been nothing short of a titanic task. There are not only numerous survey papers for explainable methods, but there are now reviews *about* survey papers as well! Two well-known meta-surveys have been published by Vilone and Lungo [151], and more recently by Ras, Xie *et al.* [120]. Together, they have analyzed over 500 explainability-related papers published in the span of two decades. Even though both meta-surveys consider the most foundational algorithms in the field, each publication proposes a rather different taxonomy for organizing XAI methods. Vilone and Lungo present a stratification where XAI methods fall into one of three categories: visualization, distillation or intrinsic. In contrast, the

structure proposed by Ras, Xie *et al*. tries to identify five fundamental characteristics from each XAI method: stage, scope, problem type, input type and output format.

At first, it seems as if not even surveys can find a common pattern in the reviews they conduct. However, there are two notions in common that can be extracted from both.

The first notion refers to the scope that explanations are meant to cover. By "scope", we refer to elements of the model that an explanation is addressing. Whenever an explanation focuses on small parts of the model *e.g*., an individual neuron, or a single input sample, we say that the scope of the explanation is *local*. Meanwhile, methods that produce explanations pertaining to the whole model, or to the entire data distribution are called *global*. For Vilone and Lungo the scope appears directly as one of their main categorization points. Meanwhile, this distinction does not appear directly in the taxonomy from Ras, Xie *et al*. However, they repeatedly refer to global and local properties when highlighting important features of each method.

The second notion refers to the "stage", or the moment in time when the explanation method takes place. On one end, explanations may happen *after* the target model has been already designed, trained, and fine-tuned. On the other end, we have explanations that are defined *before* the model is ready to make predictions. These two notions are commonly referred to as *post hoc* and *ante hoc*[2] respectively. Once more, Vilone and Lungo identify "stage" explicitly in their XAI taxonomy, drawing a clear line between the two extremes. For Ras, Xie, *et al*. the distinction is embedded in the attributes of their three categories: visualization and distillation is only comprised of *post hoc* methods, while *ante hoc* methods belong exclusively in the intrinsic category.

Together, the notions of scope and stage convey some of the most fundamental properties for XAI because they help determine *where* the explanation is focused, and *when* it happens. However, this juxtaposition of local vs. global and *post hoc* vs. *ante hoc* describes a discrete space that leads to a false dichotomy. While determining the stage of a new explanation, focusing on a single neuron will clearly result in a local analysis. But what about methods that concentrate on an entire layer? How about exploring the behavior of an entire residual block or an inception module? What is the stage of an explanation that looks into the deeper half of a 270-layer DenseNet? Clearly, these questions cannot be answered with a simple binary decision.

Similarly, pinpointing the precise moment when an explanation kicks off, presents its own set of challenges. In between *ante hoc* explanations and *post hoc* visualizations we

[2]The term *ante hoc* is preferred over *pre hoc* as the antonym for *post hoc*. Even though both refer to something that happens "before this", the prefix *ante* has a temporal connotation, similar to the abbreviation A.M. when reading time. The root *pre* or *prae* on the other hand, denotes more of a spatial relationship between objects.

can find mechanisms that take place while the model is still forming. Think for example about transfer learning. Here, a model has been fully trained already, but it then gets one or more layers adjusted to fit data from another problem. An XAI method that relies on transfer learning is therefore not going to fit into the definition of either *ante hoc* or *post hoc*. This is also the case for other mechanisms like self-supervised learning, test-time adaptation, model distillation or curriculum learning.

So, instead of adhering to this unaccommodating binary separation, we can think of stage and scope as two continuous variables that are orthogonal to each other (see Figure 1.1). This plane serves as a frame of reference for contributions made in the field of XAI. The continuity in both axes reflects the countless options to define what the explanation is focusing on, and when.

With this frame of reference, we can have a better overview of the contributions presented in this work.

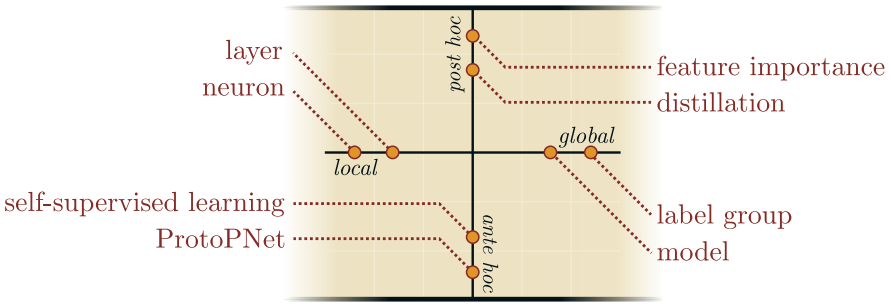


Fig. 1.1.: Notions of stage (*post hoc* vs. *ante hoc*) and scope (local vs. global) are extremes in their own continuous spectrum. Various techniques used for explainability can be placed along both axes.

1.2 Contributions of this Thesis

This work comprises one theoretical contribution, evaluations of three global properties of DNNs, and one practical application. Each is discussed in a different chapter and addresses a specific research question.

Chapter 3: Is there a unified way to compare advances in XAI? As the first contribution of this work, we introduce the XAI Handbook: a framework to unify and compare XAI methods. Recall from before, that a lack of unity when talking about fundamental constructs such as “explanation” or “interpretation” leads to developments that end up tackling different problems. Without a unified way to talk about XAI, it becomes

impossible to establish standards for governance or industry. With the XAI Handbook we provide the necessary tools that enable a universal and commensurable comparison between XAI methods.

Chapter 4: Are classifiers negatively affected by relying on datasets with overrepresented, and semantically related classes? Deep Networks are prone to exploit simple patterns in the data, even when these patterns are not the ones that should be used. One such common "shortcut" materializes when parts of the training data are overrepresented, resulting in predictions that are biased towards classes that occur more often. We study the effects of training a classifier under such circumstances in the domain of Visual Question Answering (VQA). In particular, we look at the interaction between the set of polar samples (*i.e.*, *yes* or *no* questions) and non-polar counterparts. Results strongly indicate that overrepresented polar classes play a reinforcing role for the remaining classes, and not a detrimental one.

Chapter 5: What constitutes "model capacity" and how can we measure it? Across the literature, we find countless references to the notion of "model capacity" when talking about networks that represent more or less complex patterns. However, a precise measure of "capacity" remains elusive. We propose a method to quantify model capacity as the input information that a classifier takes in before making a prediction. This method can be used to compare different high-performance classifiers in a way that is not directly related to prediction accuracy, or to the number of trainable parameters. Instead, this method reveals how much information is being used by a model. We study the implications of our proposed metric and draw connections to prior findings that remained unaccounted until now.

Chapter 6: Can our new understanding of "capacity" help improve any aspect of the model? As a use case, we exploit our proposed method for measuring model capacity to increase the robustness of image classifiers. In particular, we look at one of the greatest challenges that Deep Learning is facing today: adversarial attacks. We evaluate high-performance classifiers against standard but powerful adversarial attacks, and show that these models can attain a high level of resiliency. This is possible thanks to the proposed method for quantifying the input signal from Chapter 5, which can be used to let the necessary information reach the classifier while filtering out everything else.

Chapter 7: Is it possible to convey structural knowledge without resorting to additional labels? For creating models that are explainable by design, we need mechanisms that

convey more information about the non-functional requirements of a task. As the last contribution in this thesis, we design a neural network for solving two tasks based on the input of only one task. For this, an existing dataset is extended with new classes by grouping samples from different labels. The new groupings express a hierarchical context with respect to the original classes. By training and evaluating using this strategy, we show that these models can attain a consistently higher performance. More importantly, the additional contextual outputs provide information to understand and justify why a certain prediction has been made, be a correct or a wrong one.

As we can see, these five contributions cover a broad scope in terms of tasks, objectives and domain of applicability. For an overview of where and when these contributions fit in the spectrum of XAI, we can use the stage-scope axis we defined before, and plot where each contribution falls:

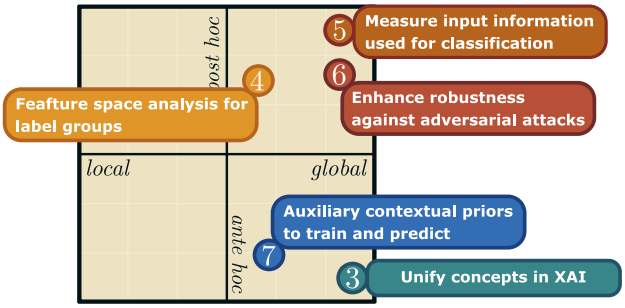


Fig. 1.2.: XAI methods proposed in this thesis, and visualized in the stage-scope spectrum. We propose two methods for explaining global properties (brown and yellow) and propose a technique to create intrinsically explainable models (in blue). The number on each point refers to the chapter where it is discussed.

From Figure 1.2, we can see that this work focuses more on the global patterns that govern neural networks. At the same time, these methods cover the stage axis by making contributions that apply to existing methods, while also demonstrating a general way to design models that are explainable by design.

This leads us to the two underlying questions of this thesis:

1. What global patterns can we explain from existing state-of-the-art image classifiers?
2. Can we create explainable models by representing additional knowledge without resorting to more annotations?

In the next chapter, we go over foundations that are necessary before diving into the specifics of each contribution. These are meant to serve as a quick reference and offers complementary details for concepts that are used in later chapters.

2 Background

This chapter is aimed at giving succinct overviews about the methods and models that have been used throughout this thesis. Background and theoretical foundations are split into independent sections that can be used as a reference for concepts that are leveraged in the following chapters.

2.1 Deep Neural Networks

"Deep Neural Networks" (or DNNs) became a popular term to refer to a broad variety of artificial neural networks (ANNs) that can deal with large-scale problems. ANNs in turn, can be generally described as a parametrized method for approximating non-linear functions. The name "neural network" comes from the structure that is used to approximate functions, namely with a series of simpler operations that are chained together, forming a "network" of transformations.[1] These individual functions are referred to as "layers" in the context of ANNs.

This section reviews the core components of high-performance DNNs that play an important role in this thesis. We will go over two types of network layers, as well as additional operations that are often included as part of the network architecture.

2.1.1 Dense Layers

Also called feed-forward layers, these are one of the original functions that were used to design neural networks. At their core, these layers are made of parametrized linear functions $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} = \hat{\mathbf{z}}$ that are in turn, the input for a non-linear function $\sigma(\hat{\mathbf{z}})$. In this case, the parameters that determine how the function fits the data, are contained in $\mathbf{W}$ and $\mathbf{b}$. See Section 2.1.2 for more details about specific non-linear activations.

As mentioned before, these operations are then chained together to allow more complex functions to be modeled. For example, an ANN made of three dense layers $f^{(1)}, f^{(2)}, f^{(3)}$

[1] Also, because their first installments were loosely inspired by structures found in the brain.

can be arranged as $\sigma(f^{(1)}(\sigma(f^{(2)}(\sigma(f^{(3)}(\mathbf{x})))))) = \hat{\mathbf{y}}$. In this case, each of the layers has its own set of parameters $\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}, \ell \in \{1, 2, 3\}$, which are collectively referred to as *model parameters*, and are often denoted by the symbol $\boldsymbol{\theta}$.

2.1.2 Non-Linear Activations

Non-linear activations are functions that are non-linear, and are used in neural networks to give them the ability to approximate intricate functions beyond the linear domain. Which non-linear function is chosen depends on factors like to co-domain of the function or properties of its derivative that makes it better suited for optimization problems.

Common non-linear operations include the Rectified Linear Unit (ReLU) [48], Exponential Linear Unit (ELU) [34], sigmoids or hyperbolic tangents.

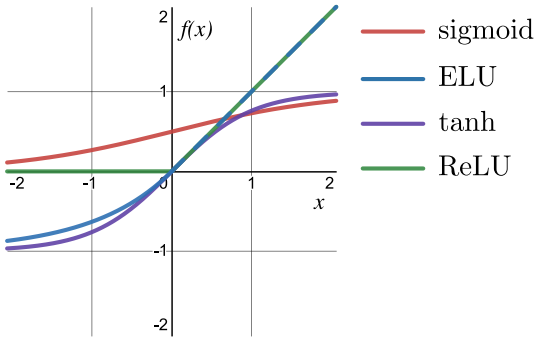


Fig. 2.1.: Non-linear functions used for ANNs.

A variation used for sequence models such as LSTMs [61] or GRUs [31] is the gated tanh. This function consists of applying a hyperbolic tangent (tanh) to a linear product, and then doing an element-wise multiplication with values between 0 and 1. More formally:

$$\text{Gated-tanh}(\mathbf{x}) = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b}) \otimes \sigma(\mathbf{V}\mathbf{x} + \mathbf{c}) \tag{2.1}$$

where σ is the logistic function, $\mathbf{V}, \mathbf{W}, \mathbf{b}, \mathbf{c}$ are trainable parameters, and $\otimes$ is the element-wise product. Because the second term has a co-domain between zero and one, we say that its values are coefficients that "gate" the result from the first term.

To constrain the properties of the output, so that it can be read as a probability distribution, a normalization function called *softmax* is used:

$$\textit{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{D} e^{x_j}} \tag{2.2}$$

where D is the number of dimensions of $\mathbf{x}$. This function is often used as the non-linear operation of the last layer of a DNN.

2.1.3 Convolutional Layers

These are layers for ANNs that define an operation for grid-like structures. This is especially useful for processing images, since they are represented as 2D or 3D tensors. Similar to dense layers, convolutions are a weighted sum between an input $I \in \mathbb{R}^{H \times W \times C}$ and weights in the form of a kernel $K \in \mathbb{R}^{H' \times W' \times C}$. The main difference is that the kernel is usually much smaller than the input *i.e.* $H' \ll H$, and $W' \ll W$ (note that the number of channels C remains constant). Considering the indices i, j relative to the input, the result of a single convolutional operation is given by:

$$S_{i,j} = \sum_{h=1}^{H'} \sum_{w=1}^{W'} I_{i+h,j+w} K_{h,w} \tag{2.3}$$

In order to apply the kernel to the full image, we say that the kernel "slides" over the input. For Equation 2.3 it means that the indices i, j change, which will cause the convolution to be applied to other areas of the image. In its most simple form, both i, j increase monotonically by one until the area of the image has been covered at least once by the kernel K. The result is a matrix S that is usually referred to as a "feature map" or an "activation map". Finally, the feature map passes though a non-linear function that is applied element-wise.

A convolutional layer comprises a set of C' kernels with the same spatial dimensions W', H'. The corresponding 2D activation maps $S^1, S^2, \ldots, S^{C'}$ are concatenated along a third dimension creating a new tensor with C' channels. This way, the convolution operation can be applied again on this result, creating a chain of convolutions that are commonly known as convolutional layers.

Note that this operation is slightly different from the original convolution defined for signal processing.[2] For a more detailed discussion on the particular differences and their implications, please refer to the reference book by Goodfellow *et al.* [50].

[2] Strictly speaking, Equation 2.3 defines a convolution with a flipped kernel, and it is better known as a "cross-correlation".

2.1.4 Pooling Layers

Pooling defines a family of operations that are designed to sub-sample a feature map. This is useful to control the size of the intermediate activations as they get progressively transformed by the layers of a DNN. The sub-sampling takes place by sliding a window through the activation, and computing a summary statistic for the values of the activation that fall into that window. This is similar to the way a convolutional layer works, but without a trainable kernel.

Two of the most common pooling operations are known as *max-pooling* and *average-pooling*. For *max-pooling*, the highest value in the window is reported, whereas for *average-pooling*, as the name suggests, the average value from the region is computed. Most modern networks apply pooling operations with windows that slide in a non-overlapping fashion *i.e.*, increments of the indices correspond to the size of the window itself. This way, a $32 \times 32 \times 128$ activation map gets reduced to $16 \times 16 \times 128$ after applying pooling with a window of $2 \times 2 \times 1$. Note that pooling is usually applied along the first two dimensions, as these are conveying spatial relations, whereas positions along the third dimension (*i.e.*, the channels) are independent of each other.

In some cases, a special kind of pooling is used to control the transition between convolutional and dense layers. These are called global-pooling, and they work by using the size of the current activation map as the size of the window. For an activation of size $32 \times 32 \times 128$, applying a global pooling results in a $1 \times 1 \times 128$ output. In general, an activation of size $H \times W \times C$ will get reduced to $1 \times 1 \times C$. The most common operation for pooling this way is the *global-average pooling* or GAP for short.

2.2 Supervised Classification

The problem of supervised classification for images can be defined as follows. Let $\mathcal{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ be a list of tensors that correspond to image samples, and let $\mathcal{Y} = \{y^{(i)}\}_{i=1}^{N}$ denote the corresponding labels for each image in $\mathcal{X}$. In a closed-world scenario, we assume that the number of labels is finite and known *i.e.* $y \in \{1, 2, \ldots, k\}$. The assignment of $\mathbf{x}^{(i)}$ to $y^{(i)}$ comes from sampling a distribution that is unknown, and that we want to approximate. Therefore, the goal is to create a function $f : \mathbb{R}^{d_\mathbf{x}} \to \{1, 2, \ldots, k\}$ that maps samples from $\mathcal{X}$ to their corresponding label in $\mathcal{Y}$.

To find a suitable function f that can reproduce these mappings, we can define a generic DNN architecture based on a mixture of convolutional, dense and pooling layers. The

general architecture of an image classifier comprises convolutional and pooling layers at the beginning of the network, and dense layers towards the deeper end of the model.

To make the architecture amenable to differentiable optimization (see Section 2.2.1), ground-truth labels are represented as a one-hot encoded vector $\mathbf{y}^\star = \mathbb{1}(y) = I_{y,\star}$ where I is the $k \times k$ identity matrix and $y, \star$ denotes the y-th row of I. At the same time, the last layer of the classifier is a dense layer with a k-dimensional output (known as the *logits*), and instead of one of the usual non-linear activations, a *softmax* operation is used (see Section 2.1.2). Mapping back a k-dimensional vector to the domain of $\mathcal{Y}$ is possible through the $\arg\max$ function on either the normalized output or the logits.

At the end, we are interested in finding values for all trainable parameters $\boldsymbol{\theta}$ that are defined in f, such that $f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$ yields the corresponding value $y^{(i)} \in \mathcal{Y}$.

2.2.1 Finding Values for $\boldsymbol{\theta}$

Finding values for $\boldsymbol{\theta}$ that approximate the function $f_{\boldsymbol{\theta}}(\mathbf{x}) \approx y$ is achieved through stochastic optimization methods.

The problem is set up as a cost-minimization function:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbb{1}(y^{(i)})) \tag{2.4}$$

where $\mathcal{L}$ is a proxy for the empirical risk; usually the negative log-likelihood (NLL) a.k.a. cross-entropy. This way, when dealing with single-label, multi-class classification scenarios, the optimization problem becomes:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} -\log f(\mathbf{x}^{(i)}; \boldsymbol{\theta})_{y^{(i)}} \tag{2.5}$$

For problems involving very deep networks and large datasets, computing the sum over every sample is computationally prohibitive, in particular because of memory constraints. To cope with this limitation an approximation of Equation 2.5 is computed over consecutive, small batches of randomly sampled data called *mini-batches*. Computing the loss over all mini-batches that make up the dataset is called an *epoch*.

2.2.2 Optimizers

Common algorithms to minimize Equation 2.5 use first order derivatives of the loss w.r.t. the trainable parameters $\boldsymbol{\theta}$. Since the loss is computed over several mini-batches, and the solution space is highly non-convex, updates to the trainable parameters are kept small via a coefficient known as the learning rate.

One of the most simple algorithms that is still widely used to train DNNs is called *stochastic gradient descent* (or SGD). As mentioned earlier, the loss function is evaluated on a mini-batch, and then the gradient w.r.t. trainable parameters is subtracted from the current parameters, after being modulated by the learning rate:

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{old} - \lambda \frac{\partial \mathcal{L}_B}{\partial \boldsymbol{\theta}_{old}} \tag{2.6}$$

where $\mathcal{L}_B$ is the average of the cross-entropy over a mini-batch with B samples, and λ is the learning rate. Equation 2.6 is computed for consecutive mini-batches, updating $\boldsymbol{\theta}$ until the loss reaches a certain stopping criterion. This criterion can be based on the number of iterations, or the value of the loss over the last couple of batches.

There are two relevant additions to SGD that have been proven beneficial for the optimization process. The first consists on adapting the value of the learning rate when more iterations of SGD are computed. This temporal adaptation is known as a learning rate schedule. Commonly used schedules for the learning rate include:

- **Step-wise learning rate**: The value of λ is updated after a fixed number of iterations. For example, when training for 90 epochs, the update rule $\lambda_{new} = 0.2\lambda_{old}$ will be executed after epochs 30 and 60. Note that there is nothing special about the values 0.2, 30 or 60; it is only important that the learning rate decays over time.

- **Linear decay**: For each new epoch $\lambda_e = \lambda_0 - (1 - \frac{e}{E})$, where e is the current epoch, λ_0 is the initial learning rate, and E is the total number of epochs that are planned for training.

- **Triangular rate**: Similar to a linear decay schedule, only that during the first few epochs, the learning rate grows linearly from zero to its highest value, before starting to decay again.

Other strategies that are also used such as cosine annealing [94] or cyclical learning rates [135] have shown to be helpful, but are out of the scope of this thesis.

The second way to improve SGD is by using an exponentially decaying moving average of previous gradients, resembling the physical notion of momentum. Because the gradients are computed on small batches, they are often noisy and do not point exactly in the direction that minimizes the loss for the entire distribution. The use of momentum helps to mitigate variability that comes from noisy gradients. The way it is implemented is by introducing an extra variable referred to as velocity:

$$\begin{aligned} \mathbf{v}_{new} &= \alpha\, \mathbf{v}_{old} - \lambda \frac{\partial \mathcal{L}_B}{\partial \boldsymbol{\theta}_{old}} \\ \boldsymbol{\theta}_{new} &= \boldsymbol{\theta}_{old} + \mathbf{v}_{new} \end{aligned} \tag{2.7}$$

where $\alpha \in [0, 1)$ controls how much of the previous gradients are preserved. When talking about the value of the momentum, we refer to the value of α.

Optimizers with Variable Learning Rate

Instead of modulating the scale of gradients using a global learning rate, researchers have looked into the effects of having individual values scaled independently, and adapt automatically over time. One of the first versions of adaptive learning rates was proposed by Duchi *et al.* [42] and was called AdaGrad. This method scales individual gradients by the inverse of the squared root of the accumulated sum of squared gradients. That is, for each individual parameter θ_i and its corresponding gradient of the loss ∇_i:

$$\theta_i^{new} = \theta_i^{old} - \lambda \frac{\nabla_i}{\sqrt{G_i + \epsilon}} \tag{2.8}$$

where G_i is the sum of squared gradients computed so far, and ϵ is a small enough value to prevent a division by zero. Note that the learning rate λ is still present in the computation. This means that this method still depends on the choice of learning rate. However, it has been empirically shown that the suggested default value of 1×10^{-3} works well for the vast majority of cases.

As summing over long training times quickly leads to small gradients and thus, very small updates for the trainable parameters, Hinton *et al.* [60] propose changing the accumulated sum by an exponential moving average. This method is known as RMSProp and has shown to work better than AdaGrad for training neural networks, as the latter is not well-equipped to work on non-convex surfaces. With RMSProp, early gradients become less relevant which keeps the optimization moving further, and towards more convex areas of the parameter space.

Right after, Kingma *et al*. [73] proposed a family of optimization strategies that combines RMSProp with momentum. Their *adaptive moment estimation* method (or Adam for short) keeps an exponential moving average of both individual gradients and momentum estimates. A generalization of this method was proposed in the same study, where the root of the sum of squared gradients is replaced by the more general $p-$norm (*i.e*., the n-th root of the sum of gradients to the power of n). A special case is known as Adamax, which uses the infinity norm $p = \infty$.

Optimization methods for non-convex problems is a research field of its own, and most of the details have been left out, as they fall out of the scope of this thesis. For more information about the relation between these optimizers and many others that became popular for training DNNs, refer to the excellent overview made by Sebastian Ruder.[3]

2.2.3 Metrics

Once a DNN has been trained, we are interested in evaluating if the learned parameters can predict samples that have not been seen during training. In other words, we want to evaluate if the learned function is representing features of the input distribution, and not only to the sample that was used for training.

Independent of the specific metric, generalization is measured by splitting the available dataset into two splits: one for training and one for testing. The model is then trained using the training set, and once it has converged, the generalization performance is measured on the test set.

During the process of designing a model, many hyperparameters (*e.g*., learning rate, non-linear activation, number of layers, size of the convolutional kernels) have to be adapted. To prevent biases in the evaluation coming from manually overfitting the hyperparameters to the test set, the training data is split once more into two sets: one for training, and one for measuring generalization while hyperparameters are still being developed. The model is then trained using the training set, and evaluated on the development set (also referred to as the *validation* set). If the model doesn't need to be adapted any further, the model is trained once more using both the training and development set. Finally, the generalization performance is computed only once on the test set.

For image classifiers, the most common metric is accuracy *i.e*., the average number of correctly classified samples. Common variations of accuracy include the top-1 and top-5 accuracy. The main difference between the two is the number of predicted labels that

[3]URL: `https://ruder.io/optimizing-gradient-descent/index.html`

are considered when establishing if the prediction is correct. For top-1 accuracy, only the label with the highest probability (assuming the output can be read in a probabilistic way *e.g.*, when using a *softmax* operation) is compared to the ground-truth. For top-5 accuracy, we use the five predicted labels with the highest probability.

2.3 Autoencoders

Autoencoders are ANNs that learn to reproduce the information of the input. In other words, they are trained to learn the identity $f(\mathbf{x}) = \mathbf{x}$. Apart from obvious, trivial solutions to achieve this, autoencoders are useful to learn intermediate representations that have more interesting properties such as sparseness or robustness to noise.

Typically, an autoencoder consists of an encoder and a decoder module, which are designed as mirrored structures. The encoder compresses the input and produces a code h which is smaller than the input. The code h enters the decoder, which scales it back up to the size of the input. This structure is better known as an undercomplete autoencoder.

One of the most interesting aspects about autoencoders, is that they can be trained in an unsupervised way. This means, that training is based only on the samples $\mathbf{x} \in \mathcal{X}$, and there is no need for additional annotations like in the supervised case (see Section 2.2). A common loss function to train an autoencoder is the mean squared error (MSE) between the original sample $\mathbf{x}$, and the reconstructed sample $f(\mathbf{x}) = \hat{\mathbf{x}}$:

$$\text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{D} \sum_{i}^{D} (x_i - \hat{x}_i)^2 \tag{2.9}$$

where D is the size of $\mathbf{x}$ and $\hat{\mathbf{x}}$.

In computer vision, autoencoders have been used for image denoising [97] or feature representation [14]. A common architectural principle for autoencoders based on DNNs is to pass features produced by the encoder directly to the mirrored section of the decoder. Instead of an entire activation map, networks used for segmentation have proposed using the indices of pooling layers as additional information to pass on to the decoder. When the decoder is scaling up the current feature activation, these indices have shown to improve the reconstruction quality. Using indices of a pool operation to scale up an intermediate activation is known as *unpooling* [110].

2.4 Visual Question Answering

The field of Visual Question Answering is a relatively new addition to the set of problems for computer vision. Being fueled by the success of DNNs on computer vision and natural language processing, researchers started looking for a way to tackle problems that required both. The work of Antol *et al*. [7] has often been cited as the first to propose this task in particular. Even though there have been others who had already proposed similar ideas, it was the work of Antol and his colleagues who first proposed a free-form, open-ended format to ask and answer questions about images.

The most popular benchmark for VQA is VQA 2.0 and consists of 658,111 triplets (Q, I, A_{10}) where Q is a string that corresponds to the question, I is one out of 123,287 images from MS-COCO [89], and A_{10} is a multiset with ten answers given by human annotators. As the answers may not always have a 100% agreement, the authors of the VQA challenge proposed an alternative way to measure accuracy:

$$\text{acc}(\hat{x}, A_{10}) = \min\left(\frac{\mu(\hat{x}, A_{10})}{3}, 1\right) \tag{2.10}$$

where $\mu(x, X)$ is the multiplicity function (*i.e*. how many instances of x are in the multiset X).

Most state-of-the-art solutions for the VQA problem use a classification architecture by focusing on the K most popular answers, and treating them as independent classes. It is common to consider only answers that appear at least eight times in the dataset. Under these conditions, a model can be trained using standard metrics for classification problems such as the negative log-likelihood.

For handling text, it is common to use word embedding techniques [95, 5, 148] based on LSTMs [61], GRUs [31], and GloVe embeddings [115].

2.5 Adversarial Perturbations

Adversarial Perturbations are small, additive changes that are intentionally added to cause a model to make a prediction mistake [49]. They were discovered in 2013 when Szegedy *et al*. [144] were trying to find hard negative samples for training.

Formally, an adversarial perturbation is a tensor $\boldsymbol{\delta}$ such that:

$$f(\mathbf{x}) \neq f(\mathbf{x} + \boldsymbol{\delta}),\ ||\boldsymbol{\delta}||_p \leq \epsilon \tag{2.11}$$

where $f : \mathbb{R}^{d_{\mathbf{x}}} \rightarrow \{1, 2, \ldots, k\}$ is a classifier and ϵ is a small bound with respect to the p-norm being used. Intuitively, this problem looks for values of $\boldsymbol{\delta}$ that are imperceptible for humans. In practice, the most common norms used to measure $\boldsymbol{\delta}$ are the 0-, 2- and ∞-norms, whereas attack algorithms focus on the last two. For images, perturbations are usually between one and sixteen pixel values for these two norms.

Most attack mechanisms find perturbations by exploiting the differentiable nature of the model, and use optimization processes that maximize or minimize a proxy function of the training loss. To achieve this, they rely on gradients produced by the target model, or even a surrogate one (in case the attacker doesn't have access to gradients of the original model). Note that, for adversarial attacks, gradients are computed w.r.t. the input, and not the model parameters:

$$\nabla_x = \frac{\partial \mathcal{L}(f(\mathbf{x}; \boldsymbol{\theta}), y^\star)}{\partial \mathbf{x}} \tag{2.12}$$

Attacks trying to make the model predict any class other than the true label, attempt to maximize the loss function of the model w.r.t. the original ground-truth $y^\star$. This setting is known as an *untargeted* attack. In contrast, an attack aimed at making the model predict a specific label $y_t \neq y^\star$, rely on minimizing the loss w.r.t. the adversarial target y_t.

A wide range of strategies to make models more robust against adversarial attacks have proven to be ineffective again and again [8, 9, 23, 22]. Knowing that attackers will rely on gradients that come from the model, defense mechanisms have focused on ways to modify these gradients. However, Athalye *et al*. [9] showed that these attempts can be circumvented by slight adjustments of the attacks hyperparameters.

For defenses based on non-differentiable methods, they proposed a counter-attack mechanism called *Backward Pass Differentiable Approximation* (BPDA). BPDA exploits the transferability of adversarial attacks *i.e*., that attacks created for one model, can be successfully used to fool a second model. With this in mind, BPDA approximates the part of the defense that is non-differentiable with a differentiable one (usually a neural network). It then uses gradients from the differentiable replacement to construct adversarial perturbations.

For defenses that use stochastic operations (*e.g*. random noise, GANs, VAEs), the *Expectation over Transformation* (EoT) can be used to circumvent them. The intuition is simple: when crafting adversarial perturbations, an attacker can include knowledge about the distribution of the perturbation, and optimize for a set of samples (instead of just one).

These two principles (non-differentiable modules and stochastic operations) account for a large portion of defense strategies that have been proposed in the literature. Together, these two patterns are referred to as *gradient obfuscation*, and are no longer considered strong defense mechanisms.

Finally, Athalye and his colleagues made a list of conditions to tell if a defense strategy is relying on gradient obfuscation. In brief, the defenses have to exhibit more vulnerability to iterative attacks than to single-step variants. Attacks crafted on gradients from the model itself should be more effective than those crafted on a different model. Random sampling should not be more successful than any gradient-based attack. Lastly, an increasing bound for ϵ should result in more effective attacks, and completely ignoring the bounds, should yield a 100% attack success ratio.

2.6 Datasets for Computer Vision

One of the main factors that catapulted the success of Deep Learning for computer vision, is the availability of large-scale datasets. For image classification, there are numerous options with hundreds of thousands or even millions of images. In this thesis, we used five of the most popular ones, where four of them have annotations for supervised training.

- **CIFAR100** [75]: contains 60,000 color images of size 32×32 that belong to one of 100 different classes. The list of classes includes different kinds of animals, furniture, vehicles, plants, people, etc. Standard splits consist of 50,000 samples for training (500 images per class) and 10,000 for testing (100 per class).

- **TinyImageNet** [86]: comprises 110,000 color images of size 64×64. They are split into 200 natural categories based on those from ImageNet *e.g.*, animals, food, furniture. They are divided into 100,000 samples for training and 10,000 for validation. There is an official testing set, but it does not come with labels. Hence, we use the official validation set as test set. For validation, we use a small portion of the training set instead.

- **ImageNet** [124]: it is one of the largest image classification datasets available with 1.2 million images across 1000 classes. The size of each image is variable, but samples are commonly scaled down to 256×256 pixels. Similarly to TinyImageNet, we use the 50,000 validation samples for testing and in turn, we take a small portion of the training set for validation.

- **YFCC100m** [149]: this dataset is one of the largest publicly available dataset of multimedia content, with roughly 99.2 million photos and 0.8 million videos hosted on Flickr, and shared under one of the various Creative Commons licenses. The dataset comes with the meta-data of the original images, but no curated labels or splits between training or validation of any sort. Images are mostly pictures taken by people all over the world, with a wide variety of scenes, objects and natural landscapes.
- **VQA 2.0** [53]: one of the most popular datasets for Visual Question Answering (VQA), and an extension of the VQA 1.0 dataset. It provides 658,111 tuples consisting of a question, an image and a set of ten answers given by human annotators. There are approximately 5.3 questions per image, with 123,287 images in total taken from MS-COCO [89]. The official training set comprises 443,757 tuples based on 82,783 images. The validation set offers 214,354 questions about 40,504 images. An official test set is available, albeit without any answers. Therefore, we adopt the same strategy from ImageNet, where we use the validation set as test set, and take a portion of the training set for validation.

3 Unifying Concepts in XAI

> *Users do not care about what is inside the box, as long as the box does what they need done.*
>
> **— Jef Raskin**

As discussed earlier in Chapter 1, the field of XAI has grown rapidly, and in response to growing demands for using Deep Learning on more and more scenarios. In particular, high-stake use-cases such as bank loans [21], access to medical treatments [123] or vehicular security [68] require automatic decisions to be backed up by some kind of human-understandable reason. However, despite continuous efforts to fulfill these demands, proposed solutions often end up focusing on entirely different problems. How can this be happening? At its core, the confusion emerges from a lack of agreement for fundamental terms like "explanation" or "interpretation".

When surveying the corresponding literature, we are quickly confronted with multiple definitions for these basic terms. Have a look at some definitions used throughout time in Table 3.1. For some researchers an "explanation" refers to a model that makes (linear) approximations of the decisions of the original model [129, 96]. For others, it represents either a method through which causal responsibility is assigned [101] or a function that maps opaque objects into "interpretable" ones [33]. This last example is of particular interest because it highlights a second problem with some definitions. Quite often, "explainability" is defined in terms of some other notion, in this case "interpretability". However, the second term is frequently left undefined, or worse, defined via a third concept that in turn lacks a precise definition.

This problem is not new and, thanks to Miller *et al*. [102] it even has its own name: "the inmates running the asylum." In essence, everyone is trying to address a different aspect in the broad field of XAI. And even though the contributions are legitimate and unequivocally valuable, the overall goal of providing standard mechanisms to justify an automatic decision remains unreachable. Without a unified view of the problem, methods that address different issues cannot be compared [15, 90]. This is especially

Tab. 3.1.: Definitions of "explanation" and "interpretation" found in XAI literature.

Source	Explanation	Interpretation
[84]	"someone who is in possession of some information about the causal history of some event (...) tries to convey it to someone else."	—
[66]	"assignment of causal responsibility"	—
[92]	"central to our sense of understanding and the currency in which we exchange beliefs. Explanations often support the broader function of guiding reasoning."	—
[16]	—	"the degree to which an observer can understand the cause of a decision"
[96]	"interpretable approximation of the original [complex] model"	—
[104]	"collection of features of the interpretable domain, that have contributed for a given example to produce a decision (*e.g.*, classification or regression)"	"mapping of an abstract concept (*e.g.*, a predicted class) into a domain that the human can make sense of"
[35]	"measures the degree to which a human observer can understand the reasons behind a decision (*e.g.*, a prediction) made by the model"	—
[41]	—	"to explain or to present in understandable terms to a human"
[78]	—	"quantifies how easy it is to understand and reason about the explanation. Depends on the complexity of the explanation"
[151]	"the collection of features of an interpretable domain that contributed to produce a prediction for a given item"	"the capacity to provide or bring out the meaning of an abstract concept"
[126]	"in human–human interaction, explanations have the function to make something clear by giving a detailed description, a reason, or justification"	—
[129]	"local approximation of a complex model [by another model]"	—

problematic for standardization and governance. Imposition of legal requirements for things like safety critical systems, depends on the existence of a well-established standard.

In recent years, the European Union started passing regulations aimed at guarding end users from unintended consequences of automatic decisions. The European General Data Protection Regulation (GDPR) became popular as one of the first reforms that explicitly mention the "right to an explanation" [44]. Despite all efforts that went into drafting GDPR, it was quickly shown that the right to an explanation can be easily bypassed by exploiting the vagueness of its terminology [127]. Even though this is only one example, it is certainly not an isolated phenomenon. The most recent installment of the AI Index

Report found that in 2021 there were fourteen AI-related bills that have been passed into law just in Western Europe, the UK and the USA alone [166].

To highlight the importance of a standard definition, we can look at a young field in ML that had a similar issue in the past. Adversarial attacks [144] studies imperceptible perturbations that, when added to the input, can cause a high-performing model to make arbitrary mistakes. Since its discovery back in 2013, a community formed around this problem, exploring ways to attack and, more importantly, ways to defend against such attacks. The problem for researchers was, that there were no standard, measurable definitions for concepts like "imperceptible" or "adversarial". Moreover, the setup for evaluating such an attack (*i.e.*, access to model internals, assuming knowledge about the training set, etc.) was being re-defined for each new publication. This led to defense strategies that were promptly deemed ineffective, mainly because of a poor definition of the problem conditions [9, 22]. This community has since, pushed for more rigorous definitions of their constraints for evaluating defense and attack methods.

Let us go back to XAI and its multiple definitions for "explanation" and "interpretation". Can't we just pick one from the pool and make everyone stick to that one? Unfortunately, this is not entirely straightforward. In recent years, these definitions have faced strong criticisms arguing that they are too vague to be operational [41], not falsifiable [80], or simply not adequate for high-stake scenarios [122]. Therefore, a new, concrete and actionable basis is needed, allowing different XAI methods to be compared.

In this chapter, we propose that such basis starts with clear definitions of the two core terms "explanation" and "interpretation". Armed with these two concepts, we can build a framework that encompasses all contributions in XAI, facilitating their comparison. We show that this framework meets expected requirements regarding fidelity, grounding and diversity [4], as well as social aspects such as contrast and selection [101].

In the next section we will have a deeper look at different initiatives to conceive sound definitions for XAI. Then, we start building our proposed framework, laying out the common vocabulary and discussing its scope. Finally, we analyze the completeness of the framework and re-interpret examples from the state-of-the-art using our proposed framework.

3.1 Limitations of Existing Definitions

There is no shortage on definitions for "explanation" or "interpretation". However, a fundamental issue with those, is that they are poorly defined, as they tend to appear on

the margin of the actual contributions, in an effort to provide rudimentary context. Is it possible to find a common ground between all of these definitions? To answer this question, we need to have look at some recurring patterns throughout the history of XAI definitions. Said patterns will allow us to consider the most prevalent properties that these terms should convey.

Going back to early work done in the area of philosophy, we see that the notion of "explanation" is always linked to causal information [84, 66]. However, more recent research has shown that non-causal questions can, and should be the target of XAI methods as well [92, 101]. Therefore, an XAI framework should accommodate both causal and non-causal mechanisms.

More recently, research on explanations mostly focused on the creation of models that approximate the decisions of a more complex one [96, 121, 78]. Under this paradigm, the "explanation" was the model that provided the approximations. Despite their popularity, these models do not fulfill critical requirements that are expected from explanations. In particular, these explanations have a distinct lack of faithfulness [122] which, according to Alvarez-Melis and Jakkola, refers to the ratio between the relevance of features that the explanation deems important, and their true relevance [4]. Furthermore, popular approximation methods like LIME [121] or SHAP [96] have shown vulnerability to malicious attacks. Concretely, a target model can be modified in order to cause explanation methods to identify irrelevant features as relevant [133]. As a result, such linear explanations won't be aligned with the underlying feature representation of the target model. Unsurprisingly, the use of non-linear approximations show similar shortcomings as well [40, 47].

Parallel to these two trends, fundamental research in ML has also delved into the nature and purpose of XAI. For most, the lack of precise definitions presents a challenge to the research community [104, 90, 120]. However, they fall into the trap of defining "explanation" and "interpretation" as vacuous collections for other terms like confidence, transparency or trust [120, 35, 41]. For others, it is more important to define the expected requirements (*i.e.*, desiderata) of an explanation [120, 90] or the ways to evaluate if a model is "explainable" or not [90, 41].

Even though all of these contributions add immense value to the field, they accentuate the lack of cohesion and get in the way of making more impactful advances. In the next section we start defining the context and building basic definitions from the ground up, paying special attention to shortcomings from the literature that have been discussed so far.

3.2 Context and Core Definitions

A solid framework must be supported by strong principles in the right context. Even though the name XAI implies that the framework must apply broadly to the field of artificial intelligence, most of the XAI literature has been dealing with Deep Learning almost exclusively. As discussed in Chapter 1, the ability to draw meaning between the model and the task, has gotten more difficult with the advent of data-driven approaches where Deep Learning reigns. It is therefore consequential that the context of XAI, and the XAI framework focus on the opaqueness of these models.

For a more concrete idea of the context, let us start by thinking about how a classification problem is solved using Deep Learning. We begin by defining the problem intuitively: we want to correctly classify all pictures of a finite set of classes, say horses, and fish. A more formal, yet general definition of the problem is to declare a function that maps the domain of images to the co-domain of classes $f : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^k$. From here on, the standard pipeline for supervised classification provides an effective scaffold to generate such a model: defining a parametrized architecture, optimizing the cost function and generating predictions through a normalization operation like the *softmax* function. A high-level diagram of these interactions is shown in Figure 3.1.

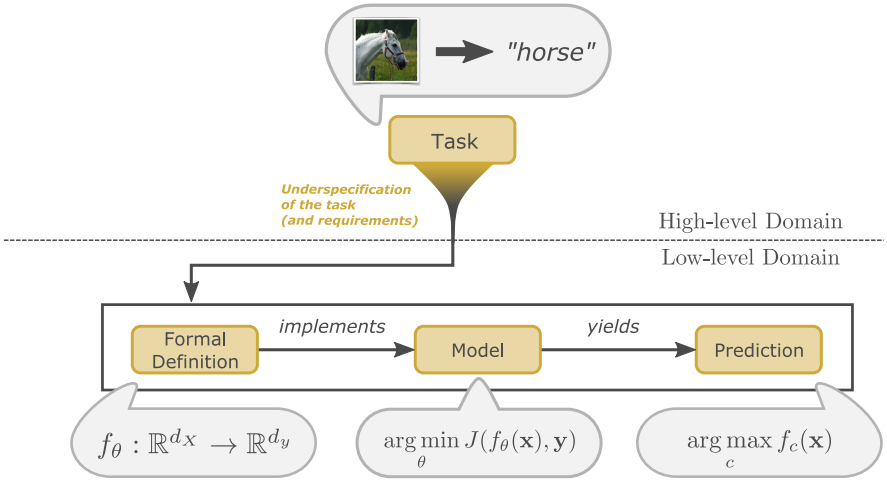


Fig. 3.1.: General overview of an ML pipeline. An intuitive problem gets formalized at the cost of under-specifying its requirements.

With this approach, and under the right conditions, a high-performance model can be easily obtained. However, one problem that goes mostly unnoticed, is the gross simplifications that take place, even before the model gets to train on the first sample. Back in 2006, Lombrozo had already raised a word of caution when working with under-specified problems [92]. More recently, D'Amour, Heller and Moldovan, together

with thirty-seven colleagues at Google and MIT, wrote a more specific analysis on the risks of deploying Deep Learning models trained on under-specified objectives.

Generally speaking, under-specification happens when critical aspects of the problem are not explicitly accounted for. In our classification example, we can reasonably expect that the class prediction should not be affected by the background of the picture. This constraint may appear obvious in hindsight, but the model has not been trained to meet this condition. With the right—or may we say the wrong—dataset, the training process can generate a model that exploits biases from the background. For example, the model could have learned that the color green is strongly indicative of an image of a horse. If this "shortcut" is unknown to us, it would be hard to justify why we can't get a correct prediction from a picture of a fish swimming through algae.

There are many more general assumptions that a problem definition often forgo. Here are three critical aspects that are usually not accounted for, and can lead to under-specification. First, the gap between information contained in a picture of an object, and the object itself a.k.a. the representation gap [134]. Second, the semantic gap or the discrepancy between different semantics that can be drawn from a single sample, when using two different representations [134]. Last, the general interaction between signs, objects and interpreters, as defined in semiotic theory [137].

Here is where under-specification enters the XAI framework. Generally speaking, all these unaccounted constraints define properties that the model either has or lacks. Discrepancies between the expected behavior of the model and the actual outcomes are justified by looking for inconsistencies with respect to said constraints. Thus, we can reinterpret the role of XAI in terms of inconsistencies regarding under-specified constraints. So, as it turns out, methods of XAI do nothing more than probing for evidence that a trained model has learned this kind of constraints! For simplicity, we borrow the term *non-functional requirements* from software engineering to refer to the set of assumed constraints and properties that should be represented, even though they are not explicitly modeled. We will circle back to the critical role of non-functional requirements once we introduce the core terminology for the framework.

Defining Core Concepts

For the XAI framework, we need to define two concepts (explanation and interpretation) without creating ambiguity or relying on additional, hollow terminology. At the same time, we need to convey the properties that have been pushed forward in countless attempts for providing viable definitions. Moreover, these terms must ideally preserve their relation to the original, textbook definitions.

We begin by referring to general English dictionaries and encyclopedic corpora for both terms. As shown in Table 3.2, we can see that our target concepts do not have a consistent definition. In fact, four out of five of these definitions reflect one of the main problems discussed earlier: "interpretation" partially depends on the word "explanation". Does that mean that the two words work more like synonyms? As we will discuss in the remaining of this chapter, this is not necessarily the case. There are significant differences between the two, making it necessary to treat them as two separate notions, albeit closely related ones.

Tab. 3.2.: Definitions of "explanation" and "interpretation" found in dictionaries. Explanations appear as objects while interpretations are associated with an action. Accessed on 2021-01-05.

Source	Explanation	Interpretation
Merriam-Webster	**act** of making plain or understandable	**action to explain** or tell the meaning of
Cambridge	**the details** or other information that someone gives to make something clear or easy to understand	**an explanation** or opinion of what something means
Oxford	**a statement** or account that makes something clear	**the action of explaining** the meaning of something
Dictionary.com	**statement** made to clarify something and make it understandable	**explain**; action to give or provide the meaning of; explicate; elucidate
Princeton	**a statement** that makes something comprehensible by describing the relevant structure or operation or circumstances etc.	**an explanation** of something that is not immediately obvious; a mental representation of the meaning or significance of something
Wikipedia	**a set of statements** usually constructed to describe a set of facts that clarifies the causes, context, and consequences of those facts	A philosophical interpretation is **the assignment of meanings** to various concepts, symbols, or objects under consideration.

Let's go back to Table 3.2. By contrasting the grammatical elements used to describe both terms, we can quickly identify that explanations are defined in terms of nouns, while interpretations are defined using verbs. In that sense, we can already think of explanations as entities that have components or interact with other entities. More concretely, explanations tend to be seen as statements about something else. With this in mind, the next steps towards defining "explanation" are identifying the kind of statements they are, and what they are targeting (*i.e.*, what is the statement about).

Recall that XAI is looking for evidence about the non-functional requirements of a task. This evidence is there regardless of whether we look for it or not. An XAI method is therefore using elements from the task (including the model, data, hyperparameters, etc.) to make statements about a certain non-functional requirement. At this point, we can start to realize that the general purpose of an XAI method already fits the definition we have for "explanation"! Though, to avoid circular arguments or leaving other terms undefined, we can formally model what an "XAI method" is, namely a mapping between existing, factual entities, and the output space of the statement. This idea aligns with Esser *et al.* [43], who define XAI methods as translations between a source and target domains.

Once more, factual entities include all components defined in the pipeline that solves the problem. These can be the model architecture, any hyperparameter used for training, the dataset, intermediate activations, etc. The most important aspect about the input space, is that it comprises objects that are axiomatically true *i.e.*, their existence is unambiguous. Properties of the output space (*e.g.*, values, dimensionality) will depend on the method itself and the kind of evidence it provides. For instance, feature importance methods usually output a heatmap that has the same dimensions as the input space of the model.[1] Perturbation methods like ROAR [62] focus on tracking changes in the accuracy, and hence produce a series of scalar values as their output space. Armed with these new insights, we can already draft a first definition for "explanation":

Definition 3.2.1 (Explanation (draft))**.** An explanation is the process of describing facts related to the non-functional requirements for a task.

Two subtleties to notice: First, we define the main entity as a process in order to convey that, formally, an explanation is just a function. Second, instead of saying that the outputs are "statements", we further identify that those statements are *descriptive* in nature, as opposed to normative ones.

The final aspect regarding explanations, deals with their purpose. Going back to the definitions from Table 3.2, we see that the finality of an explanation is to make something clear, understandable or comprehensible. But understandable to whom? This question makes us evaluate who are (or who *can be*) the consumers of an explanation. Implicitly, these textbook definitions are calling for the involvement of human actors. We can argue that explanations are primarily meant for humans. Notwithstanding, there are functions fitting our current definition of "explanation", whose outputs can be passed on to other machines, but these are considered to be just part of an automatic verification system, and not an explainable system. There are proposals, like those made by Doshi-Velez and Kim, for ways to evaluate explanations without the involvement of humans [41]. However, they also begin by noting that in ML, an "explanation" is a human-centric concept. Both schools of thought seem to have opposing views, but they can indeed be reconciled. An automatic evaluation for explanations—the kind that does not require human involvement—is achievable through metrics that reflect their fitness regarding human satisfaction.

To highlight the link between explanations and humans, we complement our initial definition as follows:

[1] For image models, it is common to preserve only the spatial information, regardless of the number of color channels.

Definition 3.2.2 (Explanation)**.** An explanation is the process of describing facts related to the non-functional requirements for a task, such that it facilitates the understanding of aspects related to said facts (by a human consumer).

To prevent confusion between the process (*i.e.*, the explanation itself), the inputs and outputs of an explanation, as well as the actors that are involved, we introduce the following terminology:

- *Explanation*: process to describe non-functional requirements (see Definition 3.2.2). Sometimes referred to as "explanation method" for additional clarity.
- *Explanandum*: input of the explanation.
- *Explanans*: output of the explanation.
- *Explainer*: actor who proposes the explanation.
- *Explanee*: actor who consumes the explanation.

Let us now look at the word "interpretation". Recall from Table 3.1 and Table 3.2, that this notion was often defined in terms of "explanation". In order to resolve the co-dependency in this definition we can focus on the other recurrent motif in some of the entries: interpretations deal with the *meaning* of something. Looking at the pipeline from Figure 3.1, we see that the output of the system (*e.g.*, the *softmax* of the last layer in a neural network) exists in the formal, low-level domain. However, the final goal is to read back those outputs in terms of the high-level, semantic entities from the original problem definition. For this, a specification needs to exist, allowing users to assign meaning to those formal structures. In the classification example, the output from the *softmax* operation can be read as a probability distribution, where the index of the highest value corresponds to the class prediction. Similarly, for every formal output, a contract needs to exist *assigning* semantic meaning to it.

Note that the Wikipedia entry for "interpretation" in Table 3.2 already talks about an *assignment of meaning*. We adhere to this structure to sketch our definition for "interpretation" in the XAI framework:

Definition 3.2.3 (Interpretation)**.** Act of assigning semantic meaning to formal primitives.

This definition is not exclusively tied to XAI. On the contrary, it is generic enough to apply to the outputs of a classic ML pipeline. Nevertheless, we can exploit this definition, as it is fully suitable to talk about the assignment of meaning for explanations. Here

is where both “explanation” and “interpretation” meet. As part of a formal process, an explanation yields results that live in the domain of mathematical primitives. In particular the explanans can be, as discussed earlier, a tensor or a scalar value. Those outputs alone lack a concrete link to the semantic realm where the non-functional requirements originated. Therefore, an “interpretation” assigns meaning to the explanans so that it can be read and evaluated in the high-level domain.

Having defined the context and core concepts, we arrive at the final version of the XAI framework, shown in Figure 3.2. The task definition originates in the high-level domain and gets characterized in terms of a low-level formal definition using a standard ML pipeline. Limitations with defining a task formally, lead to under-specified models that are exposed to learning shortcuts. Said shortcuts can be exploited to solve a task without relying on patterns that are semantically related to the task, leading to inconsistencies in the prediction process. To validate that the model adheres (or not) to solving the task without relying on shortcuts, non-functional requirements from the task can be probed using explanations. The output of these explanations is interpreted w.r.t. the semantics of the non-functional requirement, and they establish if the model does indeed represent the expected patterns.

We now have a self-containing framework, but does it work to analyze all kinds of explanations? The next section discusses how this framework enables the study of both existing and novel methods in XAI.

3.3 Testing the Universality of the XAI Framework

According to Lombrozo, explanations are a vehicle to contest existing beliefs about an output, by setting constraints to the prediction process [92]. In other words, explanations are methods that provide answers to questions about non-functional requirements. These questions fall into one of three classes: *what*, *how* and *why* inquiries. We also know from the work of Miller, that these three question types are an effective basis for selecting which non-functional requirements can be answered by explanations [101].

To validate that the XAI Framework is suitable to analyze both existing and novel methods, we need to show that all three kinds of questions can be addressed in terms of entities that exist within the framework.

***What* questions:** define the entities that are the subject of an explanation. More concretely, these questions focus on the elements from the non-functional requirements

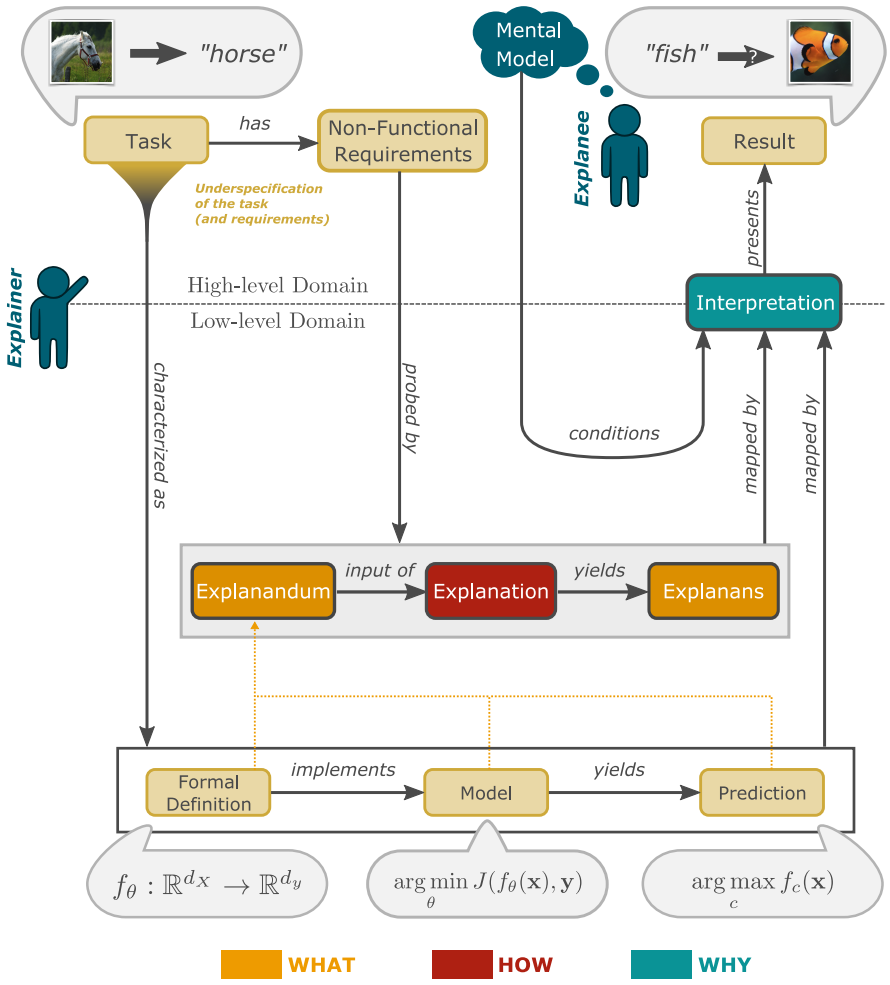


Fig. 3.2.: Overview of our proposed framework. A task defined in the high-level domain gets an under-specified characterization, leaving out non-functional requirements. "Explanations" are methods that probe for said requirements. Interpretations are mappings from the low- to the high-level domain.

that exist in the formal definition, and that are used as input to the explanation itself. It also deals with the structures that are produced by the explanation *i.e.* the explanans. An explanation therefore answers *what* elements from the formal domain are being used as input to the explanation, and *what* structure is the result of the explanation.

***How* questions:** describe the way an explanans is produced. These are the main questions addressed by the XAI literature, and they focus on describing details of the process that transforms an explanandum into an explanans. This is the question that is usually thoroughly discussed in the literature, and it has already sparked all kinds

of valuable discussions. For example, whether black boxes and linear approximations count as valid explanations [122, 133].

***Why* questions:** focus on causal factors of a non-functional requirement. Causal analysis is one of the most powerful mechanisms to validate patterns learned by ML models. With them, we can isolate the specific factors that bring about the behavior of a model. Why questions are primarily addressed by the *interpretation i.e.*, the contract between the formal and the semantic domain. Formal causal methods can sometimes be included in the explanation [93, 26]. In these cases, the resulting explanans has a straightforward link to the semantic domain.

This is how all three questions can be addressed in terms of entities within the proposed framework. *What* and *how* questions are addressed by the explanans, explanandum and the explanation itself. *Why* questions are dealt with by providing causal elements for the interpretation. Note that explanations with causal elements are not the only ones that can be tackled. For instance, showing that a model has issued an unfair prediction is already useful to discard said prediction, even if the cause remains unknown. While *what* and *how* questions must be made explicit, causal interpretations are optional, albeit preferred.

Knowing that the framework is suitable for a comprehensive array of questions, it is important to validate how the expected properties of explanations can be expressed within the framework.

3.4 Desiderata of Explanations within the XAI Framework

In the past, aside from searching for rigorous definitions, research has also tried to establish what the main properties of explanations are, and what kind of metrics can be used to compare them. This section discusses desiderata proposed by Miller [101], and by Alvarez-Melis and Jakkola [4], and show how it can be framed within the proposed XAI framework. Similarly, we will look into the properties for metrics that Doshi-Velez and Kim have put forward [41], and frame them in the context of our framework.

In an effort to understand the principles of intrinsic explainability, Alvarez-Melis and Jakkola define three main characteristics that explanations must have: fidelity, diversity and grounding. In addition, they identify a set of *quantifiable* properties for evaluating explanations, namely explicitness, faithfulness and stability. Let us go through each one

and understand what these are, and how each characteristic is addressed in the XAI framework.

Fidelity: dictates that an explanation must preserve relevant information. The source of that information lies on both the problem and the model that is being explained, but also on the non-functional requirements that are being considered. The XAI framework enforces the preservation of relevant information by constraining the domain of explanans to formal primitives that are factual and axiomatically true.

Diversity: indicates that an explanation must convey its meaning using a set of small, non-overlapping semantic concepts. By grouping the misalignment between expected and actual outputs with individual non-functional requirements, the framework promotes explanations that focus on subsets of the ML pipeline. A guarantee that the concepts do not overlap semantically, although not provided, is not hindered by elements in the framework.

Grounding: states that explanations should be expressed in terms that a human can understand. This notion is directly addressed in the definition of "interpretation" when we identified that the assignment of meaning is intended for human consumers.

Explicitness: measures how understandable the provided explanations are. This property refers to the number of assumptions that need to be made for explanans to be mapped back to the semantic domain. More generally, an explanation is more explicit if the interpretation follows straightforward from the structure of the explanans. The prime example of high explicitness is to rely explanations based on causal theory. As long as the causal graph is sound, the interpretation can be based on the semantic meaning of the nodes in the graph.

Faithfulness: evaluates the true relevance of the features being selected for, or by the explanation. There are two stages in the framework where faithfulness can be evaluated. First, the relevance of the features can be measured with respect to the non-functional requirement. In this case, the selection is being done by the explainer; she is in charge of identifying features of the non-functional requirement that will come under scrutiny. The explainer is also responsible for finding the corresponding features among existing elements in the formal domain. Second, we can talk about the relevance of features of the explanans, and their interpretation. At this stage, both the explainer and the

explanee come into play: while the explainer needs to provide an interpretation, the relevance is measured with the help of explanees.

Stability: estimates the consistency of an explanation when processing similar inputs. Of course, the measurement must be done on the explanandum, and not on the explanation itself. As this comparison takes place on the formal domain, any measure of distance can be used.

Another set of desiderata has been proposed by Miller, who urges the ML community to consider findings coming from philosophy and the social sciences, as they have been investigating the properties of effective explanations for much longer [101]. Miller concludes that explanations for XAI must exhibit these four fundamental aspects:

Contrastive aspects: Explanations must justify why a decision was taken *instead* of another one. For the XAI framework, this contrast exists between the desired and the actual prediction. When a mismatch takes place, it must be found by analyzing which non-functional requirement is not being represented by the model.

Selective aspects: Explanations should focus on only the most important factors that justify the output. Once more, constraining explanations to a non-functional requirement forces the result to be selective.

Irrelevance of probabilities: It states that the best explanation for the average case is not always the best explanation. This aspect is particularly important for local methods where explanations have to deal with individual samples or single neurons. One must be careful not to attribute failed predictions from two different samples to the same non-functional requirement.

Social aspects: It points to the primary goal of explanations, namely that they are meant to be understood by humans. This aspect is related to the notion of "grounding" from Alvarez-Melis and Jakkola's work. Again, this aspect is covered by the definition of "interpretation" that we provide as part of the framework.

In addition to desiderata, Doshi-Velez and Kim have looked beyond the explanations themselves, and discuss three strategies to measure explanations [41]. They argue that explanans can be measured with automatic methods, but also through human interaction. For fully-automatic evaluations Doshi-Velez and Kim suggest using a formal definition of

interpretability as proxy task to evaluate the quality of explanations. However, a formal definition is not trivial according to the authors themselves. Moreover, proxy tasks come with their own list of caveats, as they end up falling into the contested scenario where a simpler model explains the more complex one. In any case, all measurements of this type will inevitably live in the formal domain and hence, require an additional (formal) interpretation.

The remaining two options for measuring explanations include the involvement of human actors who weigh in on either the original problem, or a simplified version of the task. Either way, this kind of evaluation will be heavily dependent on the interpretation that is given to evaluators, rather than on the explanation itself. Only when human evaluators are well-versed in how the explanation works and how it relates to the non-functional requirement, their judgment can include aspects from the formal domain.

We see how our framework offers a universal foundation to describe, and even to enforce desiderata of explanations. Despite there being a long list of terms to describe different aspects of explanations, we have discovered that many of them are actually referring to the same properties. We show that "grounding" and "social aspects" are referring to the involvement of humans. The commonality became evident once we described both definitions in terms of our proposed framework. This review is encouraging as it showcases the benefits of having a universal and well-defined framework to talk about XAI.

Now that we know how to use the framework, what is the best way to start using it? In the last section in this chapter, we consolidate the most important recommendations to expedite progress in the field of XAI by using our framework.

3.5 Recommendations Moving Forward

With new contributions being made to XAI on a regular basis, we can exploit the XAI framework to assure that novel methods are being compared and evaluated adequately.

The most common issue with new methods is that they focus all their effort to describe the explanation itself. The downside being that the other elements (*i.e.*, explanans, explanandum, non-functional requirement, interpretation) don't get analyzed in depth. In particular, novel methods rarely discuss the link between the non-functional requirements of a task, and the explanation itself. Similarly, most publications neglect the importance of providing a precise interpretation, or motivating the selection of the explanandum.

We argue that new contributions should address the full scope of the explanation, including the interaction between the semantic and the formal domain. Efforts to define standards for governance and high-stake scenarios will be more likely to succeed once all methods can be compared. Having contributions that cover all elements around explanations leads to more fruitful discussions, with modules that can be measured using the same standards. An exception to this recommendation is when research is being deliberately addressing one part of the pipeline. For example, when addressing the limitations of an existing method w.r.t. their interpretation, explanans, etc.

Fortunately, there are contributions in the field that do pay attention to all elements in the XAI framework as shown in the following example.

Case Study: Class Activation Latent Mapping

To illustrate how an XAI method can adhere to the XAI framework, we have a look at the recent work by Kim, Choe *et al*. where they propose a feature attribution method based on Class Activation Maps (CAM) [72].

In their paper, they observed that explanans produced by CAM violate several axioms for explanations such as implementation-invariance, sensitivity or completeness [141]. These axioms ultimately relate to non-functional requirements that CAM fails to preserve, even though they are present in the target model.

Next, the authors propose CALM: an explanation that produces feature attribution maps *i.e*., a measure of how relevant each input feature (each pixel) is when making a prediction. Being inspired on CAM, CALM relies on the same explanandum and assumptions about the model (*e.g*., that the network is fully convolutional). The explanation on the other hand, operates vastly different, and comprises a probabilistic representation, together with a latent model of the location of semantically relevant features. For the explanans, CALM produces a well-defined joint distribution between the predicted locations for semantic features and the class probabilities. The authors analyze advantages of having such an explanans and contrast it to the one CAM produces.

Finally, the paper discusses the different interpretations that CAM and CALM have. They start by noting that, originally, CAM did not provide an interpretation. Therefore, aside from an interpretation for CALM, the authors try, to the best of their abilities, to provide one for CAM as well. Armed with both interpretations, they proceed to identify fundamental issues with the interpretation of CAM, and contrast it to that of CALM.

Without going into all the details of the paper, we see that Kim, Choe *et al*. have proposed a new explanation that addresses all parts of the XAI framework. Despite using a slightly

different language than the one we propose, the paper discusses (and motivates) their findings in light of non-functional requirements and interpretations. We see this kind of research as an encouraging example that others in the field can strive to follow. However, we argue that using a standard language can help in communicating more effectively the extent of their contribution.

In Summary

The field of XAI suffers from "the inmates running the asylum," and it is hindering progress in establishing standards for high-stake scenarios and governance. We propose an XAI framework that allows for existing and new contributions in XAI to be measured and compared. The framework is based on grounded and precise definitions for "explanation" and "interpretation" complementing the classic ML pipeline. We discuss how the framework is compatible with desiderata of explanations and provide an example of how future research can describe their contributions more effectively by addressing all stages of the XAI framework.

In the next chapter, we move to a more specific question regarding biases that arise from training on datasets with overrepresented classes that are semantically related.

4

P ≈ NP, at least in Visual Question Answering

> *Language serves not only to express thoughts, but to make possible thoughts which could not exist without it.*
>
> **— Bertrand Russell**

Neural networks are lazy. But not because they run slowly or because some neurons refuse to run. They are lazy because they will try to exploit the simplest patterns they can find. From an optimization perspective, this is of course efficient, but it is also often problematic. Regardless of how semantically irrelevant some features are, a model will rely on them as long as they show up consistently during training.

For large datasets—the kind you need for training deep networks—it is impossible to track unintended biases that can end up in the training set. Part of the assumptions of the ML pipeline is that the training set is a representative sample of the much larger (often infinitely so) input space. Unfortunately, the sampling process is imperfect for very complex tasks, resulting in datasets that are correctly annotated, but that also contain patterns that don't belong to the task.

This is especially true in computer vision where natural images are scraped from the internet and evaluated by human annotators within a narrow scope. For example, annotators of the ImageNet dataset were instructed to assess the presence of certain target classes, regardless of the number of objects or surrounding clutter [124]. The motivation to ignore these factors is to promote diversity in the sampling process. However, this opens up the possibility for unforeseen biases to leak into the final dataset. In the end we can't know for sure if one class of ImageNet can be correctly identified because it always contains the same number of objects, but also because of any other unintended pattern that is consistent across samples of that one class.

The problem for ML models arise when the spurious correlations end up being easier to represent than the features of the target class. Borrowing again from ImageNet, we find that most images for the class SIBERIAN HUSKY show the animal in front of a snowy background. Thus, the brightness of the background becomes a feature that an ML model can exploit to predict SIBERIAN HUSKY. A similar issue has been identified on the image captioning dataset MS-COCO [89], which is known for having biases due to the co-occurrence of objects and backgrounds (*e.g.*, giraffes shown in front of grass [168]).

While these examples may seem harmless, critical situations arise when biases are linked to societal standards for fairness. For example, a quantitative study by Stock and Cisse concluded that image classifiers are exploiting biases related to racial stereotypes [138]. Zhao *et al*. also showed that models for video classification learn to reproduce and amplify gender biases that are present in the imSitu dataset [169].

So far, it seems like the unintended biases that should be avoided heavily depend on the application or the dataset. However, there is one common condition that can introduce adverse biases into any kind of ML model. The performance of a model depends, in part, on how well-represented (sampling-wise) each class is within the dataset. With no prior information about each class, most common ML benchmarks rely on a uniform sampling to form the dataset. Some of these datasets were described in Chapter 2, and we see that they, in fact, adhere to a uniform sampling *i.e.*, the number of images per target class remains constant.

The motivation to stick to a uniform label distribution is straightforward. Imagine training a neural network to classify horses and fish. Now, suppose that your training data consists on one million horse images, but only ten samples showing a fish. Under normal conditions, a model that trains using SGD on random mini-batches will quickly learn to predict the class HORSE every time. From an optimization perspective, the model can quickly minimize the loss function by making minimal changes to its weights. Statistically, most mini-batches will have only horses, while fish will occur only once for every one-hundred thousand images. By only predicting the class HORSE this model will have an accuracy of over 99.9%. Even though the conditions of this example are extreme, it illustrates the impact of using non-uniform sampling for classification problems.

If most modern datasets use uniform sampling, is there any reason to worry about this issue? There are indeed several. The obvious incentive for studying this limitation is that we cannot guarantee that future datasets will be compiled in the same way. Sometimes, practical limitations will make it impossible to get uniformly sampled data. For instance, data distributions with very long tails have been a major challenge for training autonomous vehicles using ML. As this example suggests, another motivation is that currently, some disciplines simply don't have any uniformly sampled benchmarks.

Staying within the realm of computer vision, the fascinating field of visual question-answering (VQA) has suffered multiple setbacks that are ultimately linked to the data. In this chapter we will study the effects of sampling biases for VQA, and evaluate the impact that these have when training high-performance models. In particular, we will focus on the confounding effects of including semantically related questions that have different labels.

4.1 The Problem with VQA Datasets

A few years after the AlexNet network set a new record for image classification, Antol *et al*. showed that similar models could be used to answer more specific questions about images [7]. As one of the firsts to envision the problem, they called it Visual Question-Answering, or VQA for short. In general, a VQA model works by predicting the answer to an arbitrary question about an image. The twist: questions can be written in plain, unstructured text, just as humans would. Models that predict answers for VQA have to solve a complex objective that involves multiple data modalities (text and images), while jointly representing patterns from natural language processing (NLP) and object detection. For more details about the task, refer back to Section 2.4.

Despite the added complexity, advances in VQA showed that deep neural networks were powerful enough to produce results that were not far from human performance [131, 161].[1] As the field grew, so did the efforts to compile a comprehensive dataset that could be used as gold standard. Two years after VQA was proposed, Kafle *et al*. reported an evaluation on six large datasets for this task [67]. In the same paper, Kafle also showed that at least five of those datasets suffered from some form of sampling bias.

Among the scrutinized datasets is one of the original, and most widely used sets at the time—the one known as VQA 1.0. It has been reported that using the questions alone (*i.e*., without the reference image) was already enough to attain non-trivial performance [53, 67]. For example, always giving the answer TENNIS to all questions starting with "*What sport...*" results in an accuracy of well over 30% for sport-related samples. Kafle *et al*. also noted that human annotators tend to ask questions about objects that appear in the image. Hence, it is not surprising that questions starting with "*Do you see a...*" can be answered affirmatively 79% of the time!

Motivated by these shortcomings, Goyal *et al*. proposed a supplement for VQA 1.0 in an effort to balance these exact same issues.[2] Here is how they did it: Given a tuple

[1] For VQA 2.0, human performance is 85.01% while MCAN reportedly scored 70.9%.
[2] They estimated that affirmative questions starting with "*Do you see a...*" were as high as 87%.

(Q, I, A) corresponding to a question, an image and an answer; they find a second image I' for which the original question Q results in a different answer $\bar{A} \neq A$. Thus, the newly created tuple $(Q, I', \bar{A})$ compensates biases in the question by having at least two associated answers. A model that tries to solve the VQA task cannot rely on text alone anymore, but it will be forced to condition the answer on the visual features. They called the resulting, more balanced dataset VQA 2.0 [53].

VQA 2.0 soared in popularity, becoming one of the most competitive benchmarks with its own dedicated workshop at CVPR between 2017 and 2021.[3] However, the novel dataset was still facing criticisms. Because it was based on VQA 1.0, critics like Kafle questioned the limitations of having coarse categories with lots of samples [67]. For example, he noted that questions about colors are much more frequent than those asking about spatial relations. Therefore, a model that specializes in color—a feature that is also easy to represent—will score higher than a more complex one that excels in spatial reasoning.

Following Kafle's argument, this chapter looks into another class disparity that, to our surprise, has not been studied in depth despite comprising roughly 38% of the whole dataset.

There are Too Many Polar Questions

Among the set of possible answers that VQA models can give (recall from Section 2.4 that answers are represented as independent labels, and models follow a similar structure to image classifiers) there are two labels that have by far the most samples per class. We are talking about questions that can be answered with either "*yes*" or "*no*". It is easy to argue why these two labels define a category of their own, as questions that ask for a "*yes*" or a "*no*" follow identical structures, and the answers are complementary to each other (*e.g.* an object is present or not, something has a certain color or not, there are exactly k objects in the image or not). We refer to questions that can be answered this way as "*polar questions*". Similarly, we use the term "*polar answers*" to refer to the label set $\{\text{YES}, \text{NO}\}$. In contrast, everything that is neither a polar question nor a polar answer is referred to as *non-polar question* and *non-polar answer* respectively. For simplicity, we also use the notation P and NP to refer to polar and non-polar elements.

Figure 4.1 shows an overview of the class distribution of VQA 2.0 regarding polar and non-polar samples. We see how, among the 3129 classes that comprise the dataset, polar questions make up 38% of the dataset, or 19% per class. In contrast, the remaining non-polar classes have just over 0.02% of samples per class, in average.

[3] https://visualqa.org/workshop.html. Accessed on 2022-04-01.

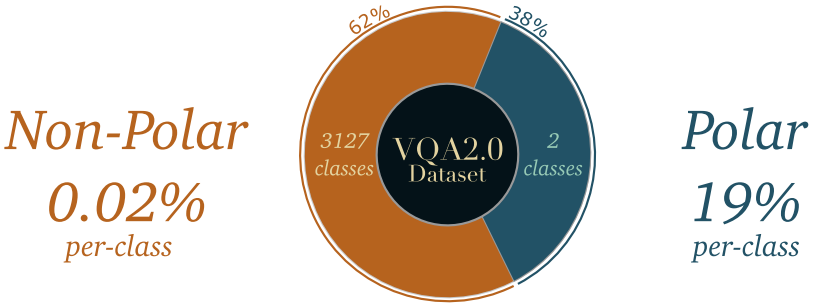


Fig. 4.1.: Distribution of polar (P) and non-polar (NP) samples in VQA 2.0.

Critics of polar questions argue that they are too simple, and therefore, they have been excluded from other datasets like Visual Genome [74] or Visual7W [172]. In stark contrast, datasets like SHAPES [6], abstract VQA [168] or NVLR2 [140] have opted for gathering data that focus exclusively on polar questions.

For datasets with both polar and non-polar samples, there are usually a notable, higher number of polar questions than non-polar ones. Polar questions are easier to formulate and verify in general (unless they are being purposely used to express complex boolean expressions [139]). The research community has come up with different strategies to cope with the disparity between polar and non-polar samples. Agrawal *et al*. propose using alternative splits for VQA 1.0 and VQA 2.0 tailored towards measuring the ability of a model for representing unknown composite concepts [3, 2]. For instance, a model that is trained on samples about "*green plates*" and "*white shirts*" is later evaluated on questions related to "*white plates*" and "*green shirts*". Moreover, regularization methods have been proven effective for mitigating the presence of biases that is being picked up by language models [119, 2].

In addition to these ideas, reporting the accuracy for the polar questions separately has now become a standard practice. However, this separation of evaluation metrics is only done at test time. For training, models usually optimize on the entire dataset[4], and rely on uniform random shuffling.

As discussed at the beginning of the chapter, this training regime can be introducing unintended biases. Methods such as mini-batch re-sampling [88] can be used to counteract the effects of overrepresented polar samples. But even then, it is still not guaranteed that models don't end up specializing in polar questions simply because they appear more often. In fact, looking at results from the VQA challenge[5], we can verify that polar questions are correctly classified more often than their non-polar counterparts.

[4] One notable exception is the VGQA model by Agrawal *et al*. which consists of an architecture that process polar questions separately [2].

[5] `http://visualqa.org`

Can't we just remove polar questions until they match the number of samples per class of the non-polar set? Shouldn't that be enough to eliminate any potential biases? At first sight, it seems like a straightforward idea. However, there is another particularity about polar questions that makes the answer more nuanced. Have a look at the example in Figure 4.2. When asking a polar question like "*Is there a white animal?*", the model needs to represent features related to animals, and to the label WHITE to answer correctly. Similarly, the non-polar question "*What color is the horse?*" requires that the model represents the animal, and the color in question. In this case, we say that the polar question relates to (or talks about) a non-polar concept.

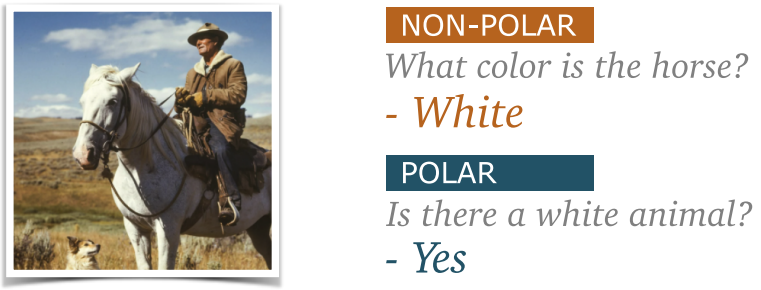


Fig. 4.2.: Example of a polar (P) and a non-polar (NP) question. Sometimes, polar questions convey information about a non-polar class. Here for example, both samples refer to the color WHITE.

This last observation suggests that polar and non-polar samples are, in some cases, strongly linked in the semantic space. With this idea in mind, we define the two main questions in this chapter: Can we measure the impact of training with overrepresented polar samples? Are the features of polar-questions interfering with those needed by non-polar samples or are they complementary?

In the remaining of this chapter we define a series of experiments aimed at answering these questions. The next section describes the experimental setup, which involves a high-performance VQA classifier and the VQA 2.0 dataset. Next, we delve into an ablation analysis of the training conditions to compare the effects of using polar and non-polar samples on the feature space of the model. We then focus on the relationship between polar questions that relate to non-polar concepts. Finally, we gather all the results, and rule on whether the interaction of polar and non-polar samples is detrimental or beneficial.

4.2 Experimental Setup

In this section we describe the core components and notation for all experiments. First, we motivate the selection of our baseline VQA classifier. Then, we discuss different constraints that will be introduced as part of the ablation analysis, allowing us to quantify potential biases coming from overrepresented polar questions.

4.2.1 The Model

To evaluate biases in VQA 2.0 we first need to pick a model that serves as a reference for all experiments. Ideally, we need to strike a balance between simplicity and performance. Models that rely on external data or pre-trained components add constrains to the setup that are orthogonal to the problem we want to focus on. On the other hand, choosing a model that is too simple could cap the diversity of feature representations that are needed to solve the VQA task. For this reason we choose the model proposed by Anderson *et al.* [5]. As the winner of the VQA challenge in 2017, the architecture consists on a few simple modules that deliver a consistent high performance.[6]

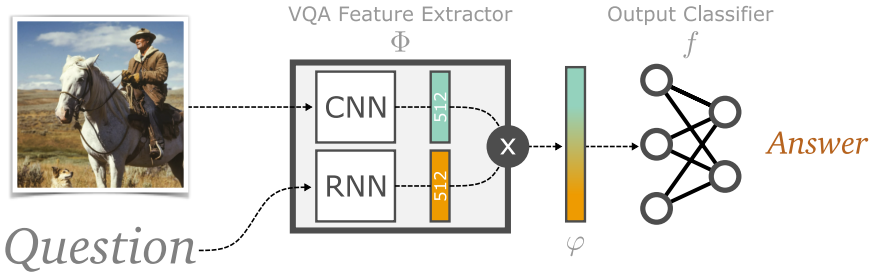


Fig. 4.3.: Overview of the chosen VQA model. It consists of three modules: a CNN and an RNN that produce a joint textual-visual embedding (VQA feature extractor or Φ), followed by a shallow 2-layer output classifier (denoted as f).

An overview of the model in question is shown in Figure 4.3. There, we see that the model consists of three main components. The first two are a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN). They are responsible for processing the image and text information independently, projecting their features into their own lower-dimensional space. The embeddings that result from each of these components are multiplied element-wise with each other, forming a joint embedding of the question and the image. We denote this joint embedding as φ. This embedding is

[6] Even though the model came out in 2017, performance of other architectures has improved by only 7.6p.p. in the following three years, according to Papers with Code (URL: `https://paperswithcode.com/sota/visual-question-answering-on-vqa-v2-test-std`. Accessed on 2022-04-01).

then passed on to the last component of the VQA architecture: the output classifier. The architecture of the output classifier comprises two fully-connected layers with a softmax operation at the end, resulting in an output vector of size $N = 3129$ when the entire VQA 2.0 dataset is used.

To simplify the nomenclature when describing the upcoming experiments, we introduce the following notation. The first part of the model *i.e.*, up to the point where the joint embedding φ is produced, is referred to as the VQA feature extractor, and is represented by the Greek letter Φ. For referring to the remaining part of the model *i.e.*, the one that produces the output based on φ, we assign the letter f. Under this notation, a prediction $\hat{A}$ can be expressed as $\hat{A} = f(\varphi)$ where $\varphi = \Phi(Q, I)$ and the tuple (Q, I) corresponds to the question and the image of a VQA query.

When a module has been trained on only polar samples, we will use the sub-index P for that module. This way, a VQA model that has been trained only on polar samples is denoted as $f_{\mathrm{P}} \circ \Phi_{\mathrm{P}}$. Similarly, a model trained on just non-polar samples will have its modules denoted with the sub-index NP. The sub-index Ω will be used to emphasize that a module has been trained on both polar, and non-polar samples.

4.2.2 Overrepresentation Biases

The purpose of our experiments is to elucidate two main questions. The first one asks if the overrepresentation of polar samples causes a model to prioritize them, sacrificing performance on the non-polar ones. If this were the case, a model that only had to train on non-polar queries, would show higher performance than a model trained on both polar and non-polar samples. Thus, our first experiment examines precisely these two scenarios and compares the accuracy of the model $f \circ \Phi$ when trained on:

- The entire VQA 2.0 dataset (polar and non-polar) $\rightarrow f_{\Omega} \circ \Phi_{\Omega}$
- Only polar samples $\rightarrow f_{\mathrm{P}} \circ \Phi_{\mathrm{P}}$
- Only non-polar samples $\rightarrow f_{\mathrm{NP}} \circ \Phi_{\mathrm{NP}}$

4.2.3 Interfering Feature Representations

For the second question, we want to establish if non-polar features worsen when they are represented in the same space that polar samples occupy. To probe this feature space, we propose the following setup. We start by training a model using only polar-samples *i.e.*, $f_{\mathrm{P}} \circ \Phi_{\mathrm{P}}$. Then, the resulting feature extractor Φ_{P} is used to extract features from

non-polar samples that, in turn, serve to train a non-polar classifier f_{NP}. The resulting hybrid model $f_{\text{P}} \circ \Phi_{\text{NP}}$ represents the alignment between the polar and non-polar spaces. Evaluating the accuracy of this hybrid model will tell us whether a polar feature space can be used to answer non-polar questions.

Finally, as discussed in the last part of Section 4.1, there are semantic relations between polar and non-polar samples when the former asks about the presence of a non-polar class. To study the impact of this semantic relation, we need to find a subset of polar questions that have a semantic link to non-polar classes. Considering that VQA 2.0 provides over 650 thousand samples, a manual search for semantic links is unfeasible. Instead, we define an automatic criterion that finds potential matches via syntactic analysis.

For this, we gather all words from each non-polar label, and use those to find exact matches within the list of all polar questions. For example, the non-polar class HORSE will match to the polar sample with the question "*Is there a white horse?*". This process is repeated for all non-polar classes, and each question can be matched to any number of non-polar classes. Finally, non-polar classes get sorted according to the number of matching polar questions. Non-polar classes that had the most matches are considered to be *well-covered* by polar questions. This expression simply means that there are many polar questions that talk, in one way or another, about the matched non-polar class. A toy example of the matching and sorting method is shown in Figure 4.4.

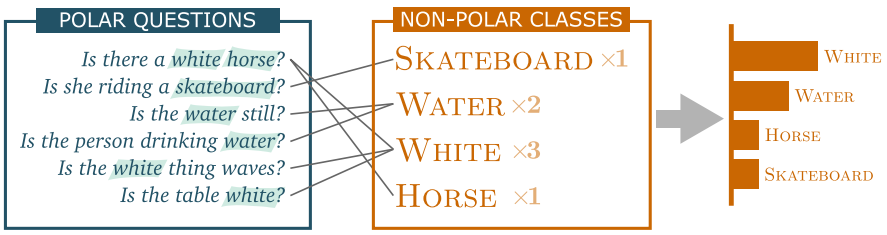


Fig. 4.4.: Matching non-polar classes to polar questions. At the end, non-polar classes get sorted according to the number of matching polar questions.

The sorted list of well-covered non-polar classes can be used to evaluate the hybrid model $f_{\text{NP}} \circ \Phi_{\text{P}}$. Here is the rationale behind this experiment. Recall that the polar feature extractor Φ_{P} has been already trained on all available polar samples. Within that polar training set, concepts that relate to the non-polar classes are going to be conveyed to the feature extractor. With enough polar samples mentioning a specific non-polar concept, we can establish if non-polar concepts that are well-covered are also better represented in the polar space of Φ_{P}. If this is the case, the accuracy of the non-polar classifier f_{NP} that uses features from Φ_{P} should be higher for those non-polar classes that are well-covered.

4.3 Implementation and Results

We begin by establishing the baseline performance of the reference model. Similar to the original evaluation by Anderson *et al.* [5], we select samples from VQA 2.0 whose answers occur at least eight times. This leaves us with 3129 classes, including the polar labels YES and NO, resulting in the data distribution discussed earlier, and summarized in Figure 4.1. In order to minimize the influence of orthogonal aspects in our experiments, we choose not to pre-train on the Visual Genome dataset like Anderson *et al.* did.

For the CNN, a hyperparameter k plays an important role in determining the number of sub-regions to use for computing the image embedding. Following the recommendations from Teney *et al.* [148], we set $k = 36$. Additionally, we use ReLUs instead of GatedTanh as non-linear operations, because they don't suffer from vanishing gradients, are more computationally efficient, and have a negligible impact in performance.

The model is optimized using Adamax [73] with an initial learning rate of 2×10^{-3} on the full training set. After convergence, the standard VQA accuracy is reported on the validation set, as done by Anton *et al.* [7]. Moreover, depending on the experiment, we report the VQA accuracy w.r.t. polar questions, non-polar questions or all questions in the validation set. These partitions are denoted with P, NP, and Ω respectively.

Tab. 4.1.: Experimental results. The first column corresponds to the trained model that results from training on all (Ω), polar (P), or non-polar (NP) samples. The three following columns report single-model VQA accuracy when the model is evaluated on all, polar or non-polar samples in the validation set.

Model	Accuracy (%)			Description
	Ω	P	NP	
$f_\Omega \circ \Phi_\Omega$	62.4	80.4	51.4	Baseline
$f_\mathrm{P} \circ \Phi_\mathrm{P}$	—	79.6	—	Only polar
$f_\mathrm{NP} \circ \Phi_\mathrm{NP}$	—	—	51.6	Only non-polar
$f_\mathrm{NP} \circ \Phi_\mathrm{P}$	—	—	28.7	Hybrid model

Going back to the first question about the influence of overrepresented polar samples, Table 4.1 already gives us some interesting insights. The results of the first two rows indicate that the performance of a model trained with polar samples is almost identical to the model that trains on the entire VQA 2.0. A similar result is observed when the model is trained only on non-polar samples (first and third rows).

In general, the accuracy of the model is unaffected by the overrepresentation of polar samples, even when training with the entire dataset. This result strongly indicates that

training on a disproportionate high number of polar questions, does not trivially cause the model to optimize for them. There is however, a small increment w.r.t. the baseline in the accuracy of non-polar samples when the model is trained only on those ($+0.2p.p.$). On the other hand, the baseline has done a better job at answering polar questions *i.e.*, when training on the entire dataset. This suggests that, if anything, the non-polar samples have complemented the polar feature space when the model is trained on polar and non-polar samples. However, these effects are small, and the overall outcome proves that overrepresentation of polar questions in VQA 2.0 is not hindering performance.

In fact, we can quickly verify that the weighted average of the two exclusive models (rows 2 and 3) would result in a lower accuracy than the baseline itself. We can think of these two models as an ensemble. To estimate the accuracy of said ensemble, we need to assume that the right model is used according to the polarity of the question *i.e.*, all polar questions are evaluated by $f_{\mathrm{P}} \circ \Phi_{\mathrm{P}}$ and the non-polar samples are processed by $f_{\mathrm{NP}} \circ \Phi_{\mathrm{NP}}$. Under this condition, the total accuracy of the ensemble can be estimated as $79.6\%(\frac{11}{29}) + 51.6\%(1 - \frac{11}{29}) \approx 62.22\%$, where $\frac{11}{29}$ is the ratio between polar and non-polar samples. This places the ensemble $0.2p.p.$ below the accuracy reported for the baseline. Note that the assumption we make renders this scenario unrealistic, but it proves that the baseline would be a better option than the theoretical ensemble. This result also confirms that the baseline has not been negatively affected by training on the entire dataset, albeit the disproportionate number of polar samples.

Delving into the second question, we have just established that the polar and non-polar features don't exert a strong negative influence on each other when represented within the same space. This leaves us with two plausible alternatives. The first one is that polar and non-polar features don't overlap, and each one simply occupies a different, disjoint area on the feature space. The second option is that the features do overlap, and the overlap is complementary. In other words, training on polar questions helps the model represent features that are needed to answer non-polar questions, and vice versa.

Based on the first experiments (rows 1 to 3 in Table 4.1), it is not possible to determine if the features are indeed complementary or not. The difference between the accuracy of the baseline model and the two versions that train exclusively on polar and non-polar samples is small. Moreover, the training conditions for each model are invariably different due to the total number of samples available for training.

This is where the last experimental setup comes into play. We use a polar feature extractor Φ_{P} on the set of non-polar samples. The result of training and evaluating a non-polar classifier f_{NP} on all features from Φ_{P} is reported in the last row of Table 4.1. We observe that even though the accuracy of the hybrid model is nowhere near that of

the baseline ($-22.7 p.p.$), the 28.7% true positive rate lies substantially above random chance.

At this point, we suspect that the semantic link between polar questions and non-polar classes may be playing a role. To verify if this is the case, we sort non-polar classes by the number of matching polar questions, as described in Section 4.2.3. The hybrid model is evaluated once more, only this time we compute the accuracy over subsets of non-polar samples that are well-covered by polar questions.

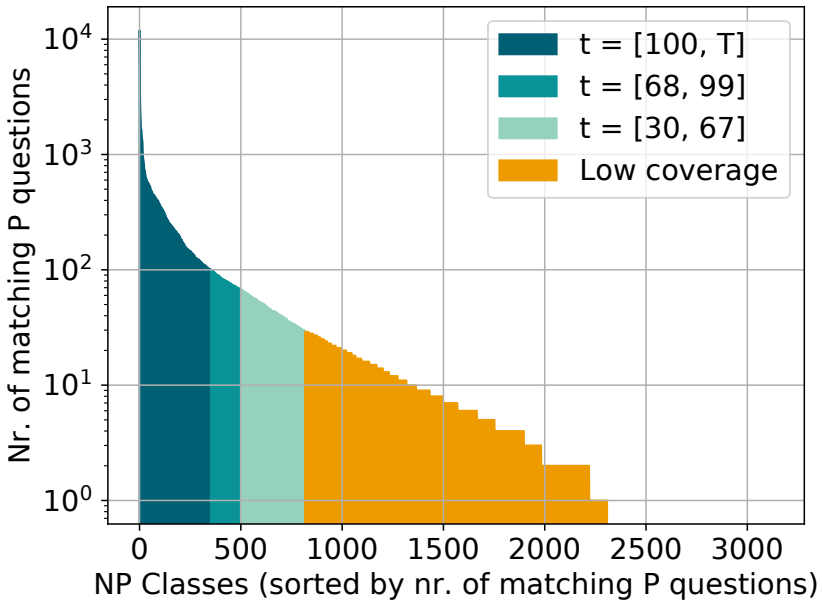


Fig. 4.5.: Non-polar classes after being sorted by the number of matching polar questions.

Figure 4.5 shows the histogram of non-polar classes after being sorted according to the number of matching polar questions. As we can see, well over 70% of non-polar classes have matched to at least one polar question. For this evaluation, we need to focus on non-polar concepts that are *well-covered* by polar questions. Thus, we discard non-polar classes that have less than thirty matching polar questions (shown in yellow). For the remaining non-polar classes, we define three brackets that reflect how well-covered they are: The first bracket [■] contains non-polar classes with 100 or more matching polar questions. The second bracket [■] considers the top 500 well-covered non-polar classes that are not in the first bracket. The third bracket [■] corresponds to non-polar classes that have at least 30 matching polar questions, but are not in any of the first two brackets. We refer to the these brackets by the interval of matching polar questions that each one defines. This way, the first, second and third bracket are denoted by the intervals $[100, T]$, $[68, 99]$, and $[30, 67]$ respectively, where T refers to the maximum number of matches.

Tab. 4.2.: Accuracy of the hybrid model $f_{\mathrm{NP}} \circ \Phi_{\mathrm{P}}$ when considering non-polar classes falling intro brackets with $t_0 \leq t \leq t_1$ matching polar samples.

	Matching P questions $[t_0, t_1]$		
	$[100, T]$	$[68, 99]$	$[30, 67]$
Accuracy (%)	40.8	31.7	21.4
Nr. of NP classes	352	148	317

Results of this evaluation are summarized in Table 4.2. We can verify that there is a clear correlation between the accuracy of non-polar samples and how well-represented they are in polar space. Recall from the experimental setup, that the features being used, come from a feature extractor that has only been trained on polar samples! Looking at a concrete case, the first bracket shows that for 352 non-polar classes, it was possible to answer 40.4% of the samples in the validation set, based on a feature space that has only been trained using "*yes*" and "*no*" questions. For reference, the baseline in Table 4.1 indicates that this result is only $10.8 p.p.$ below a model that has been trained with non-polar questions exclusively!

Equally important are the result of the remaining two brackets. As the number of matching polar questions decrease, the accuracy over those non-polar classes decreases as well. The negative correlation strongly indicates that the semantic link between polar and non-polar samples creates a feature space that is complementary. In other words, training with polar samples that talk about non-polar classes, is enough to create a feature space that conveys enough information to answer non-polar questions.

4.4 Conclusions

In light of the results presented in the previous section, we can gather enough evidence to answer the two questions at the beginning of this chapter. We devised a strategy to measure the potential biases that overrepresented polar questions could be introducing on models that are trained on the entire VQA 2.0 dataset. Our experiments indicate that having substantially more samples for polar classes does not have a negative effect when training a model on both polar and non-polar samples concurrently. In fact, based on the results for the second question, we also discovered that the effect between polar and non-polar samples is complementary, in particular when polar questions have a semantic link to the non-polar samples.

For the second question, we were able to identify the relationship between features extracted from polar and non-polar samples. Experiments reveal a clear correlation between the performance of non-polar classes and the number of matching polar questions, showing that both polar and non-polar features complement each other.

4.4.1 Future Work

Results from this chapter are as exciting as they are promising. The synergetic relation between polar and non-polar samples has the potential of redefining the way VQA models can be trained. More exhaustive analyses of the link between samples of different polarities will enable a better estimation of the limits that the polar space has for expressing non-polar concepts. To explore further on this front, we see promise in defining alternative methods to link polar questions to non-polar classes *e.g.*, matching lemmatized terms or filtering based on a part-of-speech analysis. Ideally, a curated subset of VQA 2.0 that has validated semantic links, can shed more light into the limits that a polar training may have for learning non-polar concepts.

In summary

Neural networks can easily exploit superficial biases in a dataset, especially if a class is overrepresented w.r.t. the rest. This chapter discussed a scenario in the field of visual question answering, where one of the most popular datasets, VQA 2.0 has substantially more samples for two classes (YES and NO) than for the remaining 3127 categories. We proposed a series of experiments to measure the interaction of this imbalance in an intermediate feature space. Somewhat unexpectedly, findings indicate that polar and non-polar questions can be projected into the same space without exerting a negative influence on each other. In fact, both kinds of samples can complement each other when there are enough polar samples bearing a semantic relation to the non-polar classes.

In the next chapter we look at the concept of "*model capacity*"; a term that has been widely used in ML, but whose connotation is poorly understood.

5 What do Deep Networks Like to See?

> *The eye sees only what the mind is prepared to comprehend.*
>
> — **Robertson Davies**

Sometimes, powerful and deep neural networks don't end up learning as much as we expected them to. Some experts will think it's the lack of data; there are also those who will blame the training parameters, and others will say it was the model capacity. But wait, what is "model capacity"?

This chapter delves precisely into this question, studying the extent by which we can measure model capacity for large image classifiers.

5.1 What is Model Capacity

In their renown book "Deep Learning", Goodfellow *et al*. [50] mention two special cases of capacity. When considering a model in isolation, independent of data or any specific learning algorithm, we can conduct a theoretical analysis on the kinds of functions that the model is able to represent. For example, the model of order two $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ is well-equipped to fit all constant, linear, and quadratic functions. This property is known as the *representational capacity* of the model.

In contrast, when the model is being trained on concrete data, and using a particular learning algorithm, the number of functions that are representable decreases. How much it decreases depends on the specific version of the learning algorithm and, of course, the data. For example, if we train our previous toy model on data that is only linearly dependent, the space of solutions will not include quadratic functions (*i.e*. only

solutions where $\theta_2 = 0$). When talking about the solution space of a model used with this additional context, we speak of *effective capacity*.

Unfortunately, the scientific literature will often refer to "model capacity" without providing any precise definition. It is true that the role of this expression is also not central to support any major claims, and an appeal to the reader's common sense often suffices to settle the confusion. In general, researchers have used this expression to give a sense of scale, or how big a model is. We see the term being used by Szegedy *et al*. when talking about inefficient use of (adding more) convolutional filters for their Inception architecture [142]. Similarly, work on transfer learning and knowledge distillation has been using the term "model capacity" this way [125, 158]. The general implication is that a network with a larger capacity has more trainable parameters, more layers or more convolutional filters. In short, they are all referring to *representational capacity*.

Between the lines, however, we see that most of them try to convey other factors that go beyond counting the number of weights. This ambivalence is perfectly captured by the work of Wang *et al*. [153]. In their paper about developmental networks, they "explore . . . networks that grow in model capacity as new tasks [are] encountered." Their core contribution focuses on adding depth (or width) to maximize performance when fine-tuning a model. While discussing the growth of these models, the authors describe it in terms of model capacity. At one point, they even clarify that their work is aimed at increasing the *representational* capacity of the model—something they are indeed doing. However, their experiments are tailored to show performance improvements after fine-tuning on a downstream task. By involving data and a learning algorithm in their evaluation, the authors end up measuring *effective* capacity instead. Without a distinction between representational and effective capacity, the authors imply that a direct link exists between the number of trainable weights and the ability for the model to represent more information.

If adding representational capacity improves performance, doesn't it mean that the representational capacity has also increased? As it turns out, some researchers have been already questioning the link between the number of parameters a model has, and its ability to solve a task. For some, controlling capacity clearly goes beyond adding trainable parameters. For example, the early work of Neyshabur *et al*. [108] found that "size does not behave as a capacity control parameter, and in fact there must be some other, implicit, capacity control at play."

Their ideas, even though they were empirically validated on small, single-layer perceptrons, were fundamental for the analysis of Zhang *et al*. [165] on network generalization. A central idea of this highly cited paper revolves around finding factors that control the ability of models to represent information beyond the training set (*i.e*., generalization).

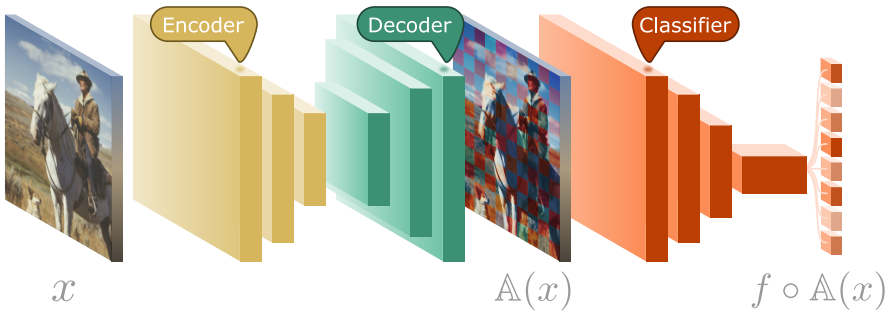


Fig. 5.1.: Overview of the proposed method and model. A pre-trained AE is fine-tuned with gradients flowing through a pre-trained image classifier whose parameters are fixed. After fine-tuning the combined network, images reconstructed by the AE preserve more information required by the classifier.

In a series of experiments, they challenged the idea of a straight link between effective and representational capacity by showing that modern networks can easily memorize large datasets without representing any actual information. Their main takeaway, similar to Neyshabur's, is that we still lack a precise formal measure to talk about representational capacity. This last statement was echoed again in 2021, after the authors republished their work, and noted that, "despite significant progress on theoretical understanding of deep learning in the past few years, a full mathematical characterization of the whole story [including model capacity] remains challenging" [164].

At this point, it seems as if not even fundamental research can elucidate on this matter. Even though representational capacity is straightforward to manipulate, factors that control effective capacity remain elusive. We seem to know more about what *isn't* effective capacity, but we still lack insights into the fundamental elements that control what a model can learn.

Our idea is to start small. So, instead of characterizing all necessary conditions that control effective capacity, we concentrate on measuring one sufficient condition. Concretely, we study the amount of information that is used by a model to make a prediction.

To this end, we propose a parametrized method that represents the input space by using autoencoders. Starting from a pre-trained autoencoder, we then fine-tune its decoding layers in order to represent the latent processes of the classifier which are responsible for filtering out input information. An overview of this setup is shown in Figure 5.1.

We then compare image reconstructions to the original samples, and establish the characteristics of the input space that different models rely on when making predictions. This allows us to test several hypotheses w.r.t. the amount of information used for classification. Is all information on each sample treated equally? Is information for foreground objects more important than the one from the background? Are there

differences between information at the center of the image and information found towards the borders? By addressing these questions we can reveal important properties of the processes that control effective capacity within a model.

Moreover, measuring the results in terms of information, allows us to approximate an upper bound on the amount of information that the model uses for classification. In doing so, we are essentially estimating an upper bound for the *effective* capacity that each model has. Results show that high-performance image classifiers rely on just a fraction of the information coming from the input space. These outcomes tell us that the effective capacity of modern neural networks is drastically reduced w.r.t. their representational capacity. Lastly, we show that the amount of information used by different architectures varies substantially, and also that this variation is not correlated to representational capacity nor performance.

The rest of this chapter is organized as follows: In the next section, we discuss theoretical reasons that support the use of input information as a vehicle to probe and discover properties of effective capacity. Then, we describe our proposed method including details about the autoencoder, how the fine-tuning takes place, and what metrics are used to compare reconstructions. Finally, we present and discuss the results of our experiments, drawing links to previous work that can be re-interpreted using our results.

5.2 Input Information: a Sufficient Condition for Controlling Effective Capacity

To understand our strategy, we need to clarify the constraints of the problem. First, we are restricting our study to deep (convolutional) neural networks for large-scale image classification problems. This means that we work with over-parametrized models *i.e.*, having more trainable parameters than those needed by an optimal solution. In other words, we consider models that have enough representational capacity. Moreover, as we focus on the input space, we are assuming that the learning algorithm (*e.g.*, optimization algorithm, regularization, learning rate) remain constant.

Next, we lay out the reasons why manipulation of input information is a sufficient condition to control effective capacity.

From Zhang's work [164], we know that the notion of effective capacity is linked to the generalization capabilities of a model. In turn, generalization is evaluated by how well the feature extractors in the model correspond to characteristics of the underlying data distribution, and not just to any peculiarities of the training sample. However,

information represented by these feature extractors depends mainly on the data that was used to train the model. In other words, the filters of the model only converge to patterns that are present in the data. Therefore, learned feature extractors are bounded by the amount of information that passes through them.

This last argument also implies that a model can't represent more information than that coming from the input distribution. Consequently, information of the training set represents an upper bound for the amount of information that a model can learn. Intuitively, a model that is trained on images of horses and fish will only have filters that extract information related to either class; we don't expect that these filters convey information about elephants unless the features are identical.

The strong dependency between the input space and effective capacity can be exploited to gain more insights about the latter. In particular, any process that defines a smaller upper bound for the input information, will also be an upper bound for the amount of information that the model is able to represent. Thus, we can say that this process is limiting (or controlling) the effective capacity of the model. Note that said factor may not account for all possible aspects that regulate model capacity. For that reason we highlight that a process bounding the input information is only a sufficient condition to control model capacity, not a necessary one.

The question at this point is: How low can the upper bound get?

5.2.1 How Low Can We Go?

For modern benchmarks like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [124] the amount of information in the input space defines the kind of rich and vast problem that makes effective solutions so fascinating. One of the first significant constraints to the input space comes from the reduction in dimensionality applied to samples that are used for training. For state-of-the-art classifiers, images are typically projected into a $224 \times 224 \times 3$ dimensional space, normally after the smallest dimension of the original image is rescaled to 256 pixels. But even then, an upper bound defined by just the number of input dimensions is still too high—particularly for this kind of natural images that exist in a space with over 150 thousand dimensions.

Even though we haven't even tried to measure how much information does the input space of ImageNet have, we know it is possible to constrain it further. As part of a follow-up work for the Inception architecture, Szegedy *et al*. [143] noted that input dimensionality provides merely a rough estimate of information content. Moreover, several dimensionality-reduction methods like t-SNE or UMAP have successfully exploited

the low-dimensionality of the semantic manifold that is associated with this kind of high-dimensional problems [150, 99].

By using autoencoders, we can steer reconstructions to only preserve information that a classifier uses to make predictions. Our empirical evaluation from Section 5.4 shows that a smaller upper bound can certainly be attained.

5.3 Methods

In this section we go over the modules and algorithms to control the input information used by image classifiers. First we describe the architecture of the autoencoder, the pre-training scheme and fine-tuning strategy. Second, we define information metrics that operate on input and reconstruction spaces, helping us determine what information is used for classification. Third, we compare the relationship between classifiers—and how much information they use—by evaluating different classifiers on reconstructions computed for other models.

5.3.1 Controlling the input space with Autoencoders

Our first goal is to gain control over the process that generates input data. Pure generative approaches like GANs [51] can be unstable and hard to train, especially for higher resolution samples like the ones comprising the ImageNet dataset. Instead, we get around this issue by making use of autoencoders. They allow us to obtain an approximation of the input space with learnable parameters that can be controlled and optimized afterwards.

As we aim at measuring information for high-performance image classifiers, we are interested in autoencoders that can encode and reconstruct large images. To this end, we choose an architecture initially proposed for image segmentation, popularly known as a SegNet.[1] The network consists on an encoder-decoder structure based on two mirrored copies of the VGG16 model from Simonyan *et al*. [130]. Besides the addition of batch-normalization layers across the model [65], the decoder is equipped with transposed convolutions, as well as *unpooling* layers that use the indices from the encoder (see Section 2.3 for more details about unpooling). As SegNet was originally proposed for

[1]The ideas of SegNet were published independently by two research groups between late 2014 and 2015. Badrinarayanan *et al*. [10] introduced the name SegNet while Noh *et al*. [110] used the name "unpooling" to refer to pooling indices used for upsampling.

segmentation, limiting the output channels to three is the only required modification to turn the original architecture into an autoencoder.

We implement the macro-architecture from Badrinarayanan *et al*. [10] that uses two sets of thirteen convolutional layers for encoding and decoding. The use of pooling operations and slow increase of the number of channels, creates a bottleneck in the middle of the autoencoder that has a compression ratio of 6:1.

5.3.2 Pre-training

After the architectural details have been sorted out, we proceed to train the autoencoder on the reconstruction task. This can be achieved by using an unsupervised loss function that punishes any differences between original samples and their reconstructions. Consequently, a trained autoencoder will approximate the equality $\mathbb{A}(\mathbf{x}) = \mathbf{x}$, where $\mathbb{A}$ represents the autoencoder.

As labeled data is not needed at this point, we can leverage the more than 99 million public domain images from the YFCC100m dataset [149] for training. Images that are part of this curated collection share important characteristics with ImageNet: both include mostly images of the real world, with indoor scenes, urban areas, natural landscapes, etc. We train using the mean squared error (MSE) per sample as the optimization function:

$$\text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{D} \sum_{i}^{D} (x_i - \hat{x}_i)^2 \tag{5.1}$$

where $\hat{\mathbf{x}} = \mathbb{A}(\mathbf{x})$ is the reconstruction of the original sample $\mathbf{x}$, and D is the total number of sub-pixels of the input. Apart from the MSE loss, we optimize using stochastic gradient descent (SGD) with momentum set to 0.9 and an initial learning rate $\lambda = 0.01$.

Due to the magnitude of the dataset, we observe that the autoencoder converges after training for only one epoch. For comparison, in terms of forward-backward passes, training for one epoch in the YFCC100m is roughly equivalent to training for 83 epochs on ImageNet! In order to include a learning rate schedule for training, we check if the MSE loss has not decreased after optimizing on two million images. If this is case, the learning rate is reduced to $\lambda_{\text{new}} = \lambda_{\text{old}}/5$. Note that after training, there is no need to evaluate on a validation set, as the autoencoder has only seen all training images exactly once. This means that the loss value of the last few training batches are a good estimate of the generalization loss.

5.3.3 Fine-tuning

Once the autoencoder can successfully reproduce information from the input space, we can forward its reconstructions to a pre-trained image classifier. Just as with a classifier alone $f(\mathbf{x})$, the composite function $f(\mathbb{A}(\mathbf{x}))$ can work the same way, except that the intermediate reconstruction is produced by the autoencoder. It is therefore possible to pass an original sample $\mathbf{x}$ through the ensemble, and evaluate the loss that the classifier used during training. During back-propagation, this allows the flow of gradients to extend past the parameters of the classifier, and into those of the autoencoder.

More formally, we can divide the composite function into encoder, decoder and classifier, each with its own set of parameters as follows:

$$f(\hat{\mathbf{x}}) = f(\mathbb{A}(\mathbf{x})) = f_{\boldsymbol{\theta}}(\mathcal{D}_{\boldsymbol{\phi}}(\mathcal{E}_{\boldsymbol{\psi}}(\mathbf{x}))) \tag{5.2}$$

where $\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\theta}$ are the parameters of the encoder $\mathcal{E}$, decoder $\mathcal{D}$ and classifier f respectively.

We can update just the parameters of the decoder by calculating the derivative of the classification loss w.r.t. the input of the classifier, and then applying the chain rule w.r.t. the parameters of the decoder $\boldsymbol{\phi}$:

$$\nabla_{\boldsymbol{\phi}} = \frac{\partial \mathcal{L}_f}{\partial \hat{\mathbf{x}}} \cdot \hat{\mathbf{x}}' = \frac{\partial \mathcal{L}_f}{\partial \mathcal{D}_{\boldsymbol{\phi}}(\mathcal{E}(\mathbf{x}))} \cdot \frac{\partial \mathcal{D}_{\boldsymbol{\phi}}(\mathcal{E}(\mathbf{x}))}{\partial \boldsymbol{\phi}} \tag{5.3}$$

Recall that our goal is to measure effective capacity *i.e.*, how much information is a model representing once it has been trained. For this reason, we assume that the classifier has been already trained to convergence on ImageNet. Furthermore, we keep the classifier frozen and enter a fine-tuning stage that updates only the decoder w.r.t. the classification task. Fine-tuning can be done using SGD, the loss function of the classifier *i.e.*, cross-entropy, and back-propagating according to Equation 5.3.

We know that information at the bottleneck of the autoencoder already carries enough information to let the decoder reconstruct the original input sample. Our proposed fine-tuning stage lets the decoder learn what parts of the input have to be preserved, and what parts can be ignored in order to maintain the performance of the classifier.

Once the fine-tuning has converged, we are left with an encoder-decoder architecture that reconstructs information that a classifier uses to predict. For our experiments, we fine-tune ensembles that correspond to a SegNet autoencoder followed by one of five possible image classifiers: LeNet [82], AlexNet [76], VGG16 [130], Inception-v3 [143] and ResNet50 [59]. Apart from the five obvious high-performance classifiers, we

include a re-implementation of LeNet to compare the effects of using a model with low representational capacity.

To prevent ambiguities when referring to different models, we add a subscript for autoencoders that have been fine-tuned on one of these classifiers. The letter corresponds to the first character of the classifier's name. (We refer to the classifiers as [L]eNet, [A]lexNet, [V]GG16, [I]nception-v3 and [R]esNet50.) For example, a SegNet that has been fine-tuned on LeNet will be denoted $\mathbb{A}_L$; when fine-tuning with ResNet50, the resulting encoder-decoder will be denoted as $\mathbb{A}_R$, etc. Similarly for classifiers, we use subscripts with the letter of the specific model. That way, f_I refers to an Inception-v3 and f_A to AlexNet. Composite models will be denoted using the function composition operator $\circ$ so, instead of $f_V(\mathbb{A}_V(\cdot))$, we write $f_V \circ \mathbb{A}_V(\cdot)$.

At this stage, we can consider three possible scenarios regarding information that the decoder will preserve or discard:

1. Is the available information equally useful for the classifier? If this is the case, we won't see any major differences between the reconstructions of the autoencoder before and after fine-tuning.

2. Is information going to be semantically discernible? In this case, we expect that information for foreground objects will be preserved more than the information of the background.

3. Is information in the center of the image more important than the contents around the borders? It is well-known that image classification datasets suffer from center-biases *i.e.* information relevant for prediction appears in the center of the image. Therefore, it is possible that the classifier assigns a higher credence to information that is closer to the center.

We will revisit these three questions after evaluating the results in Section 5.4.3.

5.3.4 Measuring Information

After fine-tuning the decoder of an ensemble $f \circ \mathbb{A}$, we can measure changes in the amount of information between original and reconstructed images. As a measure of information, we use the normalized mutual information (nMI):

$$\text{nMI}(X;Y) = \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} P_{(X,Y)}(x,y) \log \left(\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)} \right) \tag{5.4}$$

This metric allows us to estimate how much information from a target sample can be predicted based on a reference sample. For images, each reference sample is compared on a sub-pixel level to the target sample. Note that this metric is invariant to translations, which accounts for trivial shifts of the pixel values *e.g.* if the reconstruction is brighter everywhere.

In order to address the three questions we asked in Section 5.3.3, we focus on nMI comparisons that use the original input as a reference, but also different reconstructions from the same ensemble. The following two metrics are based on nMI, and differ only on the samples that are used as reference:

- **In-Class nMI (IC-nMI):** the nMI is calculated between the original sample, and the reconstruction of a fine-tuned autoencoder (or also the reconstruction produced by the pre-trained SegNet). A high IC-nMI means that most information from the original input has been preserved, and therefore, it is necessary for the classifier.
- **Between-Class nMI (BC-nMI):** calculated between two random reconstructions of the same fine-tuned autoencoder. In this case, each individual sample represents its own class. This metric is expected to be low, as it is unlikely that two random samples have any pixels in common. A high value indicates that a constant pattern exists between two samples (assuming there are no identical images).

For both metrics, we report the average nMI and standard deviation over the validation set of ImageNet. For BC-nMI, instead of evaluating every combination of two images, we shuffle the validation set of ImageNet and draw consecutive pairs of samples totaling 25 thousand pairs. A graphic description of both metrics is presented in Figure 5.2.

5.3.5 Comparing Information Between Models

With fine-tuned autoencoders, we can convey the information that one classifier uses for prediction. So far, the metric to evaluate information is relative to the classifier itself *i.e.*, how much information is dropped by one model. Given two classifiers that drop the same amount of information, how can we establish if they end up relying on the same information?

Thanks to the identical structure of the fine-tuned autoencoders, it is possible to ask if information preserved by one classifier is the same for other models. We can achieve this by evaluating the accuracy of pre-trained classifiers when being fed reconstructions produced by a different fine-tuned autoencoder. For example, we evaluate acc$(f_R \circ \mathbb{A}_A)$

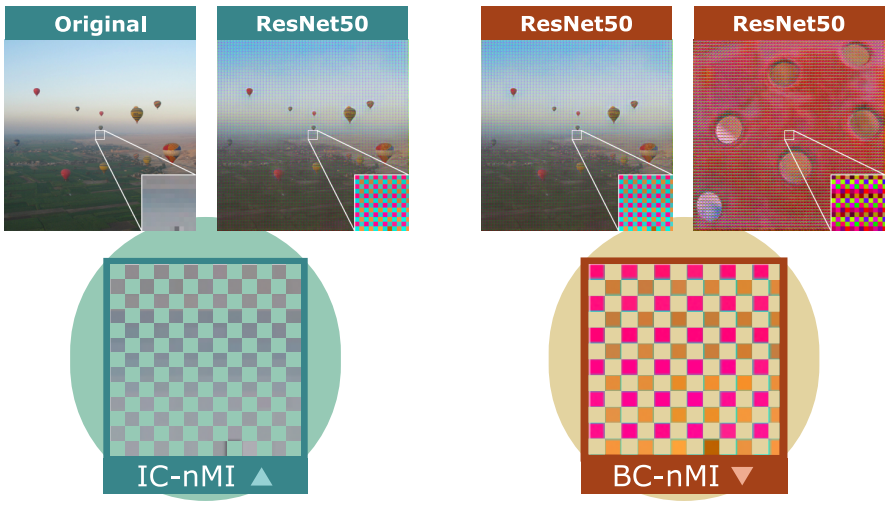


Fig. 5.2.: Visual representation of the proposed nMI-based metrics. IC-nMI represents input information preserved by one reconstruction. BC-nMI represents input information that is constant across reconstructions.

i.e., the accuracy of ResNet50 when using reconstructions produced by an autoencoder that has been fine-tuned for AlexNet.

In general, we evaluate the accuracy of ensemble models of the form $f_j \circ \mathbb{A}_i$, where $i \neq j$. Once again, we report the accuracy on the validation set of ImageNet for every valid pair of $i, j \in \{L, A, V, I, R\}$.

5.4 Results

Now that we have a plan for the experiments, this section goes over the results on each stage of the general method. At the end of each subsection we discuss the meaning of the results in the context of effective capacity, addressing the questions that were issued during the description of the methodology.

5.4.1 Pre-training

The final MSE of the autoencoder converged to 7.77×10^{-4} after training over the entire YFCC100m dataset. This value was stable for the last 8.5 million images, with fluctuations of only $\pm 1 \times 10^{-6}$. Nonetheless, we confirm that this result applies to unseen data by evaluating on the validation set of ImageNet. There, we obtain a reconstruction error of 8.05×10^{-4}.

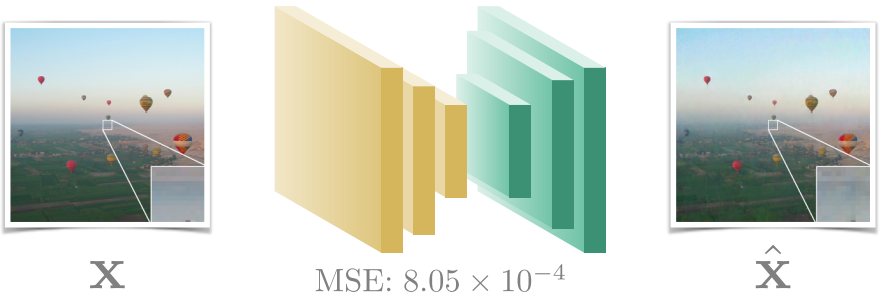


Fig. 5.3.: Reconstructions of the SegNet autoencoder after being pre-trained on the YFCC100m.

To get a better sense of scale, an error this small is equivalent to getting a perfect reconstruction for 80% of the image, while the remaining 20% is off by only one pixel value. Another way to think about it is if one of the three color channels changes by just a single value every five pixels (4.85 to be more precise). This is arguably an error too small to be even perceptible by the human eye, as it can be seen in Figure 5.3.

This reconstruction performance guarantees that almost all information from the input space is being represented by the autoencoder.

5.4.2 Fine-tuning

We use five instances of the pre-trained autoencoder from Section 5.4.1 and fine-tune the decoder with gradients from one of the following architectures: LeNet, AlexNet, VGG16, Inception-v3 and ResNet50. We use the pre-trained classifiers provided by the PyTorch project[2], except for LeNet which we have re-implemented ourselves.

For fine-tuning, we use the 1.2 million images from the training set of ImageNet, and report top-1 and top-5 validation accuracy. A summary of the fine-tuning results is shown in Table 5.1.

Results show that fine-tuning the decoder, didn't have a negative effect on the original performance of the classifier. We see that the less accurate models, LeNet and AlexNet, benefited the most from predicting on inputs that came from their fine-tuned autoencoder. Inception-v3 on the other hand, shows a slight decrease in performance.

Overall, performance fluctuations are small, and indicate that the decoder has learned to reconstruct information that is relevant for classification. We attribute changes in performance to noise patterns that are being filtered out in the reconstruction (for models whose performance improves). Conversely, for Inception-v3, small noise patterns

[2] `https://github.com/pytorch/vision`, commit 10a387a

Tab. 5.1.: Top-1 and top-5 validation accuracy (%) of different composite models on ImageNet.

Network	top-1	diff	top-5	diff
f_L	32.30		54.63	
$f_L \circ \mathbb{A}_L$	34.85	+2.55	57.79	+3.61
f_A	54.96		77.98	
$f_A \circ \mathbb{A}_A$	56.13	+1.17	78.96	+0.98
f_V	71.35		90.50	
$f_V \circ \mathbb{A}_V$	71.65	+0.30	90.55	+0.05
f_R	74.02		92.01	
$f_R \circ \mathbb{A}_R$	74.94	+0.92	92.27	+0.26
f_I	77.12		93.25	
$f_I \circ \mathbb{A}_I$	76.71	-0.41	93.03	-0.22

can cause a few predictions to fail. For context, a decrease of $0.4p.p.$ on a set of 50 thousand images corresponds to a change in prediction for just 200 samples.

As we are ultimately interested in measuring the global patterns of each model, and how much information it takes, we consider that the fine-tuned autoencoders are sufficiently preserving the input signal that is used for classification. Next, we have a look at the reconstructions, and see how much of the original information is left.

5.4.3 Measuring Information

Once we have fine-tuned all five autoencoders, we can proceed with the analysis of the reconstructions. First, we have a look at the overall appearance of the samples produced by each decoder, as well as by the pre-trained SegNet (see Figure 5.4). Reconstructions from all fine-tuned decoders have suffered perceivable changes that cover the entire area of each image.

More interestingly, we notice the appearance of checkerboard artifacts that seem consistent across samples produced by the same autoencoder. We observe that reconstructions for VGG16 and Inception-v3 appear to have more subtle patterns than those for ResNet50 and AlexNet. On the other hand, samples optimized for Inception-v3 have suffered a global change of hue, similar to a gamma correction.

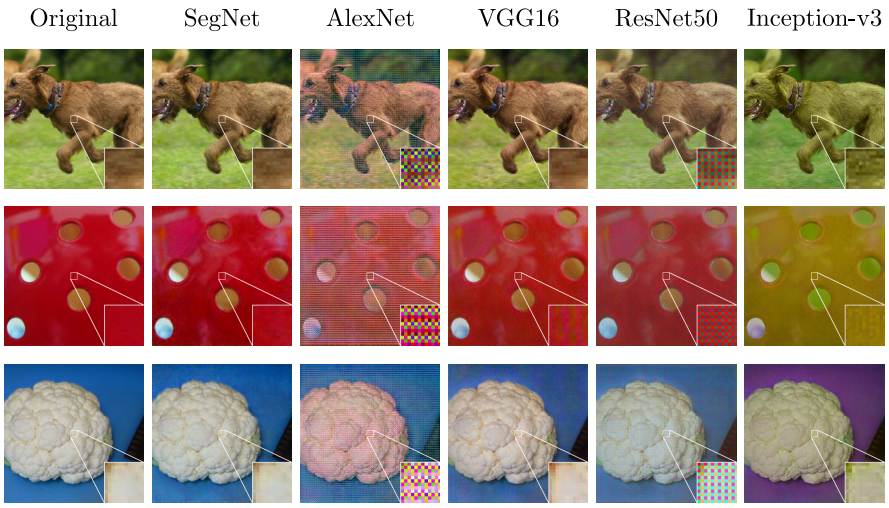


Fig. 5.4.: Reconstructions of fine-tuned autoencoders. Checkerboard artifacts are consistent across samples but different between models.

To verify the consistency of the checkerboard artifacts we use the information metrics proposed in Section 5.3.4, namely IC-nMI [▲] and BC-nMI [▼]. Results for these two metrics are shown in Figure 5.5.

These experiments provide support for some of our initial observations. We corroborate through the BC-nMI that the checkerboard patterns are lower for Inception-v3 and VGG16 samples. In contrast, the BC-nMI of reconstructions for AlexNet and ResNet50 are substantially larger, confirming that their artifacts are constant throughout the dataset.

In terms of information, these constant patterns reveal that there are positions in the images that carry no information for the model. This way, we can establish that ResNet50 and AlexNet discard the most information, as the checkerboard artifacts are more constant for their reconstructions.

The other aspect of information deals with the values that have been preserved w.r.t. the original samples. We see that the original pre-trained SegNet preserves the most information, according to its IC-nMI. Regarding classifiers, VGG16 and Inception-v3 are again those that preserve most of the original input information, as shown by the higher values in IC-nMI.

At this stage, we can revisit the hypotheses from Section 5.3.4 and answer them in light of the results we have so far.

1. Is the available information equally useful for the classifier? As performance on all classifiers remains stable when using samples that have filtered out a good portion

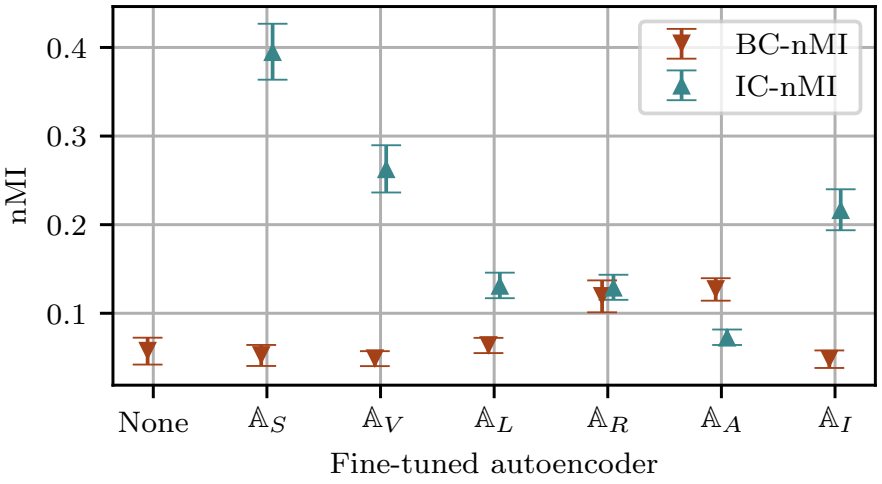


Fig. 5.5.: Normalized mutual information for input samples reconstructed from different autoencoders. IC-nMI measures information between reconstructions of the same sample while BC-nMI focuses on information between reconstructions from the same autoencoder.

of the information, we conclude that this is clearly not the case. Depending on the classifier, some information is going to be important, while some is completely irrelevant.

2. Is information going to be semantically discernible? Thanks to the BC-nMI, we know that checkerboard artifacts are, to some degree, constant across images, regardless of where the subject and background appear. We conclude that information from the input is not discernible for the network at this stage.

3. Is information in the center of the image more important than the contents around the borders? Similarly to the previous question, the constant positioning of the artifacts indicates that there is no preference for information towards the middle of the image compared to the contents towards the edges.

5.4.4 Comparing Information

We compare the fluctuations in validation accuracy when models rely on the input reconstructions from other models. Results of this evaluation are shown in Table 5.2.

As we can see from these results, the accuracy varies drastically depending on the combination of model and autoencoder. Notice that the accuracy of any given combination depends strongly on the accuracy of the lowest performing classifier. In particular, for every pair of models f_i, f_j, we observe that the accuracy of the composite alternatives $f_j \circ \mathbb{A}_i$ or $f_i \circ \mathbb{A}_j$ is consistently limited by the classifier that had the lowest performance

Tab. 5.2.: Top-1 validation accuracy (%) on ImageNet for classifiers using different fine-tuned autoencoders.

	f_L	f_A	f_V	f_R	f_I
$\mathbb{A}_L$	34.84	30.77	4.16	47.30	43.52
$\mathbb{A}_A$	2.11	56.13	0.97	9.25	53.75
$\mathbb{A}_V$	29.29	53.62	71.63	73.00	74.00
$\mathbb{A}_R$	1.63	49.72	7.10	74.94	72.49
$\mathbb{A}_I$	18.29	30.24	45.55	45.40	76.71

on the original input. For example, evaluating f_R on reconstructions from $\mathbb{A}_A$ results in an accuracy that is lower than the original AlexNet. The other way around also holds: evaluating $f_R \circ \mathbb{A}_A$ yields an accuracy that is still below the original performance of AlexNet.

To capture this behavior, we define a normalized metric based on accuracy we call the *minimum classification ratio* (CR_{min}). This metric is defined as the accuracy of the ensemble divided by the lowest accuracy among the respective classifiers.

Formally, for the ensemble $f_j \circ \mathbb{A}_i$, the CR_{min} metric is defined as follows:

$$\text{CR}_{\text{min}}(f_j \circ \mathbb{A}_i) = \frac{\text{acc}(f_j \circ \mathbb{A}_i)}{\min(\text{acc}(f_i), \text{acc}(f_j))} \tag{5.5}$$

In Table 5.3 we present the CR_{min} values for all ensembles, based in the results from Table 5.2. The rows of CR_{min} values show how much information can be extracted from the signal that was tailored for that classifier *i.e.*, the signal used to fine-tune the autoencoder.

Tab. 5.3.: ImageNet CR_{min} (%) for classifiers using different fine-tuned autoencoders.

	f_L	f_A	f_V	f_R	f_I
$\mathbb{A}_L$	100.0	88.30	11.93	135.75	124.90
$\mathbb{A}_A$	6.06	100.0	1.74	16.47	95.75
$\mathbb{A}_V$	84.05	95.53	100.0	102.82	104.23
$\mathbb{A}_R$	4.67	88.57	9.99	100.0	96.74
$\mathbb{A}_I$	52.50	53.87	64.16	60.59	100.0

These metrics reveal interesting properties about the use of information from different models. From the diagonal of Table 5.2, we see that the models perform best when provided with information from their own fine-tuned decoder. Table 5.3 shows that the signal from VGG16 can be used more effectively by other models, aligning with the

nMI measurements from Figure 5.5. In contrast, the signal from Inception-v3, despite preserving most of the information, cannot be used by other classifiers as effectively.

Even though the signal from VGG16 preserves the most information, the model itself is not extracting as much as it could. According to Table 5.3 both ResNet50 and Inception-v3 perform better with the signal from VGG16, than VGG16 itself. We can conclude from this observation that VGG16 has less effective capacity than ResNet50 or Inception-v3. Despite VGG16 having almost six times more parameters than ResNet50 or Inception-v3, the smaller networks manage to make a more effective use of the information.

A similar dynamic can be observed between Inception-v3 and ResNet50. From Figure 5.5, we see that ResNet50 introduces strong artifacts, leaving less original information to process. However, Table 5.3 shows that Inception-v3 is capable of performing almost at the same level of the original ResNet50. More interestingly is seeing that the opposite does not hold. Even though there is more information available in the input space of Inception-v3, ResNet50 cannot use it effectively.

By now, we see several cases where signals are compatible with other models, and some where they are not. Intuitively, it seems as if some models "work" with other signals and some don't. More importantly, these interactions between models and signals are not symmetric.

One way to capture this partial order is by using Formal Concept Analysis (FCA) [155]. In general, FCA provides a way to examine partial order relations between elements w.r.t. a set of binary attributes. The output of FCA is a lattice that conveys a hierarchy between elements and their attributes. Elements that are placed high in the lattice have all the attributes connected below. Similarly, attributes that are placed lower are shared among connected elements placed higher in the lattice.

In our case, we are interested in analyzing classifiers, and how well they work with signals from other classifiers. Hence, we can let each classifier be an element, while the attributes correspond to the signal from each fine-tuned autoencoder. The interpretation of an element i having the binary attribute j is that "the signal for model j works well for classifier i." After building the intuition for applying FCA, we still need a more formal method to define attributes, such that it can be represented as a binary function. For this, we can use the CR_{min} metrics from Table 5.3 and define a threshold t such that everything above t is set to one, and zero otherwise.

Since we want to convey the notion that a model works "well" with a given signal, we set $t = 90\%$ which yields the lattice shown in Figure 5.6.

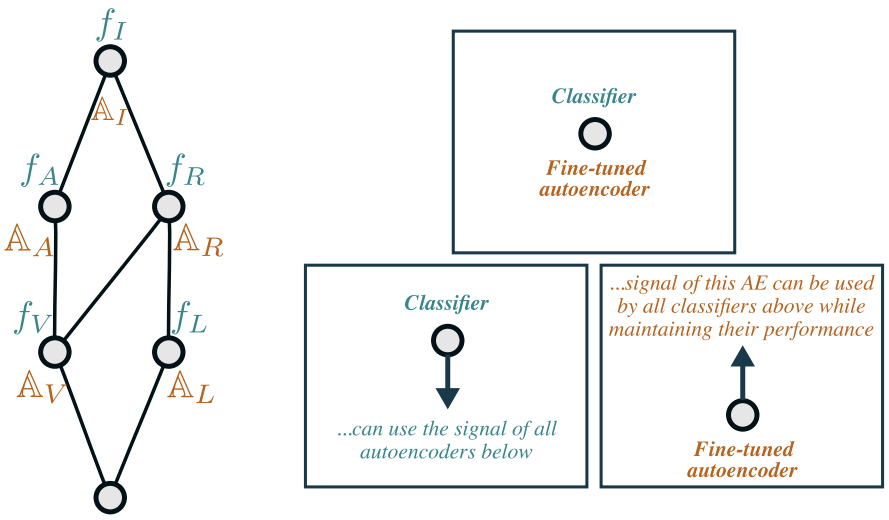


Fig. 5.6.: FCA lattice using $\mathrm{RC}_{\min}$ and a threshold $t = 0.9$.

The relations in the lattice confirm some of our previous observations. Starting from the top, Inception-v3 is the classifier that "works well" with all other signals. (The interpretation of "works well" is given by the $\mathrm{RC}_{\min}$ with a threshold $t = 90\%$ and means that "the model has an accuracy equal to the lowest performing one between itself and the model used to fine-tune the signal".) The signal for Inception-v3 is also the one that does not convey enough information for any of the other models.

Looking at the lower nodes of the lattice, we see that signals from VGG16 and LeNet are sufficient to make all the other high-performance classifiers work well. For LeNet, this is partially because it has the lowest performance of all and hence, any other model will only need to match its accuracy. It is therefore surprising that the more powerful VGG16 doesn't meet this criterion. Finally, signals from AlexNet and ResNet50 can only be exploited effectively by Inception-v3, despite having the highest loss of information, according to Figure 5.5.

5.5 Relation to Previous Work

Throughout the years, experimental results accumulate, but sometimes some results remain unaccounted for. We found a few cases where the results either align or can be explained through our findings regarding effective capacity.

Back in 2014, Zeiler *et al.* [163] reported that AlexNet was highly sensitive to local structures. Knowing the magnitude and consistency of the checkerboard artifacts we

have seen in the signal of AlexNet, it is reasonable to expect local changes to have a large effect in the final prediction for this model.

Raghu *et al*. [118] have shown that parameters of the first layers have more impact on a prediction than those found in deeper layers. If those changes are causing the model to drop more information, we can expect them to have a stronger impact in performance than deeper layers, where information already is limited.

Visualizations of input relevance done by Montavon *et al*. [103] show maps that are coarser for CaffeNet—a model similar to AlexNet—than for an Inception-like architecture. This phenomenon is consistent with the work of Lapuschkin *et al*. [79] and is in line with the measures of information from Section 5.4.3.

While exploring the limits of AlexNet, Bau *et al*. [11] trained a variant of the model that had wider layers and used global average pooling. They noted that the accuracy was still similar to the original architecture, even after increasing by four and even by eight the number of filters of the last convolutional layer. They hypothesize that the effective capacity of the network had already been exhausted. In light of our results, we know that the amount of information used by AlexNet is already quite small. It is likely that information reaching the deeper end of the architecture is already too simple (*i.e*. linearly separable), and cannot benefit from the additional representational capacity.

5.6 Conclusions

This chapter has tackled the challenging task of quantifying effective capacity. Because the ML community still lacks a well-defined framework to talk about this issue, we propose a strategy to measure model capacity by focusing on a sufficient condition: input information.

We have discussed and measured input information that image classifiers use to make predictions. We propose an architecture based on autoencoders that reconstructs the input space, with the goal of tuning out all information that is not used for classification. Metrics based on mutual information allow us to measure the reconstructed input, and discover which architectures use more or less information. A comparison between models is also possible thanks to an analysis of the partial order that exists between models that can use the signal that other models are trained on.

Our results let us conclude that VGG16 has a lower effective capacity than both ResNet50 and Inception-v3, despite having more trainable parameters. Inception-v3 has the highest effective capacity, as it is able to produce correct predictions based on signals from all

other models. The strong and constant artifacts from reconstructions based on AlexNet, lead us to conclude that its effective capacity is limited, despite having more than double the number of parameters than ResNet50.

In Summary

What is model capacity? When referring to *effective* capacity, we still don't have a good idea. We lack the tools to identify what are the necessary conditions that control model capacity. However, this chapter identifies one sufficient factor that let us measure the effects that capacity has in high-performance classifiers. By modeling the input space with autoencoders, we are able to measure how much of that information is used to make predictions. Aggregating these measurements over a set of classifiers, allows us to do a comparative analysis of the models that use more information and hence, which models have more effective capacity than others.

In the next chapter, we exploit these insights, and the properties of fine-tuned autoencoders to tackle another pressing challenge in ML and computer vision: adversarial attacks.

6 Robustness against Adversarial Attacks by Limiting Capacity

> *All that I desire is to be enriched by intensely exciting new thoughts.*
>
> — **René Magritte**

Barely one year after AlexNet set the new standard for image classification, Szegedy *et al*. [144] found that neural networks were not as robust as initially thought. While searching for hard negatives (*i.e*., samples that should be predicted with high probability, but are given a lower one instead), their work showed that adding small perturbations to any image, would cause a model to make a mistake. Methods that generate this kind of perturbations are now known as adversarial attacks, and counteracting their effects has proven to be an ambitious goal.

How are these attacks operating, and what can be done to mitigate their effects? This chapter shows that we can leverage our new insights about the effective capacity from Chapter 5 to make models more robust against adversarial attacks.

6.1 Introduction

The discovery of adversarial attacks sparked an immediate interest from the research community. The idea of imperceptible changes throwing off a high-performance model, was not only intriguing but also worrisome. As mentioned in Chapter 1, ML models are proliferating into many high-stake scenarios, partly because of their perceived robustness, and consistent track record. At the same time, numerous strategies to generate tiny perturbations are prompting users, and developers alike, to question the reliability of their models.

Research on adversarial attacks can be grouped according to one of three main objectives. In essence, most efforts are spent studying attack and defense mechanisms, but at the same time, they try to understand why these attacks are so effective.

On the attack front, we have algorithms that generate adversarial perturbations under different conditions. Most well-known attacks assume that the model is differentiable and hence, can calculate how the output of the model gradually changes as the input is slightly perturbed. In other words, they extract gradients from a target model in order to calculate the smallest perturbation that can change the prediction. These principles have been followed by popular gradient-based attacks like FGSM [52], DeepFool [105] or C&W [24]. Other attacks that do not directly rely on gradients, design strategies that probe the predicted output, and approximate the rates of change used by gradient-based methods. Prominent examples of this strategy have been proposed by Brendel *et al*. [17] and Papernot *et al*. [113].

Another concerning property of adversarial attacks, is how easy it is to re-use them for different models. Liu *et al*. [91] showed they could easily craft attacks based on one network, and then transfer them successfully to fool a different model. Others took this idea even further and found that even a single adversarial perturbation could be applied to an entire dataset and achieve a high misclassification rate [106, 116]. These "universal" perturbations were later exploited by Brown *et al*. [18] to craft physical perturbations that could be printed and used in the real world.

In contrast to attackers, efforts to counteract the effects of adversarial perturbations have been more transient, but not for lack of trying. There are no shortage of ideas on how to detect or suppress adversarial perturbations [46, 114, 136, 87, 100, 38, 157]. Moreover, prophylactic approaches have concentrated their attention in obfuscating the gradients that attackers rely on [19, 55, 156, 152, 117]. The rationale is that gradients that are small, or otherwise unstable can't be used by attackers to craft adversarial perturbations.

Unfortunately, these initiatives have been short-lived. Time and time again, attackers have shown that they can overcome adversarial defenses without changing their underlying method. In one of their most cited papers Athalye, Carlini and Wagner showed that seven out of eight papers that were accepted at ICLR 2018 could be easily circumvented [9].[1] They showed that most defenses were rendered useless by simple adjustments to a few hyperparameters, or by creating proxy functions to approximate hard-to-get gradients. They repeated this exercise with two other ideas that appeared in that year's CVPR conference [8].

[1] Their publication gained popularity partly because it was uploaded to arXiv just a few hours after the list of accepted papers at ICLR had been made public.

The third area of research is centered on understanding the origins of adversarial attacks. This is a highly debated arena where no real consensus has arisen. Initially, Goodfellow *et al*. [52] challenged the original idea that non-linearities inside the model were the main cause of adversarial vulnerability. Instead, they claimed that the linear parts of neural networks (*e.g*., when using ReLUs) were in fact the culprits. Tackling the issue from another angle, Nguyen *et al*. [109] showed that models were also vulnerable to synthetic images. They were able to craft samples that do not resemble natural pictures, but are nonetheless classified with high confidence. These findings are in line with Szegedy's who showed a similar behavior using random images, and optimizing them to get a high prediction confidence. Other aspects like optimizers, and the structure of decision boundaries have been considered as the possible cause of adversarial vulnerability [146, 45, 107]. All these ideas, have sparked heated debates, but no final verdict has come out of them yet. Without a clear horizon, we are left with more somber outlooks like the one made by Shafari *et al*. They argue that adversarial attacks are fundamental to the feature space of modern ML models, and thus, inevitable [128].

6.1.1 Constructing a Defense Mechanism

Even though the community around adversarial attacks remains active, effective strategies to guard against adversarial attacks are scarce. One prominent defense strategy that remains undefeated to this day is known as "adversarial training". Initially proposed by Athalye *et al*. [8], the most effective way to defend against adversarial attacks is simply to include perturbed samples in the training set. Similar ideas have been pushed forward by Kannan *et al*. [69] or Sinha *et al*. [132], although the core strategy remains the same. However, adversarial training comes with its own limitations. For instance, it doesn't work for pre-trained models, it becomes robust mostly to those attacks that are included in the training set, and there is often a large trade-off in performance.

Is there anything else we can do? One advantage of results in XAI, is that they can be often exploited for other purposes. We saw from Chapter 5 that it is possible to measure model capacity by controlling information of the input space. At the same time, adversarial attackers are confined to the input space when introducing malicious perturbations. Thus, a common ground exists (*i.e*., the input space) where these two forces meet when trying to modify the input that reaches the model.

One of the main findings from the previous chapter is that fine-tuning an autoencoder with gradients from a classifier, produced reconstructions that had less information than the original samples. In particular, we were able to establish that the missing information corresponded to fixed positions in the image. These fixed positions are equivalent to

having a function that sets certain features in a hyperdimensional space to a constant value. In turn, this constancy can be interpreted as a projection from a high- into a lower-dimensional plane.

Let us now think about the fine-tuning process that taught the decoder to fix those pixel positions. During back-propagation, the gradient of the classifier w.r.t. the reconstruction from an autoencoder had to be high in positions where the final reconstruction changed the most. In other words, the decoder, as it was fine-tuning, was able to lower the loss—and the gradient magnitudes—when it produced the constant checkerboard artifacts at the output. Looking at how adversarial attacks rely on gradients, we can infer that image positions that produce high gradients are those that have a bigger influence on the model prediction if changed. Therefore, an attacker would be most likely to succeed if it perturbs those positions.

So, knowing how fine-tuned autoencoders operate, and how adversarial attacks work, we are interested in evaluating how much can our fine-tuned autoencoders mitigate the effects of adversarial attacks. The setup for the robust model is similar to the one in Chapter 5, with a fine-tuned autoencoder preceding a classifier as shown in Figure 6.1.

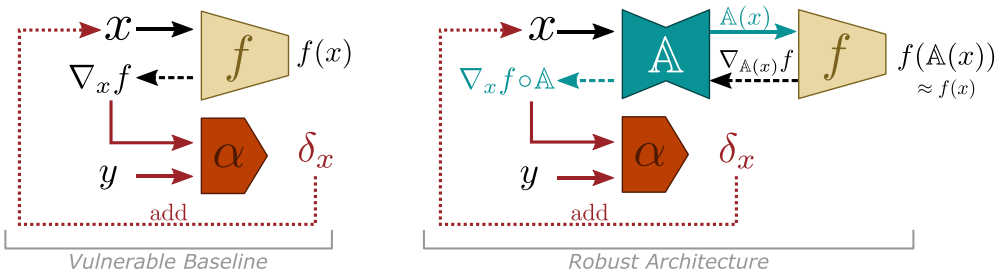


Fig. 6.1.: Overview of the proposed architecture. The use of a fine-tuned autoencoder mitigates the effectiveness of adversarial attacks α when placed before a pre-trained classifier.

From the start, we identify two properties that makes this setup robust to potential attacks. First, as discussed above, reconstructions from the autoencoder re-project positions of the image with high gradients. Now, the fine-tuned autoencoder is most certainly not producing reconstructions whose gradients are exactly zero at those positions. This means an attacker can still detect those vulnerable places, and create perturbations that modify their values. However, for those cases, having an adversarial image reconstructed through the fine-tuned autoencoder will set values at those positions back to a more constant value.[2]

A second property that works against adversarial attackers is the two-stage training of the autoencoder itself. Concretely, the part that corresponds to the encoder has only

[2] As reconstructions are based on imperfect approximations, there is still room—although arguably smaller—for perturbations to leak, leading to an incorrect prediction.

been trained on an unsupervised loss, and is not biased towards any class-relevant information. We know that, for the reconstruction task, areas with sharp corners or high color contrast are difficult to represent for autoencoders. These in turn, are the areas of an image that still produce a high reconstruction error. We also know that an attacker has to get gradients that pass through (and back from) the encoder. Even if these gradients are computed on the classification loss, the function that the encoder approximates is still most sensitive to areas that it can't reconstruct well.

In consequence, we hypothesize that an attacker will get gradients that are more relevant to the reconstruction task, and not to the classification objective. There are of course, parts of the structure of an image (*e.g.*, the outline of an object) that are relevant for the classification task. However, these regions are much sparser, and are not necessarily the ones that the classifier is most sensitive to.

Having analyzed the properties that our setup has in order to counter gradient-based adversarial attacks, we run a comprehensive evaluation that measures the extent by which a model can be defended using this paradigm.

In the remaining parts of this chapter we formalize and evaluate our proposed mechanism to mitigate the effects of adversarial attacks. The next chapter gives a quick recap of the defense architecture. Afterwards, we define the conditions under which the attacks take place, including a description of the dataset, parameters of the attacks, and evaluation metrics. Finally, we compare our results to vulnerable baselines and discuss the advantages of our proposed approach.

6.2 Methods

In this section we present a brief overview of the model architecture but more importantly, we describe the experimental setup to measure adversarial robustness. Furthermore, we propose three ablation experiments to identify factors that make the ensemble more resilient to adversarial perturbations.

6.2.1 Model Overview

As in Chapter 5, we build a composite architecture that consists of two main modules. The first is a high-performance image classifier that is pre-trained on a large scale dataset. Concretely, we use ResNet50 and Inception-v3 architectures trained on ImageNet as base classifiers. The second module consists on an encoder-decoder architecture that

precedes the classifier, with the goal of controlling the input signal that reaches the classifier. This last part, starts as an undercomplete autoencoder based on the SegNet architecture [10, 110], and pre-trained via an unsupervised reconstruction loss on the YFCC100m dataset [149]. After, the decoder is fine-tuned by forwarding reconstructions to the classifier, and back-propagating according to the supervised loss used originally for the classifier. For more details about the architecture and its training process, please refer to Chapter 5.

6.2.2 Threat Model

As part of the recommendations to assess adversarial robustness, Athalye *et al*. [9] insist on the importance of defining all conditions under which attacks take place. Thus, we compile a list with all relevant aspects for our evaluation of robustness against adversarial attacks a.k.a. the threat model. The values for hyperparameters and choice of algorithms are based on the setup used by Guo *et al*. [55], as they were one of the firsts to run large-scale evaluations on ImageNet.

- **Dataset**: As mentioned above, we use ImageNet as the standard to train and evaluate. The classifiers are all pre-trained using this dataset, as well as the fine-tuned autoencoders. Adversarial attacks are computed on the full validation set (50 thousand samples). Performance metrics are based on the average output of this set.

- **Image Classifiers**: we report results on two high-performance image classifiers, namely Inception-v3 and ResNet50. Both classifiers have been pre-trained on ImageNet, with no special considerations regarding adversarial vulnerability. When used alone, they are considered (vulnerable) baseline models. We use the notation from Chapter 5 to refer to classifiers *i.e*., we add a subscript corresponding to the first character of their names. This way, Inception-v3 and ResNet50 will be denoted as f_I and f_R respectively.

- **Defense Method**: it comprises a pre-trained classifier that precedes a fine-tuned autoencoder as described in Section 6.2.1 and illustrated on the right side of Figure 6.1. We refer to the ensemble of autoencoder and classifier as a Structure-to-Signal Networks or StSNet (pronounced *es too es net*) and denote it formally as $f_i \circ \mathbb{A}_i$ where $\mathbb{A}_i$ is the autoencoder that has been fine-tuned on classifier f_i. Note that the autoencoder always corresponds to the classifier it is connected with. One exception is a baseline mechanism consisting of just the pre-trained autoencoder (without the fine-tuning step) followed by a ResNet50, which we denote as $f_R \circ \mathbb{A}_S$.

- **Perturbation Magnitude**: an important property of adversarial attacks is the size of the perturbation that is being added to clean samples. The intuition is that small perturbations are imperceptible and hence, difficult to detect; having bigger perturbations leads to more noticeable artifacts. However, these larger perturbations are more likely to cause a model to make a mistake, which is the ultimate goal of an adversarial attack. Therefore, we can measure the effectiveness of an adversarial attack (or the robustness of a model) by how predictions change as the magnitude of perturbations grow. We report how imperceptible adversarial attacks are by measuring the normalized L_2 norm of the perturbation:

$$\bar{L}_2(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{n=1}^{N} \frac{||\mathbf{x}_n - \hat{\mathbf{x}}_n||_2}{||\mathbf{x}_n||_2} \tag{6.1}$$

  where samples that have not been perturbed are denoted as $\mathbf{x}$ (*i.e.* the clean samples), adversarial samples are denoted as $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$, and N corresponds to the total number of samples in the set.

- **Defense Strength Metric**: to measure how robust a model is against adversarial attacks (as the attack strength increases), we plot accuracy against the average perturbation magnitude. Note that only the set of true-positive samples is used to compute the perturbation magnitude, as false-positives are already considered adversarial samples with $\boldsymbol{\delta} = \mathbf{0}$.

- **Attack Methods**: all models (baseline and robust variants) are tested against three well-known gradient-based attacks. These methods cover a range of attacks that are both highly effective but also efficient to compute. After each perturbation is applied to an image, some attacks may have added changes that do not correspond to valid pixel values. Therefore, we make sure that perturbed images are being cast back to valid pixel values before running the evaluation. We conduct all experiments using *untargeted* adversarial attacks. Under this condition, an attack is considered to be successful as soon as the ground-truth class $y^\star$ is no longer the predicted class. This is in contrast to *targeted* attacks (which are harder for attackers) where a preselected target class $y_t \neq y^\star$ has to be predicted for the attack to succeed. The attacks used for evaluation are:

  - *Fast Gradient Sign Method (FGSM)* [52]: an effective single-step method that is shown to be transferable across different models. This method relies on the sign of the gradient of the classification loss w.r.t. the input:

$$\hat{\mathbf{x}} = \Gamma(\mathbf{x} + \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} \mathcal{L}_f)) \tag{6.2}$$

where Γ is a clipping function that makes sure that the perturbed image remains in the range of valid images. For FGSM, we evaluate perturbations with $\epsilon \in \{0.5, 1, 2, 4, 8, 16\}$.

– *Basic Iterative Method (BIM)* [77]: also referred to as i-FGSM, it is an iterative version of FGSM. The final adversarial sample is the result of iterating n times over the following function:

$$\hat{\mathbf{x}}_0 = \mathbf{x}; \quad \hat{\mathbf{x}}_i = \Gamma(\hat{\mathbf{x}}_{i-1} + \epsilon \, \text{sign}(\nabla_{\hat{\mathbf{x}}_{i-1}} \mathcal{L}_f)) \tag{6.3}$$

We use $\epsilon \in \{0.5, 1, 2, 4, 8\}$, and a fixed number of iterations $n = 10$. Consequently, perturbed images have changes that are at most 10ϵ big.

– *Carlini-Wagner L_2 (C&W)* [24]: an optimization-based method that has proven to be effective even when models are being equipped to handle adversarial attacks. This method starts with the premise of finding a small perturbation $\boldsymbol{\delta}$ that causes a classifier to predict something other than the ground-truth target $y^\star$. Formally, the attack finds a $\boldsymbol{\delta}$ such that $f(\mathbf{x} + \boldsymbol{\delta}) = y$ where $y \neq y^\star$. To find such $\boldsymbol{\delta}$, the authors propose solving an optimization problem based on the following proxy function:

$$\arg\min_{\boldsymbol{\delta}} ||\boldsymbol{\delta}||_2^2 + c \cdot \phi(\mathbf{x} + \boldsymbol{\delta}) \quad \text{s.t.} \quad \mathbf{x} + \boldsymbol{\delta} \in [0, 1]^n \tag{6.4}$$

where $\phi(\mathbf{x}) = \max(\max(Z(\mathbf{x})_{i \neq y} - Z(\mathbf{x})_y), -\kappa)$ is the proxy function that controls the confidence with which the classifier predicts the adversarial class y instead of any other. The function $Z(\mathbf{x})$ returns the logits of $f(\mathbf{x})$ (*i.e.*, the last linear layer, before normalization), and κ controls the magnitude of the confidence. For the optimizer, we use Adam[73] and let it run for up to 100 iterations with an initial learning rate of 1×10^{-3}. To control ϵ we follow the same strategy from Guo *et al.*, and set the hyperparameters $c = 10, \kappa = 0$. Once we have a perturbation $\boldsymbol{\delta} \in [0, 1]^n$, we multiply it by $\epsilon \in \{0.5, 1, 2, 4\}$.

- **Other Considerations**: even though we are not working under the assumption that the model has to comply with real-world security standards, we assume that the attacker has knowledge about the model architecture and training conditions. This scenario is known in the literature as White-Box attacks, and it is the most challenging setting to evaluate adversarial robustness. Cases where the attacker has only partial (or no) knowledge about the model or how it was trained, are referred to as Gray-Box and Black-Box respectively.

With a full description of the threat model, running the experiments can already give us strong evidence on whether our proposed mechanism is effective against adversarial attacks. Nonetheless, we also do more analyses on the properties of the ensemble that contribute the most to the defense.

6.2.3 Probing the Projection Space

Experiments form the previous section are designed to show that an ensemble $f_i \circ \mathbb{A}_i$ causes an increase in robustness against adversarial attacks. However, there is no evidence regarding the properties of the ensemble that are contributing the most to said improvements. Most critically, we need to show that the increased robustness is not caused by gradient obfuscation, something Athalye *et al.* [9] have systematically circumvented.

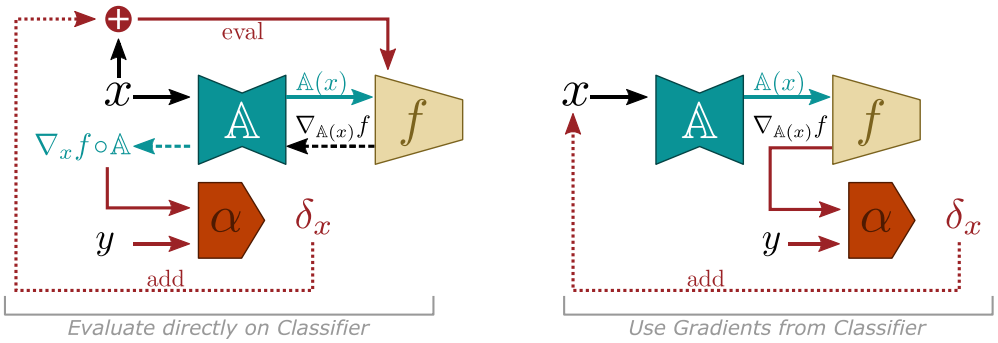


Fig. 6.2.: Ablation experiments. Left: Craft adversarial attacks on the ensemble but forward the perturbed sample directly to the classifier. Right: Use gradient information from the classifier and feed the resulting adversarial sample through the ensemble.

To address this gap, we propose the following three additional experiments:

Feed-Forward Directly into the Classifier

In Section 6.1.1 we saw that the fine-tuned decoder acts as a projection mechanism for pixels that appear fixed for all samples. Therefore, perturbations that modify those positions won't have a strong influence, as the decoder will project them back to a constant value.

To measure this effect, we propose the following ablation experiment: We let the attacker compute perturbations on the ensemble $f_i \circ \mathbb{A}_i$. Once the attack is ready, we feed-forward the perturbed sample directly into the classifier f_i. This way, the attack is not weakened

by the projection of the decoder, and any changes to the pixel values that would end up fixed otherwise, can directly affect the classifier.

Our hypothesis is that the attacker doesn't exploit these positions, in part because of the strict projection that the decoder imposes there. Therefore, small pixel adjustments for those positions lead to no changes in classification, precisely because said adjustments get squashed by the decoder. Overall, the results of this experiment should be comparably robust to the original experiments that pass adversarial attacks through the whole ensemble.

An illustration of this setup is shown in Figure 6.2 (left).

Adversarial Attacks on Gradients from the Classifier

To measure the extent by which the decoder squeezes values to a constant pattern, we propose a second ablation experiment.

We assume that the attacker has access to the gradients from the classifier, and uses those to produce adversarial perturbations. However, attacks are always feed-forwarded through the entire ensemble. Figure 6.2 (right) shows an overview of this setup.

Gradients that come directly from the classifier should still be more robust than those of the classifier alone. Recall that the sample that first reaches the classifier is being projected by the fine-tuned autoencoder. Pixel positions that are projected to constant values by the decoder will hence generate smaller gradients than if the original image were used. However, we expect that the gradients coming from the classifier are still exposing enough information to the attacker to craft effective perturbations. Even if the autoencoder manages to dampen parts of the adversarial perturbations, these will still be effective enough to cause large drops in performance.

Compare Gradients from the Encoder and the Classifier

An important factor that helps in counteracting adversarial perturbations lies on the unsupervised pre-training of the encoder. We argued that, because the encoder is still most sensitive to reconstruction artifacts, gradients will point at structural elements in the image such as sharp edges or corners. These gradients are not directly related to the classification task and therefore, adding perturbations on those areas won't lead to strong adversarial effects.

To evaluate how different these changes are, we measure the similarity of the distribution of gradients when these are computed for different parts of the ensemble model. In particular, we extract gradients from an StSNet based on ResNet50 ($f_R \circ \mathbb{A}_R$) and compare it to gradients from ResNet50 alone, as well as to an ensemble based on a pre-trained autoencoder $f_R \circ \mathbb{A}_S$. Additionally, we compare gradients extracted from just the autoencoders $\mathbb{A}_S$ and $\mathbb{A}_R$ using the unsupervised MSE reconstruction loss.

As a similarity measure, we use the structural similarity (SSIM) proposed by Wang *et al*. [154]. This metric consists on a locally normalized mean squared error computed via a sliding window over the spatial dimensions that correspond to the image.

According to our hypotheses, we expect more similar structures from gradients produced by autoencoders, as opposed to those produced by the classifier alone. Large changes in the gradients between the classifier and ensembles that have autoencoders will suggest that an attacker is dealing with two different signals, even though they are produced by the same input.

6.3 Results

This section presents the results of all experiments described in Section 6.2. First, the robustness evaluation of the overall model is conducted, followed by three ablation experiments that help in narrowing down the sources of robustness.

6.3.1 Robustness of StSNets

We compute the accuracy curves of two ensembles based on ResNet50 and Inception-v3 with their corresponding baselines on increasingly stronger adversarial attacks.

As we can see from Figure 6.3, our proposed defense mechanism consistently mitigates the effects of all gradient-based attacks, even those with larger perturbation magnitudes. Without any defense mechanism, Inception-v3 seems slightly more robust than ResNet50 (mostly noticeable with FGSM attacks). When paired with a fine-tuned autoencoder, this tendency becomes larger, but overall, both StSNets perform substantially better than their baselines.

Results based on the ensemble $f_R \circ \mathbb{A}_S$ show that the autoencoder alone is already providing some levels of robustness against adversarial attacks. These results are in-line with observations by Meng *et al*. [100] and Liao *et al*. [87] who used specially trained

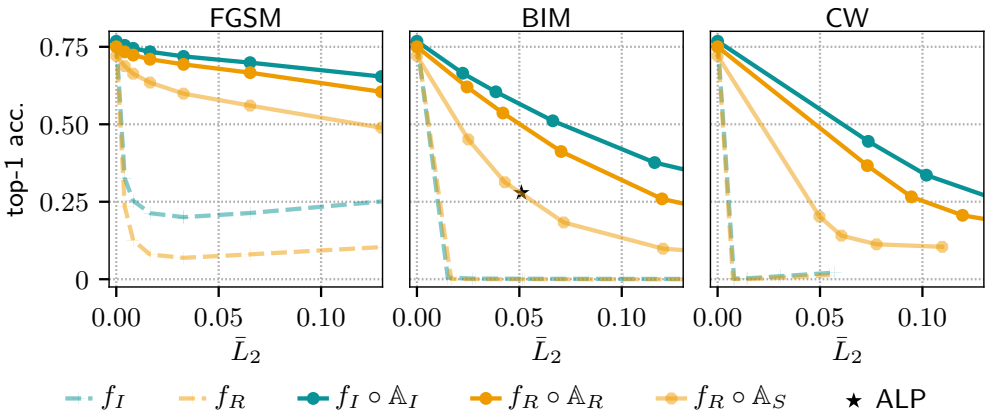


Fig. 6.3.: White-Box attacks on ResNet50 (f_R) and Inception-v3 (f_I) as baseline (dashed) and robust ensemble (solid). An ensemble with a pre-trained autoencoder $f_R \circ \mathbb{A}_S$ is also evaluated (light solid). Reported results for Adversarial logit pairing (ALP) [69] added as reference.

encoder-decoder models, albeit at a smaller scale.[3] We attribute these gains partly to the bottleneck of the autoencoder, and to the self-supervised loss function that is used to train these models.

For context, we add results for the state-of-the-art defense by Kannan *et al.* [69] on White-Box settings called Adversarial Logit Pairing (ALP). We observe that the model is on par with our $f_R \circ \mathbb{A}_S$ baseline and is comfortably outperformed by both StSNets.

These results let us also verify that the robustness of the defense does not come from obfuscated gradients. According to Athalye *et al.* [9], there are two characteristic properties[4] to spot defenses that rely on gradient obfuscation. The first sign of gradient obfuscation is when single-step attacks are more effective than iterative methods. We see that this is not the case for our experiments, with both iterative attacks (BIM and C&W) being more effective than the simpler non-iterative FGSM. A second characteristic of obfuscation is that attacks with larger budgets (in our case, increasing $\bar{L}_2$) are not progressively harder to predict. This is again something we don't see in our experiments, with model accuracy dropping as the perturbation norm increases.

6.3.2 Ablation Analysis

We proceed to compute adversarial robustness on two controlled scenarios in order to gather more evidence about factors that are making StSNets more robust.

[3] These defenses have been since shown to be weak against adaptive attacks [23, 8].

[4] The authors speak about five characteristics, but the other three refer to scenarios that do not apply to our analysis; for example the performance of Black-Box and White-Box attacks, unbounded attacks *i.e.*, values of ϵ that are arbitrarily large, and random sampling.

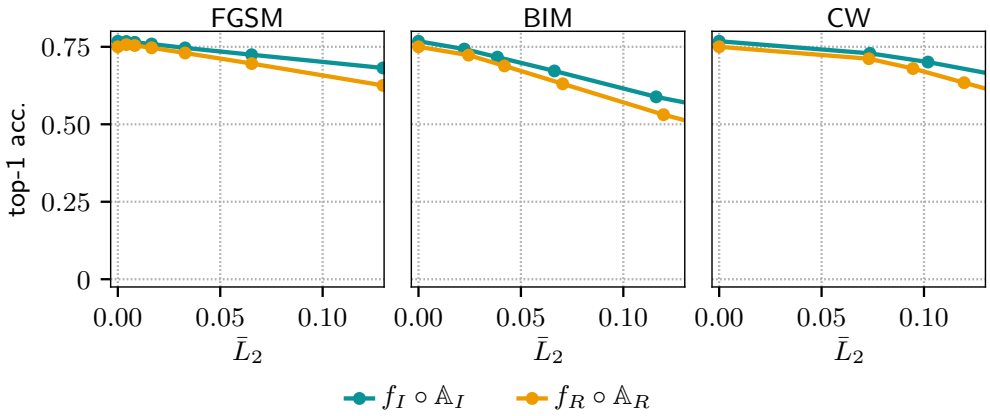


Fig. 6.4.: Ablation analysis: perturbations are computed on gradients from the ensemble StSNet, but are fed directly to the classifier.

Feed-Forward Directly into the Classifier

The first scenario is shown in Figure 6.2 (left) and consists of attacks that are fed directly into the classifier. Results of this experiment on both ResNet50 and Inception-v3 ensembles is shown in Figure 6.4.

This ablation shows increased levels of robustness across all three attacks, and for both classifiers. In contrast to our initial hypothesis, these results exceed our expectations, with results that are better than in the regular adversarial setting. This indicates that attacks are exploiting vulnerabilities of the autoencoder, and not so much those of the classifier. As part of our initial assessment, we discussed that the structure of gradients coming from autoencoders is different from gradients extracted directly from the classifier. With this experiment, we find evidence suggesting that this is indeed the case, and that the unsupervised loss used for pre-training helps in deflecting adversarial attacks.

Adversarial Attacks on Gradients from the Classifier

As our second ablation experiment, we compute attacks based on gradients from the classifier, while forwarding samples through the entire ensemble, as show in Figure 6.2 (right).

As expected, this scenario yields adversarial attacks that are almost as effective as they are on the classifiers alone. There are two forces at play in this case: on one side, we have reconstructed images $\mathbb{A}(\mathbf{x})$ with some of the original values being fixed during decoding. Any perturbation introduced at these positions will have less of an impact because the perturbed image will pass through the autoencoder during evaluation. This

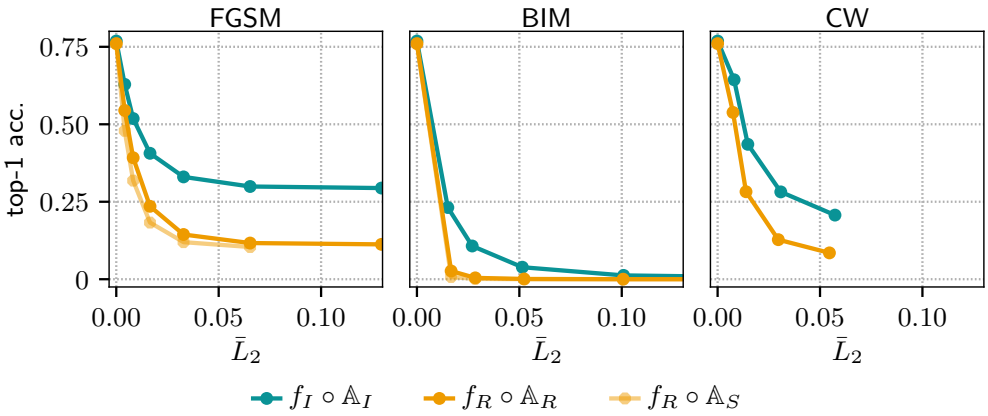


Fig. 6.5.: Ablation analysis: perturbations computed on gradients from the classifier, and evaluated on the ensemble $f_i \circ \mathbb{A}_i$.

phenomenon alone can already justify the slight improvement in performance w.r.t. the baseline from Figure 6.3.

One aspect that is different, is the structure of the gradients that the attacker receives. Through input gradients, the classifier can still reveal information about places that the autoencoder is good at reconstructing such as blobs, solid colors, or low frequency artifacts. Perturbations around these areas can pass through, and get reconstructed by the fine-tuned autoencoder forcing a wrong prediction on the side of the classifier.

Compare Gradients from the Encoder and the Classifier

The last ablation analysis focuses on the different structures between gradients produced by either autoencoders or classifiers. For this, we compare the structural similarity between gradients from a ResNet50 baseline and several other autoencoder-based models, as described in Section 6.2.3. Note that SSIM is symmetric, and thus we only show values of the upper triangular side for clarity.

Tab. 6.1.: Pairwise mean SSIM of input gradient magnitudes for ResNet50 (f_R) with and without being passed through different autoencoders. SSIM values of f_R are the least similar to any autoencoder-based ensemble.

SSIM	f_R	$f_R \circ \mathbb{A}_R$	$f_R \circ \mathbb{A}_S$	$\mathbb{A}_R$	$\mathbb{A}_S$
f_R	1.00	0.17	0.18	0.12	0.14
$f_R \circ \mathbb{A}_R$	—	1.00	0.40	0.46	0.32
$f_R \circ \mathbb{A}_S$	—	—	1.00	0.37	0.36
$\mathbb{A}_R$	—	—	—	1.00	0.36
$\mathbb{A}_S$	—	—	—	—	1.00

Results from the structural similarity support our initial presumption about the nature of gradients that can be computed through autoencoders. In Table 6.1, we see pairwise comparisons between four ensembles based on autoencoders and the classifier alone. The first row shows that there is a clear drop in structural similarity between gradients from the classifier alone and any other combination of autoencoder-based ensembles. We can conclude from these results that gradients coming from autoencoders are different to those given by a vulnerable classifier. The structure of gradients from fine-tuned autoencoders are even more similar to those produced by regular autoencoders despite being computed on entirely different losses!

Figure 6.6 shows gradients from an image that is passed through all ensembles evaluated at this point. We can visually confirm that gradients from autoencoder-variants are higher around the outline of the object. In contrast, gradients from the classifier alone are more scattered all over the image.

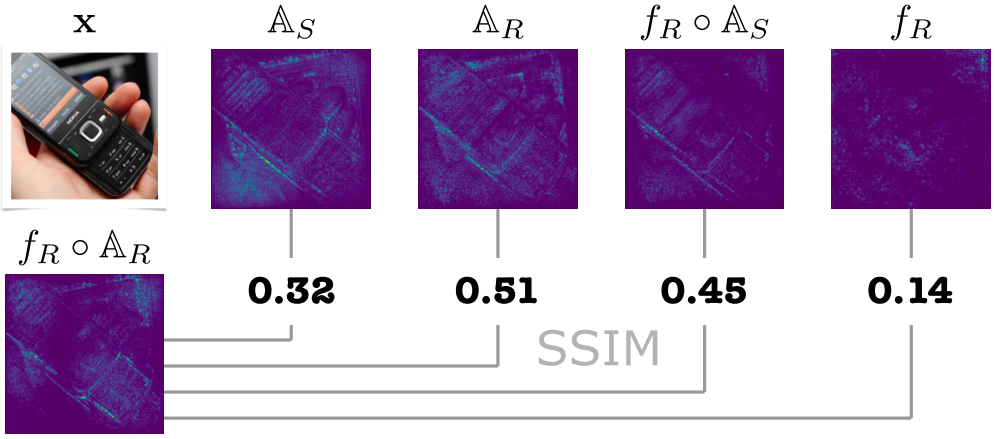


Fig. 6.6.: Visual evaluation of gradient distribution for ensembles based on ResNet50. Structural similarity (SSIM) is high among ensembles that are based on autoencoders, and smallest w.r.t. the classifier alone.

6.4 Conclusions

After conducting a wide array of experiments, we can conclude that fine-tuned autoencoders are an effective mechanism to increase robustness against gradient-based adversarial attacks. Thanks to our ablation analysis, we were able to identify two contributing factors to the emerging adversarial resilience. First, fine-tuned autoencoders project reconstructions into a lower-dimensional manifold, preventing some vulnerable pixel positions to be exploited. More importantly, the unsupervised pre-training of the autoencoders makes input gradients sparser, and more focused on structural parts of

the image. These gradients cannot be used effectively to craft adversarial attacks, in comparison to gradients coming directly from the classifier. We can also conclude that the defense mechanism does not exhibit signs of typical gradient obfuscation tactics.

This ensemble has also several advantages over other defense mechanisms, in particular over adversarial training. Most notably, StSNets can be used to protect pre-trained models, something that is not possible with adversarial training. Moreover, because fine-tuned autoencoders don't affect the original performance, StSNets don't make compromises between robustness and accuracy.

6.4.1 Future Work

These experiments originated as a case study for findings that came from the field of XAI. However, we see adversarial attacks as a tool that can contribute back to XAI, shedding light on unresolved issues about the behavior of modern ML models. In regard to StSNets, we are interested in exploring the impact that the structure of the autoencoder can have when reconstructing samples. In particular, we want to study factors that could increase model robustness like the size of the bottleneck or the information contained in the reconstruction.

We see these ideas as manifestations of the curse of dimensionality [13], and we would like to explore ways to compress reconstructions even further. For example, Resizer Networks, as proposed by Talebi and Milanfar[145] are promising architectures that can manipulate the size of reconstructions without affecting performance.

In Summary

In this chapter, we had the opportunity to use findings regarding model capacity from Chapter 5, and exploit them to increase the robustness of image classifiers. In particular, we made use of fine-tuned autoencoders as filters to prevent adversarial attacks from reaching the classifier. Robustness was evaluated for two high-performance classifiers, and against three popular gradient-based attacks. In all scenarios, our proposed defense strategy yielded robust results that even outperform alternative defenses. We were able to show that our results are not caused by trivial gradient obfuscations, and that using our method does not suppose a compromise in performance.

As the last contribution of this work, we move away from analyzing existing models and tackle the challenge of designing architectures that are more explainable by design.

7 Ante hoc Explanations with Self-Supervised Auxiliary Objectives

> *. . . AI research is a way of thinking about thinking that forces you to be specific. It calls your bluff if you think you understand thinking, but don't.*
>
> **— David Chapman**

So far, we have navigated the world of existing models, discovering ways to measure and understand some of their characteristics. Moving forward, we have the opportunity to come up with novel architectures that fulfill the need for more "interpretable" models. But how can we achieve this?

This chapter explores a systematic way to build models that are more interpretable without requiring additional data. We show that sticking to design principles that enrich traditional models with more structure, results in methods that converge faster, perform better, and whose predictions are easier to justify.

7.1 Introduction

What exactly makes a model more interpretable? At first, this question may seem too generic to be answered in any actionable way. Luckily for us, we have already defined a precise vocabulary to talk about XAI in Chapter 3 that we can use to us deconstruct this question further.

According to our XAI framework, the models we are studying are just representations of an oversimplified problem that originated in a high-level, semantic space. In this context,

explanations emerge as a way to corroborate if the model is indeed representing the non-functional requirements of the problem. With these two ideas in mind, a model can become more interpretable if it can directly convey information about non-functional requirements, without any additional steps. For instance, we could think of an image classifier that is trained on images of horses and fish, but also learning about the different backgrounds that these two classes can have: green grass, blue water, the beach, etc. During prediction, such a classifier can produce two outputs that indicate what object and what background has been identified. This result can be directly used to verify whether the model is predicting the class because of the foreground or the background.

One logical step towards creating interpretable models would be to learn the features of these additional requirements, using the same algorithms from the main model: defining a supervised loss, training a differentiable module using gradient descent, etc. While this is indeed an ideal option, it is certainly not a viable one. A critical bottleneck in this case is finding a suitable set of labeled data. However, it is not viable to depend on more and more annotations every time we want to include a new non-functional requirement. Therefore, a more attainable goal is to find ways in which non-functional requirements can be represented without resorting to new annotations.

What kind of additional objectives can be represented without more labels? While it is true that not every aspect of a problem can be represented without relying on additional ground-truth, there are still numerous strategies to exploit existing data.

One way is to use the training set as a reference basis, and offer example-based explanations for unseen samples. Networks proposed by Hase *et al*. [57] or Chen *et al*. [27] include layers that model characteristic parts of the classes they are representing. Activations of these layers can be directly interpreted as evidence for a prototypical part of an object. Alternatively, we can also find samples in the training dataset that are prototypical or exceptional w.r.t. the input distribution. Kim *et al*. [71] showed that this preselection can be used by a model to explain how typical or rare a test sample is (apart from predicting which class it belongs to).

Several training objectives that don't need labels have been proposed in the field of self-supervised learning (SSL). These functions exploit the relationships that each sample has with itself. For instance, Doersch *et al*. [39] extract a grid of patches from one image and train a model to predict the relative position between two of the patches. Noroozi *et al*. [111] extended this idea by predicting the correct order of an entire grid, similar to solving a jigsaw puzzle. Another popular approach has been to generate two versions of the same image by changing the illumination, scale, rotation, hue, etc., and then train a model to generate similar features for both [29, 58, 54, 30]. In essence, the

non-functional requirement behind these approaches is that a sample is more similar to itself than to any other.

If the dataset has annotations, these can also be exploited to express additional non-functional requirements. By knowing what class a sample belongs to, Deng *et al*. [36] constructed a hierarchical exclusion graph that conveys the idea of exclusivity to refine predictions: if a sample belongs to one class, then it doesn't belong to any other. More recently, Khosla *et al*. [70] have exploited labeled data to expand a self-supervised objective that applies to samples of the same class: a sample is more similar to both itself and to samples of the same class, but not to other samples.

Unfortunately, SSL methods suffer from a few drawbacks that make them unsuitable as interpretable objectives. One of them, is that SSL objectives are mostly used as a pre-training mechanism. Models that were trained using SSL, are later fine-tuned on a supervised task, without any guarantee that the structure of the non-functional requirement is being preserved. Furthermore, it is not clear how some SSL objectives (*e.g*., jigsaw puzzles or self-similarity) can be interpreted to support a supervised task.

We draw inspiration from these approaches, and propose an objective that expresses a different kind of non-functional requirement. Ideally, we want to leverage the effectiveness and flexibility of SSL objectives, and the interpretability of prototype-based models. In this chapter, we study relations of subsumption and grouping that emerges from a set of independently labeled data. Modeling groups allows a classifier to learn the property of subsumption; something that has already been used successfully to generate explanations for formal verification systems [98]. We show that this kind of relationship is useful to convey additional meaning for state-of-the-art ML models, adding more interpretable structure to their output.

7.1.1 Finding Structure in Groups

We begin with a simple observation: given a set of apples, and a set of cherries, we can create a larger set by simply joining all the apples and cherries together. Given that each of the original sets had its own property, the resulting union now represents a broader concept, which can be expressed as the union of the two original concepts. This exercise has also left us with a hierarchical structure where the larger set *subsumes* the two original ones.

This sort of hierarchy is usually present in ontologies or knowledge graphs, where a general concept *e.g*., "fruit" subsumes the more specific ones like "apple" and "cherry". In this case, the relation between specific and general concepts is semantic (*e.g*., apples and

cherries come from plants). However, we can create this kind of hierarchical relationship based on other criteria that doesn't rely on additional knowledge. Concretely, we want to study groups of classes that have similar visual features.

How can we transfer this idea to ML and neural networks? We know that if a model can correctly predict the class HORSE for a test sample, the same model should also be able to succeed if the sample belonged to a set of classes that *includes* horses. This also means that, to some extent, training with a group of classes that includes horses should be beneficial to train a horse classifier. To test this, we first need to group labels that define new, more generic classes (*i.e.*, group classes) that can also be included during training. Once we have these groups, we need a suitable architecture that can be trained with both the original labels and the newly formed group classes.

Since the additional group labels serve a supporting role in solving the original task, we can model the classification problem defined by the group annotations as *auxiliary tasks*. These auxiliary tasks can then be solved in parallel with the original classification problem as if they were independent of each other. However, as we know that both main and auxiliary tasks share a strong semantic link, we can make them depend on each other by using a shared feature space. Because we generate auxiliary tasks without resorting to additional (external) annotations, we call this setup *Self-Supervised Auxiliary Learning* or SSAL for short. Figure 7.1 shows a toy representation of the proposed setup.

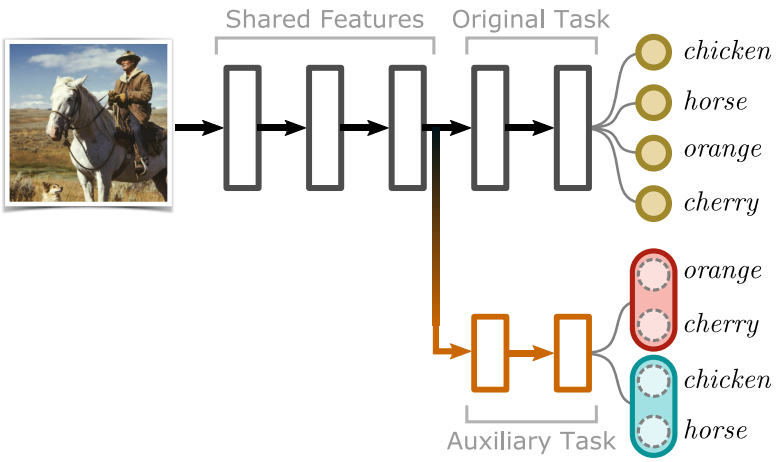


Fig. 7.1.: Overview of SSAL models. Starting from a shared feature space, a supervised task and an auxiliary branch are used to train and predict. The auxiliary task stems from the original labels following a mutually exclusive grouping.

The rest of this chapter will look into how can we find groups and adapt them to train a neural network. The next section describes the grouping criterion and architectural design of the model. Moreover, we identify a set of hyperparameters that emerge from

this setting and explore their impact. Finally, we evaluate the new model on three different image classification benchmarks and analyze the benefits that its structured output has in terms of interpretability.

7.2 Methods

This section describes the methods we use for generating groups of classes, the model architecture that supports the use of group classes, hyperparameters, and benchmarks.

7.2.1 Grouping Criterion

Our goal for the grouping criterion is two-fold. First, we want to have groups that have a similar size *i.e.*, each group has (whenever possible) the same number of labels. Secondly, we want labels within a group to share as many visual features as possible. More formally, given a set of k labels $\mathcal{Y}_k = \{y_1, y_2, \dots y_k\}$ we need to create a partition $\mathcal{Y}_G$ that comprises G non-overlapping subsets of $\mathcal{Y}_k$.

To measure similarity between the original labels, we evaluate a pre-trained model on the validation set of the data whose labels we want to group. Based on the results of the evaluation, we construct a $k \times k$ confusion matrix C where entries $c_{i,j}$ correspond to the number of samples with ground-truth $y_i \in \mathcal{Y}_k$ that got classified as $y_j \in \mathcal{Y}_k$. This matrix already gives us an initial idea of which labels are more similar to each other. The intuition is that visually similar classes will be confused more often, than labels that don't share too many visual features.

However, values in the confusion matrix cannot be yet interpreted as a measure of similarity. To do that, we transform values in C into a distance matrix as follows. First, we subtract the values from the diagonal in C, and then divide all elements by the sum of the remaining entries (Equation 7.1). Afterwards, we subtract the matrix of all-ones $\mathbf{1}$ from our partial result (Equation 7.2), and then we turn it into a distance score by averaging the off-diagonals (Equation 7.3).

$$\hat{D} = \frac{C - CI}{-\text{tr}(C) + \sum_{i,j} c_{i,j}} \tag{7.1}$$

$$\hat{S} = \mathbf{1} - \hat{D} \tag{7.2}$$

$$\mathcal{D}_c = \frac{1}{2}(\hat{D} + \hat{D}^T) \tag{7.3}$$

Now that we have a measure of similarity thanks to the distances in $\mathcal{D}_c$, we can proceed to group labels using a greedy clustering. Similar to Yan *et al*. [159], the clusters prioritize similarity between labels, but we add a constraint that keeps cluster sizes balanced.

This is how it works. Clustering begins by defining G empty clusters, and a new set $\mathcal{G}$ with a copy of all labels in $\mathcal{Y}_G$. Next, the label in $\mathcal{G}$ with the highest average distance to all other labels is removed from $\mathcal{G}$, and assigned to an empty cluster. Once there are no empty clusters, the element in $\mathcal{G}$ with the smallest average distance to a cluster of less than k/g labels is removed from $\mathcal{G}$ and assigned to that cluster. In case of a tied metric (*i.e*., the label can be assigned to more than one cluster), the label is assigned at random to one of the clusters involved in the tie. Clustering concludes when there are no more elements left in $\mathcal{G}$.

Note that this algorithm can be easily adjusted to produce the opposite result *i.e*., generate clusters with visually distant labels. This can be achieved by skipping the inversion of the normalized confusion scores in Equation 7.2.

The output of the clustering algorithm is a surjective mapping $\gamma : \mathcal{Y}_k \rightarrow \mathcal{Y}_G$ assigning a unique group class to each of the original ground-truth labels. This way, each sample in a labeled dataset becomes as a triplet $(\mathbf{x}, y, \gamma(y))$ that represents the input sample, the original label from $\mathcal{Y}_k$, and the group label from $\mathcal{Y}_G$.

For this clustering method, there is one important hyperparameter that can influence its output, namely the number of groups G that are used for clustering *i.e*., the number of elements in $\mathcal{Y}_G$. We consider values in the range $2 \leq G \leq k/2$ which guarantees that there will be no empty clusters, and that each cluster has at least two associated labels.[1]

When we consider only the pairs $(\mathbf{x}, \gamma(y))$ in a dataset, we are talking about the *auxiliary task* of the set. In contrast, tuples of the form $(\mathbf{x}, y)$ are referred to as the *original task*. Note that, in isolation, the auxiliary task can be treated as a supervised problem, and hence can be tackled by using the same methods that solve the original task.

7.2.2 Model Architecture

Once we have produced an auxiliary task, we need an architecture that can use both sets of labels to train and predict. A flexible and well-known strategy is to rely on designs used for multitask learning (MTL). The main idea of MTL is to design models that can

[1] Assuming there are four or more classes in the original task.

somehow share what they learn while being trained in parallel. Even though these ideas were already around during the early 90s, it was Caruana [25] who first showed that neural networks were also suitable models for MTL.

Our proposed model follows the structure of a hard parameter-sharing architecture with three components, as shown in Figure 7.1. The model starts with a low-level feature extractor, whose output is shared between all modules that come after. Next, common features from the shared space are passed on to branches with different objectives. One branch is the original classification task with labels from $\mathcal{Y}_k$ and denoted as $f(\cdot)$. The second branch corresponds to the auxiliary task that is based on the group classes $\mathcal{Y}_G$ and denoted as $g(\cdot)$. During prediction, the ensemble produces two outputs: one for the original classification target $f(\mathbf{x}) = \hat{y}$, where $\hat{y} \in \mathcal{Y}_k$, and another for the auxiliary task $g(\mathbf{x}) = \bar{y}$ where $\bar{y} \in \mathcal{Y}_G$.

Note that this architecture is not limited to having exactly one auxiliary branch. The clustering method from Section 7.2.1 can be run several times with a different number of groups, producing multiple mappings from $\mathcal{Y}_k$ into different partitions $\mathcal{Y}_{G1}, \mathcal{Y}_{G2}$, etc. In that case, we refer to a set of different groupings $\gamma_1, \gamma_2, \ldots$, all based on the original labels $\mathcal{Y}_k$. Every set of group classes γ_i will have a dedicated auxiliary branch $g_i(\cdot)$ in the ensemble. Correspondingly, each sample in the dataset gets represented by the tuple $(\mathbf{x}, y, \gamma_1(y), \gamma_2(y), \ldots)$.

For the implementation details, we base the architecture of the ensemble on a high-performance classifier like ResNet50 [59], WRNs [162], SENets [63], or DenseNets [64]. The shared feature extractor consists on the first b blocks of the architecture[2], and the remaining parts of the model correspond to the branch that solves the original task. The auxiliary branch also starts after the first b blocks of the model, and its precise architecture will depend on hyperparameters like the number of group classes or the point of attachment. For ensembles with more than one auxiliary branch, b corresponds to the earliest attachment point among all branches. Subsequent branches are always attached to the part of the model that solves the original task *i.e.*, there are no auxiliary branches that attach to other auxiliary branches.

As we can see, one important hyperparameter of this ensemble is the value of b *i.e.*, the point at which the (first) auxiliary classifier attaches to the original model. An early point of attachment allows branches to process generic, lower-level features, but leaves less shared parameters for all tasks to align and regularize each other. The second hyperparameter of this setup is the number of auxiliary branches that can be attached, and the point in the architecture at which they are placed.

[2]Blocks consist on a series convolution, batch-normalization and non-linear operations chained together; depending on the depth of the network, these blocks change in size.

7.2.3 Model Training

As the definition of MTL indicates, training of the main, and auxiliary tasks happens in parallel. For an ensemble with multiple branches, each sample that is forwarded will generate multiple outputs: one for the main task $f(\mathbf{x})$, and one for each auxiliary branch $g_i(\mathbf{x})$. Since annotations for each branch are available, we can use an independent classification loss function for each branch. They can be conveniently expressed together as a sum of individual losses and hence, the entire network can be updated simultaneously, including the shared parameters. The total loss $\mathcal{L}_T$ for any given sample is given by:

$$\mathcal{L}_T = \lambda_f \mathcal{L}(f(\mathbf{x}), y) + \sum_i \lambda_i \mathcal{L}(g_i(\mathbf{x}), \gamma_i(y)) \tag{7.4}$$

where $\mathcal{L}$ corresponds to the cross-entropy, and the coefficients λ_f and λ_i control the relevance of each individual loss term.

7.2.4 Model Prediction

Multitask Learning does not directly stipulate how to join all predictions once the training phase has completed. This is often not a problem, as tasks are not directly related, and can be ignored during evaluation.

For our ensemble, we are interested in combining predictions from all branches into a final joint prediction. To this end, we consider two ways in which we can achieve this:

Joint Probability: the final prediction is represented as the joint probability of the original prediction and the auxiliary classifier, such that:

$$P(y|\mathbf{x}) = \mathit{softmax}(f(\mathbf{x})_i \cdot g_{\gamma(i)}(\mathbf{x})) \tag{7.5}$$

where $f(\mathbf{x})_i$ represents the i-th output dimension of $f(\mathbf{x})$, $g_{\gamma(i)}(\mathbf{x})$ is the output dimension of the auxiliary branch associated with the original label at $f(\cdot)_i$ and $\cdot$ represents a scalar product. When more than one auxiliary classifier is used, the output of each auxiliary branches g_i is raised to a power $\eta \in (0, 1]$.

Learned Linear Combination: Predictions from $f(\mathbf{x})$ and $g_i(\mathbf{x})$ are concatenated, and then used to train a linear classifier whose output layer corresponds to the number of labels of the original task. All branches (f and all g_i) are assumed to be already trained, and the linear classifier is hence trained in a secondary step.

Baseline: We compare these methods with a naive baseline where the auxiliary branches are discarded, and only the prediction of $f(\mathbf{x})$ is evaluated. Comparing improvements w.r.t. this baseline will help us establish the influence that auxiliary classifiers have for solving the original task.

7.2.5 Ensemble Hyperparameters

Through the last three sections, we have identified hyperparameters that play a critical role in the final performance of the model. To evaluate them in isolation, we define a series of small-scale experiments, before moving on to evaluations on larger models and large-scale datasets.

We begin by examining the number of clusters that can be used to generate an auxiliary task, as well as the position at which the auxiliary branch is placed w.r.t. the original model. In order to isolate the elements that are responsible for improvements in performance, we conduct an ablation analysis on the optimization loss, prediction method and model size.

Number of Groups

To assess the influence of using a branch with more or less group classes, we train an ensemble based on ResNet18 [59] with one auxiliary branch attached before the first residual block. We evaluate the accuracy of the ensemble when the auxiliary branch has 2, 4, 10, or 20 groups. We use two datasets with different number of classes: CIFAR100 (100 classes) and TinyImageNet (200 classes). Ensembles using CIFAR100 are trained for 20 epochs, while those using TinyImageNet are trained for 30.

The architecture of the auxiliary classifier consists of four convolutional layers with batch-normalization and ReLU activations, a global average pooling, two fully connected layers, and a final linear combination with softmax normalization. The size and number of convolutional filters, as well as the number of fully connected neurons were determined via hyperparameter search.[3] Predictions are based on the joint probability described in Section 7.2.4.

[3] Optimization of the architectural hyperparameters is evaluated on a small portion of the training set (10–15%).

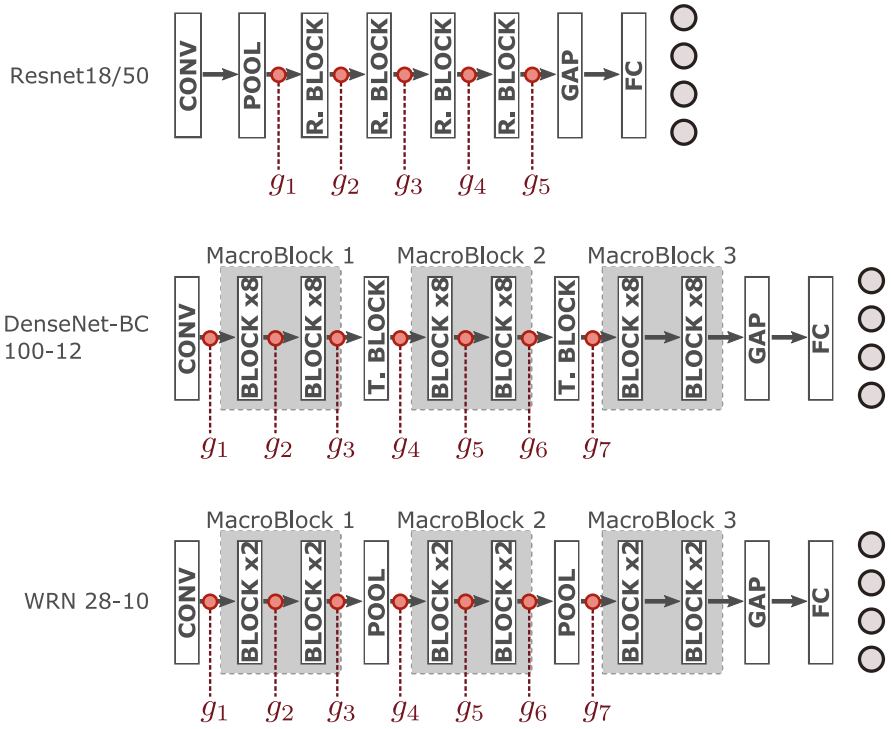


Fig. 7.2.: Positions where SSAL branches are being attached to the main network (red circles).

Branch Placement

Another important parameter of auxiliary branches is the point where it attaches along the original model. In this case, we test three deeper models as the base architecture, namely ResNet50, a Wide Residual Network (WRN 20-10) and a DenseNet (BC 100-12). All three have a similar macro architecture with well-defined blocks, giving us ample room to attach an auxiliary branch. We train and evaluate these networks on CIFAR100 while an auxiliary branch is attached at progressively deeper layers. These points of attachment include paths that lie before, after and in-between macro-blocks, as shown in Figure 7.2. The auxiliary branch has an output layer corresponding to 20 groups, and the final prediction is computed using a joint probability.

Number of Branches

We evaluate the influence of having more than one auxiliary branch as part of the MTL ensemble. For this, we train a ResNet18 on CIFAR100 with either one or two auxiliary classifiers g_1, g_2 simultaneously attached. Both auxiliary classifiers consist of two convolutional layers with batch-normalization and ReLUs, followed by an inception-like layer, global average pooling, and a linear output layer with softmax normalization.

The auxiliary branch g_1 has 20 groups, and it is attached after the first residual block of the main network. The second branch g_2 has 50 groups, and it is placed after the second residual block. Predictions of the ensemble are based on the joint probability, and a normalizing power η is set to a constant value for all branches.

We repeat this experiment with two more networks, namely ResNet50 and WRN 20-10. For these two variants, we use up to three auxiliary branches that simultaneously attach to the main architecture. The number of groups for each of the three branches is 20, 30 and 50, and their placement corresponds to the points g_1, g_2, g_3 in Figure 7.2.

Ablation Baselines

SSAL ensembles add several components to an already high-performance neural network. To understand what modules account for improvements in accuracy, we propose a series of ablation experiments. In particular, we run tests that measure the influence of the training loss, prediction method, and network size on the accuracy of the ensemble.

The use of auxiliary branches introduces additional representational capacity through the extra trainable parameters of the auxiliary branches. We test whether performance improvements can be simply be explained by the extra weights, or if the SSAL objective has merits of its own.

To this end, we train modified versions of ResNet18 on TinyImageNet that add more weights in various ways, matching or surpassing the number of parameters of a SSAL model. Moreover, we compare models that have the same architectural design of SSAL models, but are trained on a different objective.

- **WideResNet18**: has 50% more filters across all convolutional layers.
- **DeepResNet18**: adds four convolutional layers of 256 filters each with batch-normalization and ReLU activations before the first residual block.
- **DWResNet18**: similar to DeepResNet18 but doubling the number of filters of the additional convolutional layers.
- **GapCatNoSSAL**: based on a SSAL model with one branch but without the SSAL loss. The output of the auxiliary branch is concatenated with the GAP activation of the main classifier.
- **CatFCNoSSAL**: based on the SSAL architecture but without the SSAL objective. The outputs of the SSAL branch and the original network are concatenated and

passed through a fully-connected layer with 2048 neurons. The result is fed into a linear classifier for the final prediction.

- **LinearComb**: replaces the joint prediction proposed for SSAL models. After training, outputs from all branches are concatenated and passed through a linear classifier that has as many outputs as the original task.

- **SSAL**: classifier ensemble proposed in this chapter. We train and evaluate two variants with one or three auxiliary branches. For the variant with one auxiliary classifier, the SSAL branch is attached after the first residual block ($g2$ in Figure 7.2). The model with three SSAL branches uses attachment points corresponding to $g1$, $g2$, and $g3$.

All models are trained for 20 epochs with a triangular learning rate peaking at epoch 8. Each experiment is repeated three times to account for the small fluctuations caused by random initializations.

7.2.6 Classification Performance

Based on the analysis of hyperparameters, we show that SSAL models can consistently attain a higher accuracy than a variety of high-performance classifiers.

CIFAR100

For CIFAR100, we train SSAL models based on ResNet50, WRN, SENet and DenseNet. For each of these original architectures, we attach three SSAL branches with 20, 33, and 50 group classes. To guarantee uniformity on the evaluation conditions, we have re-implemented all models, and trained them from scratch so that the only difference between the original performance and the SSAL variant is the proposed surrogate objective.

For each model, we include the original performance as reported in the literature (org), results of our own re-implementation (ours), and the *LinearComb* setup from Section 7.2.5 (+LC). For SSAL models, we also report the accuracy of the original classifier *i.e.*, using the SSAL branch during training but not for prediction (+TR), and the full SSAL prediction using the joint probability (+JP).

ImageNet

To test the effects of SSAL models on large scale problems, we train a ResNet50 on ImageNet (ours), and compare it with a corresponding SSAL model equipped with three auxiliary branches. Similar to the previous experiment, each branch has progressively more groups, namely 200, 334, and 500. We report validation accuracy for SSAL models that only use the main classifier for prediction (+TR), the joint prediction (+JP), and the *LinearComb* setup from Section 7.2.5 (+LC). Moreover, a *GapCatNoSSAL* baseline (GC) is added to verity whether the behavior observed in the ablation analysis is still present in this large-scale setting.

7.3 Results

This section presents the results of experiments proposed in Section 7.2. Additionally, we discuss the benefits of the model structure to justify and interpret individual predictions.

7.3.1 Number of Groups

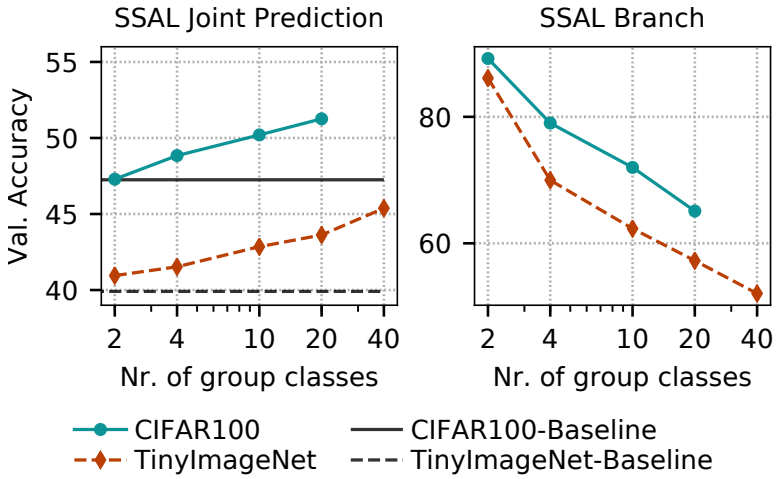


Fig. 7.3.: Influence of the number of groups on the accuracy of CIFAR100 and TinyImageNet.

We test how the accuracy varies when an auxiliary branch has an increasing number of group classes. We plot the validation accuracy on CIFAR100 and TinyImageNet for the SSAL ensemble, and for the branch alone. Results in Figure 7.3 show a constant improvement of the combined classification accuracy as the number of groups increases.

SSAL models with an auxiliary branch trained on just two groups already provides a signal that benefits the overall performance. With more group classes, auxiliary branches can represent more complex features that are more directly related to those needed to solve the main task.

On the other hand, there is a steady decrease in the raw accuracy of the auxiliary branch. This behavior is expected, as the increasing number of group classes makes the task more challenging. For a problem with a few classes, general features are already enough, and even then, there is a high probability of issuing a correct prediction at random. On the contrary, auxiliary branches have to represent more specialized features when solving a problem with more group classes. However, these features are in turn more beneficial for the original classification task.

7.3.2 Branch Placement

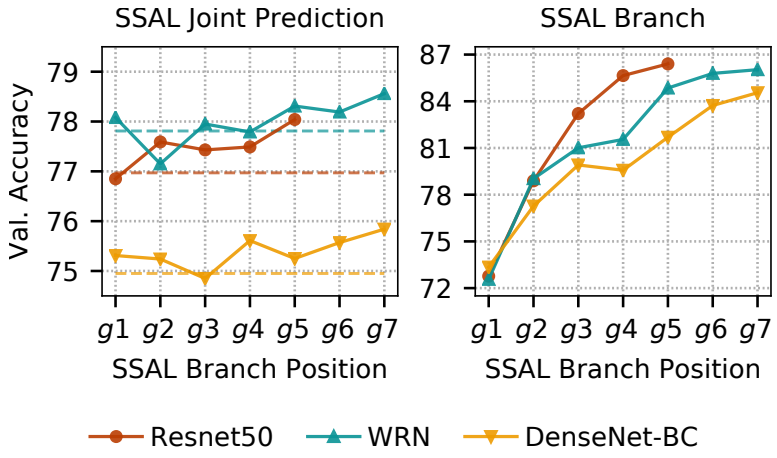


Fig. 7.4.: Accuracy when the position of the auxiliary classifier (with 20 visually similar groups) varies w.r.t. the main network.

We measure performance variations when an auxiliary branch is attached after the first few layers of the main architecture, or later. Again, we plot validation accuracy for the SSAL ensemble (left), and the performance of the auxiliary branch alone (right). Results in Figure 7.4 show that the position of the auxiliary classifier w.r.t. the main model yields better results when the auxiliary branch is attached at deeper layers. This behavior is correlated with the performance of the SSAL branch itself, which shows higher performance when it has been attached at a deeper stage within the architecture.

These results suggest that the relationship between the auxiliary branch and the model is symbiotic, as both parts of the ensemble benefit when they converge. The auxiliary

branch benefits most from having a large shared feature space, and receiving features that are already close to solving the original task.

7.3.3 Number of Branches

The number of auxiliary branches that can be attached simultaneously is another important hyperparameter we need to evaluate. For that, we measure validation accuracy on SSAL ensembles that have up to three auxiliary branches attached. Results are summarized in Table 7.1.

Tab. 7.1.: Classification accuracy for models with $||g_i||$ SSAL branches on CIFAR100. Using more branches, together with regularization, improves performance. However, the computational footprint also increases.

	$\|\|g_i\|\|$	η	Test Acc. (%)	Parameters
ResNet18	0	—	75.67	11.23M
	1	1.0	76.62	11.92M
	2	1.0	**78.23**	12.83M
ResNet50	0	—	79.13	23.77M
	1	1.0	79.70	25.07M
	3	1.0	80.36	28.89M
	3	0.3	**80.69**	28.89M
WRN 28-10	0	—	80.19	36.56M
	1	1.0	80.96	38.19M
	3	1.0	80.68	43.25M
	3	0.4	**81.08**	43.25M

As expected, increasing the number of SSAL branches has a positive impact on performance. Note that the maximum performance is achieved when the auxiliary branches have a higher regularization term η. The role of SSAL branches is to complement the decision of the main classifier, and having more auxiliary branches makes the prediction more dependent on all of them being consistent. A lower value of η makes the output of the original task the predominant signal used by the combined prediction.

7.3.4 Ablation Baselines

To test that the performance of SSAL models does not come from the additional representational capacity, but from the grouping criterion itself, we run different ablation experiments proposed in Section 7.2.5. Results from these models is reported in Table 7.2.

Tab. 7.2.: Baselines for SSAL models on TinyImageNet. Models that have a deeper architecture, wider layers, or lack the SSAL objective fail to reach the level of accuracy of SSAL models.

	Val. Acc. (%)	Diff ($p.p.$)	Parameters
ResNet18	39.9 ± 0.3	0.0	11.2M
WideResNet18	42.3 ± 0.3	2.4	25.3M
DeepResNet18	43.1 ± 0.4	3.2	13.3M
DWResNet18	43.7 ± 0.1	3.8	19.0M
GapCatNoSSAL	40.2 ± 0.3	0.3	15.6M
CatFCNoSSAL	35.3 ± 0.8	-4.6	13.6M
LinearComb	$\mathbf{44.1 \pm 0.1}$	**4.2**	12.8M
SSAL x1	$\mathbf{45.8 \pm 0.2}$	**5.9**	12.6M
SSAL x3	$\mathbf{50.0 \pm 0.4}$	**10.1**	15.6M

Looking at the first four baselines, it is clear that adding more capacity has a positive impact in accuracy. Concretely, having more capacity in the form of deeper layers shows better results than using wider layers, while a combination of both yields the highest overall improvement of up to $3.8p.p.$

As the next two baselines show, using the same architecture of a SSAL models, but without the SSAL objective (*NoSSAL), keeps or worsen performance w.r.t. the baseline. This result discards the idea that the SSAL architecture alone is responsible for the improvement in accuracy.

Finally, the last three rows of Table 7.2 show that the use of a SSAL objective yields the highest performance, even before taking the prediction method into consideration. We see that all SSAL variants outperform all other baselines with an improvement of $5.9p.p.$ to $12.7p.p.$ Note that this is already $8.9p.p.$ above the highest performing baseline (DWResNet18), all while using fewer trainable parameters. We can also verify that the use of joint predictions is consistently more beneficial than the more naive linear combination.

Overall, ablations experiments strongly indicate that SSAL models profit from the MTL architecture, the training objective, and prediction mechanism.

7.3.5 Classification Performance

Having explored the most important hyperparameters of SSAL models, we measure how much can they push the state-of-the-art performance on a small and a large-scale benchmark.

CIFAR100

Tab. 7.3.: Classification accuracy for multiple high-performance architectures on CIFAR100. Adding the SSAL objective consistently yields higher performance.

	Val. Accuracy (%)					Parameters	
	org	ours	+TR	+JP	+LC	org	SSAL
ResNet50	—	78.94	79.67	**80.61**	80.23	23.8M	28.9M
SE-WRN 16-8	**80.86**	79.02	79.02	80.20	80.03	11.1M	14.9M
WRN 28-10	80.75	80.08	80.65	**80.97**	80.68	36.6M	38.2M
DenseNet 190-40	82.82	81.06	81.85	**83.24**	83.10	26.1M	38.3M

A summary of the experiments for CIFAR100 are presented in Table 7.3. These results indicate that training with the auxiliary classifier consistently yields better performance. Looking at the (+TR) column, we see that the inductive bias of SSAL branches guides the classifier even when the auxiliary output is not used for prediction. Performance metrics improve even further when SSAL models use the auxiliary branches to issue joint predictions. Note that for WRN and DenseNet, the SSAL version even outperforms the original model (org), notwithstanding the slightly lower baseline they start from (ours).

ImageNet

We run a large-scale evaluation using ImageNet on ResNet50 to measure the extent by which SSAL improves state-of-the-art performance. Results are shown in Table 7.4. Once again, training with the SSAL objective is already improving performance even if the auxiliary branches are not used for prediction (+TR). Using a linear combination of all the SSAL branches (+LC) pushes performance slightly higher, but issuing a joint probability still works best (+JP).

Tab. 7.4.: Accuracy of SSAL models on ImageNet. Experiments are run 3 times.

	ours (org)	+TR	+JP	+LC	GC
Top-1	75.5 ± 0.1	76.4 ± 0.1	$\mathbf{76.9 \pm 0.1}$	76.6 ± 0.2	75.7 ± 0.1
Top-5	92.7 ± 0.1	93.3 ± 0.1	$\mathbf{93.7 \pm 0.1}$	93.4 ± 0.1	92.7 ± 0.1

For context, we compile results reported in the literature for methods that also convey contextual information in the loss function, use other kind of auxiliary classifiers or rely on different hierarchical priors for training (Table 7.5). We see that the benefits of SSAL lead to models that outperform the state-of-the-art in both benchmarks, CIFAR100 and ImageNet.

Tab. 7.5.: Top-1 accuracy of related state-of-the-art and SSAL models. Results for ImageNet are based on ResNet50 except the ones marked with *. Top-5 shown in parentheses, if available.

	CIFAR100	ImageNet
HD-CNN [159]	65.64	68.66 (—)*
HydraNets [147]	76.25	73.20 (—)*
COT [28]	79.46	75.60 (—)
DSL [83, 85]	81.95	76.12 (92.93)
DHM [85]	82.80	76.57 (93.24)
Aux. Train [167]	80.84	74.14 (—)*
SSAL (ours)	**83.24**	**77.00 (93.80)**

7.3.6 Interpretable Outputs

So far, we have conducted an extensive evaluation showing that SSAL models perform better than other high-performance models. From the perspective of XAI, these improvements can already be attributed to the modeling of a subsumption operation as a meaningful non-functional requirement. However, we can further exploit SSAL predictions to show that they are more interpretable than regular neural networks thanks to the auxiliary grouping criterion.

We know that the use of heatmaps has been a popular but controversial mean to interpret the output of a model. As Rudin [122] argues, heatmaps can only highlight the area that was important, but say close to nothing about the underlying features that elicit this response. We show that this gap can be addressed partly by SSAL models, thanks to the criterion used for creating the clusters that define the group labels.

Let's have a look at the samples in Figure 7.5:

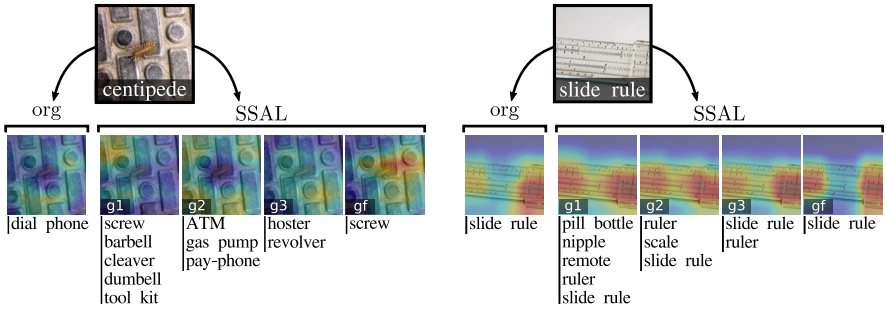


Fig. 7.5.: CAM w.r.t. each auxiliary branch g_i of the SSAL model. g_f denotes the final classification output of the SSAL model, and **org** is the CAM of a normal ResNet50. Labels within predicted auxiliary groups are shown underneath each branch.

On the left, we see an image of a centipede that has been incorrectly predicted by a vanilla ResNet50 (org) and by a SSAL variant with three auxiliary branches. Below, we show the Class Activation Maps [170] from the baseline and from all outputs of the SSAL model.

For ResNet50, we are left with a heatmap and a class prediction DIAL PHONE that cannot be justified beyond saying that, somehow, the features with high saliency resemble those of a dial phone. For the SSAL model, we not only have more saliency maps where we can look for consistencies, but we have also the labels associated with the predicted group class. We see that most groups contain labels like SCREW, REVOLVER, or TOOLKIT that share features such as metallic textures, specular reflections, and well-defined hard edges. Thus, an interpretation of the image and its salient areas can be coupled with the known, common properties that the group labels have, in order to narrow down the underlying features that cause the prediction.

Moving on to the sample on the right, we see a slide-rule that has been correctly predicted by both models. Once more, we recognize that individual labels in auxiliary groups like PILL BOTTLE or NIPPLE (*i.e.*, the mouthpiece of a baby bottle) often depict the uniform markings that characterize a ruler. These correspondences are strong indications that these markings are precisely the salient features that guided this particular prediction!

In general, we can justify the interpretation of auxiliary outputs thanks to the criterion used to generate the clusters. Since each cluster has visually similar features, an auxiliary branch that predicts a group does so because there is evidence corresponding to one of the visually similar labels.

7.4 Conclusions

After a comprehensive set of experiments, we have gathered evidence that shows an effective, systematic, and general way to create more interpretable models. We show that it is possible to add more structure to an image classifier without the need for more labeled data, by creating self-supervised auxiliary objectives based on existing annotations.

These auxiliary tasks convey the notion of category subsumption and can be implemented using architectures inspired by multitask learning. Both prediction and training can be done in parallel, with predictions being combined using a probabilistic approach with minimal overhead. An ablation analysis shows that the benefits of SSAL models don't

come from trivial changes to the architecture, and that more task-oriented features are represented thanks to the SSAL objective.

Finally, a comparison of SSAL models with high-performance classifiers demonstrates the superiority of the former for large-scale classification problems. Not only are SSAL models more accurate, but their output has a richer structure that makes individual predictions easier to interpret and justify.

7.4.1 Future Work

This work has focused on exploring one of many approaches to convey a single non-functional requirement. As part of the future work, we are interested in measuring the effects that different clustering algorithms can have for the grouping criterion. For our experiments, we have limited the clustering to non-overlapping partitions. However, visual similarities can occur at different levels, and are often asymmetric *e.g.*, tree trunks may often look like chairs, but not the other way around. At a more general level, we want to investigate other non-functional requirements beyond grouping that could also be conveyed in a self-supervised fashion.

In Summary

How can we make models more interpretable? The answer lies in the additional structure we can add to a model. As long as that structure corresponds to a non-functional requirement, we can bridge the interpretability gap and hence, get predictions that are easier to justify.

In this chapter, we have explored "grouping" as promising non-functional requirement that can even be represented without the need for additional annotations. We show how an architecture for multitask learning can be leveraged for representing hierarchical groups that comprise visually similar labels. By training and evaluating the output of the ensemble, we show that these models consistently outperform a wide array of high-performance classifiers. Most importantly, we establish how individual predictions become more interpretable thanks to the output of auxiliary branches. At the same time, the model is more interpretable, as decisions can be justified by contrasting feature attribution heatmaps with the group classes in auxiliary branches.

8 Conclusions

It's been a long way since AI sprung out of its scientific niche, and became a foundation to all kinds of industrial applications. What started as systems for automatic document layout analysis [37], or optical character recognition [81], has grown into a prosperous industry full of self-driving cars, medical diagnosis systems and satellites that help prevent natural disasters.

Advancements in ML have brought unprecedented progress to society, and we slowly start relying on machines to make high-stake decisions in our behalf. We have reached a point where our dependence on machines cannot go unquestioned. This is the point where we start making computers whose decisions we can understand.

As it turns out, peeking through the veil woven by millions of trainable parameters, and compare latent representations with our own understanding of the world, is no trivial endeavor. In this thesis, we have approached this challenge, by looking at current pressing questions in the field of XAI. We defined two main questions that helped us narrow down the cornucopia of fascinating challenges in this field.

Global Patterns of Existing Models

Our first question deals with current models; the ones that comprise the state-of-the-art, and the same ones that we struggle to explain. Can we decipher what kind of global patterns do they follow? We looked at three of those global patterns found in neural networks, and the answer is encouraging.

In Chapter 4, we studied the effects of training a VQA classifier with a dataset that suffers from a substantial overrepresentation of polar questions *i.e.*, those that are answered with either "*yes*" or "*no*". Despite the obvious challenges of training with overrepresented classes, we show that these conditions are not necessarily detrimental. In fact, we discovered that the semantic relation between polar questions and the non-polar classes is complementary. This dynamic allows a model to represent a shared feature space where non-polar classes can be answered using only features from polar questions. This kind of feature alignment improves the ability of the model to answer both polar and non-polar questions, especially for queries that refer to the same semantic concept.

Another global property that has eluded researchers for years, is the notion of model capacity. In Chapter 5, we have shown that it is possible to measure the *effective* capacity of a model by constraining the amount of information in the input space. This is possible thanks to a specially trained encoder-decoder architecture that reconstructs only the input information that is needed for classification. By measuring the information that remains after reconstruction, we get an upper bound on the amount of information that high-performance models are using. When coupled with model accuracy, these measures reveal which models use more information, and which make more effective use of that information.

These findings, it turns out, are also effective to mitigate the effects of adversarial attacks. The extensive evaluation of gradient-based adversarial perturbations in Chapter 6, shows that our proposed encoder-decoder architecture can counteract the effects of adversarial perturbations. By projecting samples to a lower dimensional manifold, and retrieving gradients from a self-supervised pre-trained encoder, our defense strategy shields vulnerable models without affecting their baseline performance.

Creating Models that are Interpretable by Design

Looking past the inscrutable nature of today's high-performance classifiers, we look at the future ahead, and entertain the possibility of starting from scratch. What can we do differently, if we had the chance to build more interpretable models from the ground up?

First, we would need to define what is meant by "*interpretable*." Taking a step back, we identify the need to have a standard, general framework to open the discourse on XAI. Chapter 3 delves into the limitations of working within the confines of a community that cannot agree on what the problem is. This chapter introduces the XAI framework in an effort to provide a standard context to compare and talk about contributions in the field of XAI. We show that the framework is grounded and precise when defining core concepts like "explanation" and "interpretation".

It is precisely by leveraging the structure of this framework that we can identify the conditions that make a model more interpretable. We show in Chapter 7 that embedding additional structure into the architecture of neural networks leads to more interpretable models. By using non-functional requirements that convey the relation of subsumption, we can define auxiliary tasks that are related to the original task, and don't require additional annotations. Training and predicting with this novel ensemble leads to results that consistently outperform state-of-the-art classifiers, all while issuing predictions that are easier to justify.

8.1 Future Work

Can we move towards more interpretable models? This work shows that it is indeed possible. In fact, results presented in these pages can be extended in a number of ways to keep moving in this direction.

Our benchmarks have been focused on classification: one of the most classic tasks in computer vision. However, more constraints and additional structure can be leveraged when considering related tasks such as object detection or instance segmentation. These are setups where the output has already a stronger semantic link with the high-level task *i.e.*, it is easier to interpret. This proximity between formal and semantic domains could play a key role in tracking more relevant non-functional requirements that can account for a model prediction.

We also have to acknowledge that, in the quest for methods that can be tracked and justified, we are implementing more inductive biases that reflect our own cognitive processes. There is still a long way before machines can reach the computational complexity and efficiency of a human brain. But in the meantime, we have to move away from the kind of oversimplified problems we are still solving today. This means defining tasks with context, using open world paradigms, and exploring efficient representations of prior knowledge that go beyond a pre-trained model.

In the meantime, we still miss a concrete set of metrics for measuring advances in XAI. How can we measure understanding? One could start by identifying a minimal set of non-functional requirements for each task, and then benchmark how consistent they are once a model has been already trained.

Are we going to reach a point where we have to abandon black-boxes altogether? Most certainly not. These are highly effective mechanisms to solve virtually any problem, as long as we can get our hands on enough training data. Many non-critical scenarios—where explainability is not required—can profit from having automatic predictions: finding what book to read next, fixing the light on an overexposed photo, etc. However, finding out more about the processes behind these models, becomes an invaluable tool to improve them, and to find new ways for solving other problems.

After all, the pace of scientific advancement will always be fueled by curiosity, and no one can stop us from asking about the things we haven't figured out yet.

Bibliography

[1] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, et al. “Neural additive models: Interpretable machine learning with neural nets”. In: *Conference on Neural Information Processing Systems* 34 (2021) (cit. on p. 2).

[2] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. “Don’t just assume; look and answer: Overcoming priors for visual question answering”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018 (cit. on p. 45).

[3] Aishwarya Agrawal, Aniruddha Kembhavi, Dhruv Batra, and Devi Parikh. “C-VQA: A compositional split of the visual question answering VQA v1.0 dataset”. In: *arXiv preprint* (2017) (cit. on p. 45).

[4] David Alvarez-Melis and Tommi S Jaakkola. “Towards robust interpretability with self-explaining neural networks”. In: *Conference on Neural Information Processing Systems*. 2018, pp. 7786–7795 (cit. on pp. 25, 26, 34).

[5] Peter Anderson, Xiaodong He, Chris Buehler, et al. “Bottom-up and top-down attention for image captioning and visual question answering”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018 (cit. on pp. 18, 47, 50).

[6] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. “Neural module networks”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016 (cit. on p. 45).

[7] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, et al. “VQA: Visual question answering”. In: *IEEE International Conference on Computer Vision*. 2015 (cit. on pp. 18, 43, 50).

[8] Anish Athalye and Nicholas Carlini. “On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses”. In: *arXiv preprint* (2018) (cit. on pp. 19, 76, 77, 86).

[9] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 274–283 (cit. on pp. 19, 25, 76, 80, 83, 86).

[10] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling”. In: *arXiv preprint* (2015) (cit. on pp. 60, 61, 80).

[11] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. “Network Dissection: Quantifying Interpretability of Deep Visual Representations”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2017 (cit. on p. 73).

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European Conference on Computer Vision*. Springer. 2006, pp. 404–417 (cit. on p. 2).

[13] Richard Ernest Bellman. *Dynamic programming*. Princeton: Princeton University Press, 1957 (cit. on p. 90).

[14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828 (cit. on p. 17).

[15] Adrien Bibal and Benoît Frénay. “Interpretability of machine learning models and representations: an introduction.” In: *European Symposium on Artificial Neural Networks*. 2016 (cit. on p. 23).

[16] Or Biran and Courtenay Cotton. “Explanation and justification in machine learning: A survey”. In: *IJCAI Workshop on Explainable AI*. Vol. 8. 2017, pp. 8–13 (cit. on p. 24).

[17] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models”. In: (2017) (cit. on p. 76).

[18] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. “Adversarial Patch”. In: *arXiv preprint* (2017) (cit. on p. 76).

[19] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. “Thermometer Encoding: One Hot Way To Resist Adversarial Examples”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 76).

[20] Brian Burke and David Smith. *Hype Cycle for Emerging Technologies, 2019*. Aug. 2019 (cit. on p. 3).

[21] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. “Explainable Machine Learning in Credit Risk Management”. In: *Computational Economics* (Sept. 2020) (cit. on p. 23).

[22] Nicholas Carlini, Anish Athalye, Nicolas Papernot, et al. “On evaluating adversarial robustness”. In: *arXiv preprint* (2019) (cit. on pp. 19, 25).

[23] Nicholas Carlini and David Wagner. “Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples”. In: *arXiv preprint* (2017) (cit. on pp. 19, 86).

[24] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *IEEE Symposium on Security and Privacy*. IEEE. 2017, pp. 39–57 (cit. on pp. 76, 82).

[25] Rich Caruana. “Multitask learning”. In: *Machine Learning Journal* 28.1 (1997) (cit. on p. 97).

[26] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. “Explaining Image Classifiers by Counterfactual Generation”. In: *International Conference on Learning Representations*. 2019 (cit. on p. 34).

[27] Chaofan Chen, Oscar Li, Daniel Tao, et al. “This looks like that: deep learning for interpretable image recognition”. In: *Conference on Neural Information Processing Systems*. 2019 (cit. on p. 92).

[28] Hao-Yun Chen, Pei-Hsin Wang, Chun-Hao Liu, et al. “Complement Objective Training”. In: *International Conference on Learning Representations*. 2019 (cit. on p. 108).

[29] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A simple framework for contrastive learning of visual representations”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1597–1607 (cit. on p. 92).

[30] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. “Big self-supervised models are strong semi-supervised learners”. In: *Conference on Neural Information Processing Systems* 33 (2020), pp. 22243–22255 (cit. on p. 92).

[31] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 2014, pp. 103–111 (cit. on pp. 10, 18).

[32] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2017, pp. 1251–1258 (cit. on p. 2).

[33] Giovanni Ciatto, Davide Calvaresi, Michael I Schumacher, and Andrea Omicini. “An Abstract Framework for Agent-Based Explanations in AI”. In: *International Conference on Autonomous Agents and Multi-Agent Systems*. 2020, pp. 1816–1818 (cit. on p. 23).

[34] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint* (2015) (cit. on p. 10).

[35] Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. “Explainable software analytics”. In: *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*. 2018, pp. 53–56 (cit. on pp. 24, 26).

[36] Jia Deng, Nan Ding, Yangqing Jia, et al. “Large-scale object classification using label relation graphs”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 48–64 (cit. on p. 93).

[37] Andreas Dengel and Gerhard Barth. “ANASTASIL: hybrid knowledge-based system for document layout analysis”. In: *International Joint Conference on Artificial Intelligence*. 1989, pp. 1249–1254 (cit. on p. 111).

[38] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, et al. “Stochastic Activation Pruning for Robust Adversarial Defense”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 76).

[39] Carl Doersch, Abhinav Gupta, and Alexei A Efros. “Unsupervised visual representation learning by context prediction”. In: *IEEE International Conference on Computer Vision*. 2015, pp. 1422–1430 (cit. on p. 92).

[40] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, et al. “Explanations can be manipulated and geometry is to blame”. In: *Conference on Neural Information Processing Systems*. 2019, pp. 13589–13600 (cit. on p. 26).

[41] Finale Doshi-Velez and Been Kim. “Considerations for evaluation and generalization in interpretable machine learning”. In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer, 2018, pp. 3–17 (cit. on pp. 24–26, 30, 34, 36).

[42] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of Machine Learning Research* 12.7 (2011) (cit. on p. 15).

[43] Patrick Esser, Robin Rombach, and Bjorn Ommer. “A Disentangling Invertible Interpretation Network for Explaining Latent Representations”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2020, pp. 9223–9232 (cit. on p. 29).

[44] Council of the European Union and European Parliament. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. European Commission, 2016 (cit. on p. 24).

[45] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. “Robustness of Classifiers: From Adversarial to Random Noise”. In: *Conference on Neural Information Processing Systems*. 2016, pp. 1632–1640 (cit. on p. 77).

[46] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. “Detecting Adversarial Samples from Artifacts”. In: *arXiv preprint* (2017) (cit. on p. 76).

[47] Amirata Ghorbani, Abubakar Abid, and James Zou. “Interpretation of neural networks is fragile”. In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3681–3688 (cit. on p. 26).

[48] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *International Conference on Artificial Intelligence and Statistics*. 2011 (cit. on p. 10).

[49] Ian Goodfellow. *Attacking Machine Learning with Adversarial Examples*. Feb. 2017 (cit. on p. 18).

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on pp. 11, 55).

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. “Generative adversarial nets”. In: *Conference on Neural Information Processing Systems* 27 (2014) (cit. on p. 60).

[52] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint* (2014) (cit. on pp. 76, 77, 81).

[53] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. “Making the V in VQA matter: Elevating the role of image understanding in visual question answering”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2017, pp. 6904–6913 (cit. on pp. 21, 43, 44).

[54] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Conference on Neural Information Processing Systems* 33 (2020), pp. 21271–21284 (cit. on p. 92).

[55] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. “Countering Adversarial Images using Input Transformations”. In: *International Conference on Learning Representations* (2018). accepted as poster (cit. on pp. 76, 80).

[56] Holger A Haenssle, Christine Fink, Roland Schneiderbauer, et al. “Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists”. In: *Annals of oncology* 29.8 (2018), pp. 1836–1842 (cit. on p. 2).

[57] Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. “Interpretable image recognition with hierarchical prototypes”. In: *AAAI Conference on Human Computation and Crowdsourcing*. Vol. 7. 2019, pp. 32–40 (cit. on p. 92).

[58] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. “Momentum contrast for unsupervised visual representation learning”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2020, pp. 9729–9738 (cit. on p. 92).

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016, pp. 770–778 (cit. on pp. 2, 62, 97, 99).

[60] Geoffrey Hinton. *Neural networks for machine learning*. 2012 (cit. on p. 15).

[61] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 10, 18).

[62] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. “A benchmark for interpretability methods in deep neural networks”. In: *Conference on Neural Information Processing Systems*. 2019, pp. 9737–9748 (cit. on p. 30).

[63] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018 (cit. on p. 97).

[64] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. “Densely connected convolutional networks”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2017 (cit. on p. 97).

[65] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456 (cit. on p. 60).

[66] John R Josephson and Susan G Josephson. *Abductive inference: Computation, philosophy, technology*. Cambridge University Press, 1996 (cit. on pp. 24, 26).

[67] Kushal Kafle and Christopher Kanan. “An analysis of visual question answering algorithms”. In: *IEEE International Conference on Computer Vision*. 2017 (cit. on pp. 43, 44).

[68] Min-Joo Kang and Je-Won Kang. “Intrusion detection system using deep neural network for in-vehicle network security”. In: *PloS one* 11.6 (2016), e0155781 (cit. on p. 23).

[69] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. “Adversarial Logit Pairing”. In: *arXiv preprint* (2018) (cit. on pp. 77, 86).

[70] Prannay Khosla, Piotr Teterwak, Chen Wang, et al. “Supervised contrastive learning”. In: *Conference on Neural Information Processing Systems* 33 (2020), pp. 18661–18673 (cit. on p. 93).

[71] Been Kim, Oluwasanmi Koyejo, Rajiv Khanna, et al. “Examples are not enough, learn to criticize! Criticism for Interpretability.” In: *Conference on Neural Information Processing Systems*. 2016, pp. 2280–2288 (cit. on p. 92).

[72] Jae Myung Kim, Junsuk Choe, Zeynep Akata, and Seong Joon Oh. “Keep CALM and Improve Visual Feature Attribution”. In: *IEEE International Conference on Computer Vision*. 2021, pp. 8350–8360 (cit. on p. 38).

[73] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on pp. 16, 50, 82).

[74] Ranjay Krishna, Yuke Zhu, Oliver Groth, et al. “Visual genome: Connecting language and vision using crowdsourced dense image annotations”. In: *International Journal of Computer Vision* 123.1 (2017) (cit. on p. 45).

[75] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009) (cit. on p. 20).

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Conference on Neural Information Processing Systems* 25 (2012) (cit. on pp. 3, 62).

[77] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112 (cit. on p. 82).

[78] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. “Faithful and customizable explanations of black box models”. In: *AAAI/ACM Conference on AI, Ethics, and Society*. 2019, pp. 131–138 (cit. on pp. 24, 26).

[79] Sebastian Lapuschkin, Alexander Binder, Gregoire Montavon, Klaus-Robert Muller, and Wojciech Samek. “Analyzing Classifiers: Fisher Vectors and Deep Neural Networks”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. June 2016 (cit. on p. 73).

[80] Matthew L Leavitt and Ari Morcos. “Towards falsifiable interpretability research”. In: *arXiv preprint* (2020) (cit. on p. 25).

[81] Yann LeCun, Bernhard Boser, John Denker, et al. “Handwritten digit recognition with a back-propagation network”. In: *Conference on Neural Information Processing Systems* 2 (1989) (cit. on p. 111).

[82] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 62).

[83] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. “Deeply-supervised nets”. In: *International Conference on Artificial Intelligence and Statistics*. 2015 (cit. on p. 108).

[84] David Lewis. *Causal explanation, philosophical papers, Vol. 2*. 1986 (cit. on pp. 24, 26).

[85] Duo Li and Qifeng Chen. "Dynamic Hierarchical Mimicking Towards Consistent Optimization Objectives". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2020 (cit. on p. 108).

[87] Fangzhou Liao, Ming Liang, Yinpeng Dong, et al. "Defense against adversarial attacks using high-level representation guided denoiser". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018, pp. 1778–1787 (cit. on pp. 76, 85).

[88] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection". In: *IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988 (cit. on p. 45).

[89] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. "Microsoft COCO: Common objects in context". In: *European Conference on Computer Vision*. Springer. 2014 (cit. on pp. 18, 21, 42).

[90] Zachary C Lipton. "The mythos of model interpretability". In: *ACM Queue* (2018) (cit. on pp. 23, 26).

[91] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into Transferable Adversarial Examples and Black-Box Attacks". In: *arXiv preprint* (2016) (cit. on p. 76).

[92] Tania Lombrozo. "The structure and function of explanations". In: *Trends in cognitive sciences* 10.10 (2006), pp. 464–470 (cit. on pp. 24, 26, 27, 32).

[93] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. "Discovering causal signals in images". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2017, pp. 6979–6987 (cit. on p. 34).

[94] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *International Conference on Learning Representations*. 2017 (cit. on p. 14).

[95] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. "Hierarchical question-image co-attention for visual question answering". In: *Conference on Neural Information Processing Systems* 29 (2016) (cit. on p. 18).

[96] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Conference on Neural Information Processing Systems*. 2017, pp. 4765–4774 (cit. on pp. 23, 24, 26).

[97] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections". In: *Conference on Neural Information Processing Systems* 29 (2016) (cit. on p. 17).

[98] Deborah L McGuinness and Alexander Borgida. "Explaining subsumption in description logics". In: *International Joint Conference on Artificial Intelligence*. 1995 (cit. on p. 93).

[99] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint* (2018) (cit. on p. 60).

[100] Dongyu Meng and Hao Chen. “Magnet: A Two-Pronged Defense Against Adversarial Examples”. In: *ACM SIGSAC conference on computer and communications security*. ACM. 2017, pp. 135–147 (cit. on pp. 76, 85).

[101] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Journal of Artificial Intelligence* 267 (2019) (cit. on pp. 23, 25, 26, 32, 34, 36).

[102] Tim Miller, Piers Howe, and Liz Sonenberg. “Explainable AI: Beware of Inmates Running the Asylum”. In: *IJCAI Workshop on Explainable AI*. 2017 (cit. on p. 23).

[103] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. “Explaining nonlinear classification decisions with deep taylor decomposition”. In: *Pattern Recognition* 65 (2017), pp. 211–222 (cit. on p. 73).

[104] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* (2018) (cit. on pp. 24, 26).

[105] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016 (cit. on p. 76).

[106] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. “Universal Adversarial Perturbations”. In: *arXiv preprint* (2017) (cit. on p. 76).

[107] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. “Robustness of Classifiers to Universal Perturbations: A Geometric Perspective”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 77).

[108] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. “In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning.” In: *ICLR Workshop*. 2015 (cit. on p. 56).

[109] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks”. In: *Conference on Neural Information Processing Systems*. 2016, pp. 3387–3395 (cit. on p. 77).

[110] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning deconvolution network for semantic segmentation”. In: *IEEE International Conference on Computer Vision*. 2015, pp. 1520–1528 (cit. on pp. 17, 60, 80).

[111] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84 (cit. on p. 92).

[112] Kasey Panetta. *5 Trends Appear On The Gartner Hype Cycle For Emerging Technologies 2019*. Aug. 2019 (cit. on p. 3).

[113] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, et al. “Practical black-box attacks against machine learning”. In: *ACM on Asia conference on computer and communications security*. 2017, pp. 506–519 (cit. on p. 76).

[114] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks”. In: *IEEE Symposium on Security and Privacy*. IEEE. 2016, pp. 582–597 (cit. on p. 76).

[115] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Empirical Methods in Natural Language Processing*. 2014, pp. 1532–1543 (cit. on p. 18).

[116] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. “Generative Adversarial Perturbations”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. June 2018 (cit. on p. 76).

[117] Rama Chellappa Pouya Samangouei Maya Kabkab. “Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models”. In: *International Conference on Learning Representations* (2018). accepted as poster (cit. on p. 76).

[118] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. “On the expressive power of deep neural networks”. In: *arXiv preprint* (2016) (cit. on p. 73).

[119] Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. “Overcoming language priors in visual question answering with adversarial regularization”. In: *Conference on Neural Information Processing Systems*. 2018 (cit. on p. 45).

[120] Gabrielle Ras, Ning Xie, Marcel van Gerven, and Derek Doran. “Explainable Deep Learning: A Field Guide for the Uninitiated”. In: *Journal of Artificial Intelligence Research* 73 (2022), pp. 329–397 (cit. on pp. 3, 26).

[121] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why should I trust you?" Explaining the predictions of any classifier”. In: *ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cit. on p. 26).

[122] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215 (cit. on pp. 25, 26, 34, 108).

[123] Cynthia Rudin and Berk Ustun. “Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice”. In: *Interfaces* 48.5 (2018), pp. 449–466 (cit. on p. 23).

[124] Olga Russakovsky, Jia Deng, Hao Su, et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015) (cit. on pp. 20, 41, 59).

[125] Deepak Babu Sam, Neeraj N. Sajjan, R. Venkatesh Babu, and Mukundhan Srinivasan. “Divide and Grow: Capturing Huge Diversity in Crowd Images With Incrementally Growing CNN”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. June 2018 (cit. on p. 56).

[126] Ute Schmid and Bettina Finzel. “Mutual Explanations for Cooperative Decision Making in Medicine”. In: *KI-Künstliche Intelligenz* (2020), pp. 1–7 (cit. on p. 24).

[127] David Schneeberger, Karl Stöger, and Andreas Holzinger. “The European legal framework for medical AI”. In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer. 2020, pp. 209–226 (cit. on p. 24).

[128] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. “Are Adversarial Examples Inevitable?” In: *International Conference on Learning Representations*. 2019 (cit. on p. 77).

[129] Maruan Al-Shedivat, Avinava Dubey, and Eric Xing. “Contextual explanation networks”. In: *Journal of Machine Learning Research* 21.194 (2020), pp. 1–44 (cit. on pp. 23, 24).

[130] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint* (2014) (cit. on pp. 60, 62).

[131] Amanpreet Singh, Vivek Natarajan, Meet Shah, et al. “Towards vqa models that can read”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2019, pp. 8317–8326 (cit. on p. 43).

[132] Aman Sinha, Hongseok Namkoong, and John Duchi. “Certifiable Distributional Robustness with Principled Adversarial Training”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 77).

[133] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. “Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods”. In: *AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186 (cit. on pp. 26, 34).

[134] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. “Content-based image retrieval at the end of the early years”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.12 (2000), pp. 1349–1380 (cit. on p. 28).

[135] Leslie N Smith. “Cyclical learning rates for training neural networks”. In: *IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2017, pp. 464–472 (cit. on p. 14).

[136] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. “PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples”. In: *International Conference on Learning Representations*. 2018 (cit. on p. 76).

[137] John F Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley Pub., Reading, MA, 1983 (cit. on p. 28).

[138] Pierre Stock and Moustapha Cisse. “Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases”. In: *European Conference on Computer Vision*. 2018, pp. 498–512 (cit. on p. 42).

[139] Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. “A corpus of natural language for visual reasoning”. In: *Annual Meeting of the Association for Computational Linguistics*. 2017, pp. 217–223 (cit. on p. 45).

[140] Alane Suhr, Stephanie Zhou, Ally Zhang, et al. “A Corpus for Reasoning About Natural Language Grounded in Photographs”. In: *Annual Meeting of the Association for Computational Linguistics*. 2019 (cit. on p. 45).

[141] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328 (cit. on p. 38).

[142] Christian Szegedy, Wei Liu, Yangqing Jia, et al. “Going deeper with convolutions”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2015 (cit. on pp. 2, 56).

[143] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016, pp. 2818–2826 (cit. on pp. 59, 62).

[144] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. "Intriguing properties of neural networks". In: *arXiv preprint* (2013) (cit. on pp. 18, 25, 75).

[145] Hossein Talebi and Peyman Milanfar. "Learning to resize images for computer vision tasks". In: *IEEE International Conference on Computer Vision*. 2021, pp. 497–506 (cit. on p. 90).

[146] Thomas Tanay and Lewis Griffin. "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples". In: *arXiv preprint* (2016) (cit. on p. 77).

[147] Ravi Teja Mullapudi, William R. Mark, Noam Shazeer, and Kayvon Fatahalian. "HydraNets: Specialized Dynamic Architectures for Efficient Inference". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018 (cit. on p. 108).

[148] Damien Teney, Peter Anderson, Xiaodong He, and Anton Van Den Hengel. "Tips and tricks for visual question answering: Learnings from the 2017 challenge". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2018 (cit. on pp. 18, 50).

[149] Bart Thomee, David A Shamma, Gerald Friedland, et al. "YFCC100M: The new data in multimedia research". In: *Communications of the ACM* 59.2 (2016), pp. 64–73 (cit. on pp. 21, 61, 80).

[150] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.11 (2008) (cit. on p. 60).

[151] Giulia Vilone and Luca Longo. "Explainable Artificial Intelligence: a Systematic Review". In: *arXiv preprint* (2020) (cit. on pp. 3, 24).

[152] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, et al. "Learning Adversary-Resistant Deep Neural Networks". In: *arXiv preprint* (2016) (cit. on p. 76).

[153] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. "Growing a Brain: Fine-Tuning by Increasing Model Capacity". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. July 2017 (cit. on p. 56).

[154] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (Apr. 2004), pp. 600–612 (cit. on p. 85).

[155] Rudolf Wille. "Restructuring lattice theory: an approach based on hierarchies of concepts". In: *Ordered sets*. Springer, 1982, pp. 445–470 (cit. on p. 71).

[156] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations*. 2018 (cit. on p. 76).

[157] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. "Feature Denoising for Improving Adversarial Robustness". In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2019, pp. 501–509 (cit. on p. 76).

[158] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. “Self-training with noisy student improves imagenet classification”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2020, pp. 10687–10698 (cit. on p. 56).

[159] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, et al. “HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition”. In: *IEEE International Conference on Computer Vision*. 2015 (cit. on pp. 96, 108).

[160] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. “Evaluating bag-of-visual-words representations in scene classification”. In: *International Workshop on Multimedia Information Retrieval*. 2007, pp. 197–206 (cit. on p. 2).

[161] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. “Deep modular co-attention networks for visual question answering”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2019, pp. 6281–6290 (cit. on p. 43).

[162] Sergey Zagoruyko and Nikos Komodakis. “Wide residual networks”. In: *arXiv preprint* (2016) (cit. on p. 97).

[163] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833 (cit. on p. 72).

[164] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding Deep Learning (Still) Requires Rethinking Generalization”. In: *Communications of the ACM* 64.3 (Feb. 2021), pp. 107–115 (cit. on pp. 57, 58).

[165] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding deep learning requires rethinking generalization”. In: *International Conference on Learning Representations*. 2017 (cit. on p. 56).

[166] Daniel Zhang, Nestor Maslej, Erik Brynjolfsson, et al. *The AI Index 2022 Annual Report*. Tech. rep. Stanford University, 2022 (cit. on pp. 2, 25).

[167] Linfeng Zhang, Muzhou Yu, Tong Chen, et al. “Auxiliary Training: Towards Accurate and Robust Models”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2020 (cit. on p. 108).

[168] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. “Yin and yang: Balancing and answering binary visual questions”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016 (cit. on pp. 42, 45).

[169] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. “Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints”. In: *Empirical Methods in Natural Language Processing*. 2017 (cit. on p. 42).

[170] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. “Learning Deep Features for Discriminative Localization.” In: *IEEE/CVF Computer Vision and Pattern Recognition Conference* (2016) (cit. on p. 109).

[171] Bing Zhu, Wenchuan Yang, Huaxuan Wang, and Yuan Yuan. “A hybrid deep learning model for consumer credit scoring”. In: *IEEE International Conference on Artificial Intelligence and Big Data*. IEEE. 2018, pp. 205–208 (cit. on p. 2).

[172] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. “Visual7w: Grounded question answering in images”. In: *IEEE/CVF Computer Vision and Pattern Recognition Conference*. 2016 (cit. on p. 45).

List of Figures

List of Tables

Appendix

A

This chapter presents some additional details on additional experiments and results discussed throughout this thesis.

A.1 Chapter 4

When evaluating the semantic link between polar and non-polar samples, we have matched non-polar classes that occur within polar questions. For this matching, we used the regular expression shown in Figure A.1:

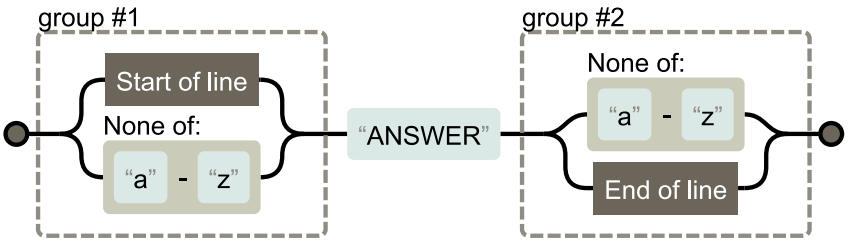


Fig. A.1.: Regular expression to match non-polar classes in polar questions. Generated with `https://regexper.com`.

Intuitively, we match strings that are not preceded nor followed by a letter, while allowing other symbols around. Table A.1 shows the top 100 non-polar classes that have been matched within polar questions.

Tab. A.1.: List of non-polar classes with most matching polar questions.

NP class	Nr. matches	NP class	Nr. matches	NP class	Nr. matches	NP class	Nr. matches
in	11936	day	733	white	536	raining	438
on	8292	girl	724	plate	532	bike	433
man	4643	toilet	717	kitchen	531	hair	430
picture	2990	sky	683	men	527	right	429
people	2469	happy	671	clean	510	clock	426
person	2199	sunny	643	sitting	506	full	425
photo	1944	plane	637	standing	501	road	421
woman	1699	bathroom	634	horse	490	healthy	415
water	1651	giraffe	628	car	483	window	414
can	1644	street	616	building	483	beach	411
s	1579	eating	603	outside	478	black	410
cat	1454	bear	601	grass	478	red	406
all	1413	both	596	real	477	old	405
train	1341	ball	587	eat	464	baby	401
room	1310	open	583	cold	461	camera	400
dog	1287	up	578	background	457	snow	400
animals	1053	ground	571	good	453	glasses	399
food	1000	looking	569	bird	452	clouds	396
animal	925	out	562	sign	451	holding	393
someone	906	table	561	light	450	flowers	387
pizza	888	trees	560	shirt	447	lights	385
bus	860	moving	557	boat	442	zoo	379
going	825	elephant	555	more	442	green	376
color	783	child	541	truck	441	down	375
boy	734	playing	540	bed	440	zebra	373

A.2 Chapter 5

In order to measure effective capacity, we have fine-tuned the decoder of a pre-trained autoencoder with gradients from different classifiers. Figure A.2 shows additional reconstruction examples based on the different fine-tuned decoders.

To get a better intuition of how constant the checkerboard artifacts are for reconstructed samples, Figure A.3 and Figure A.4 show several enlarged areas within a single sample.

When comparing classifiers, and how well they work with signals from other classifiers, we defined a threshold t for results based on the $\mathrm{RC}_{\mathrm{min}}$ metric that convey the binary notion of "working well". Figure A.5 shows the resulting FCA lattices that use different values of t.

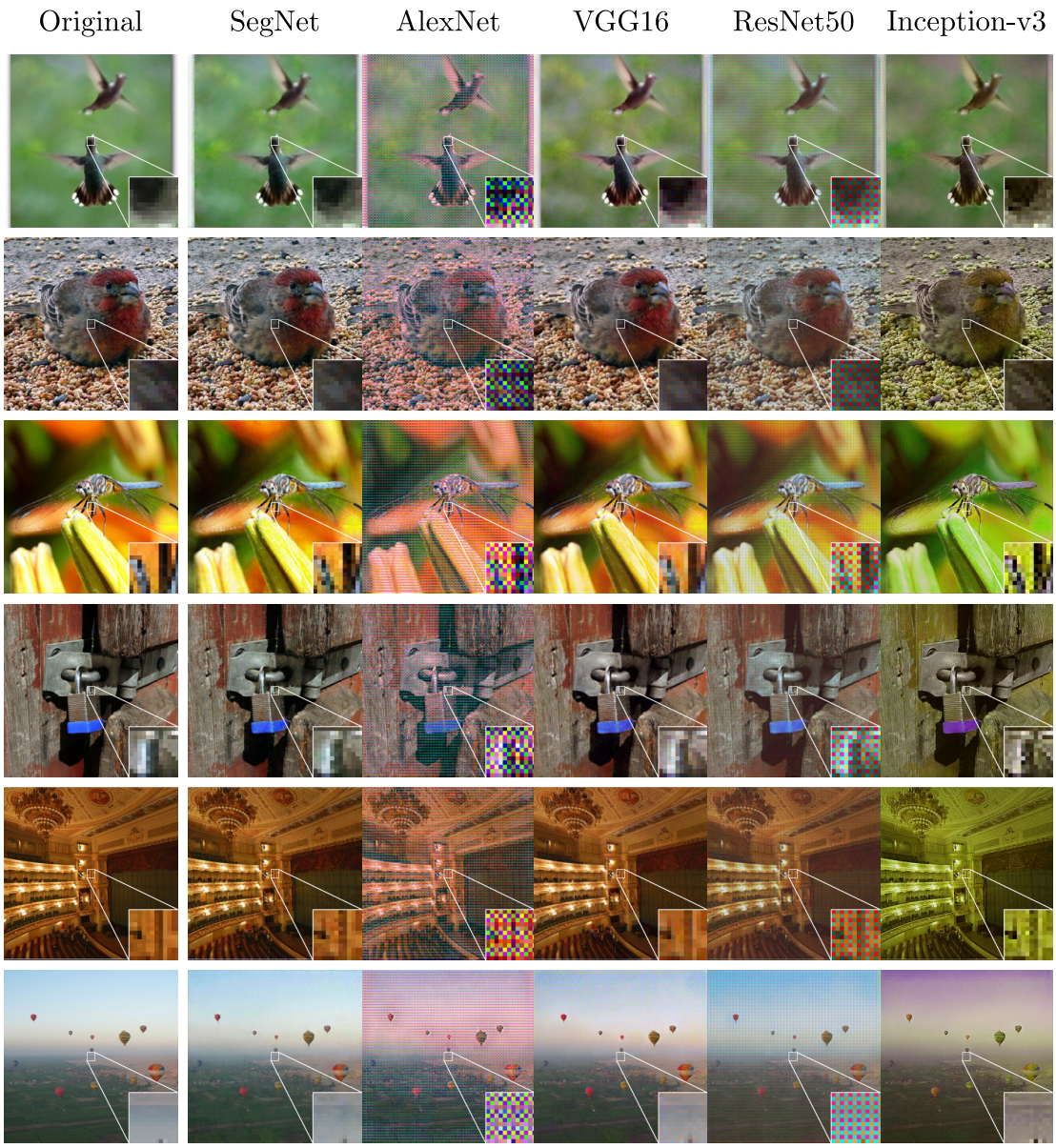


Fig. A.2.: Reconstructions of fine-tuned autoencoders.

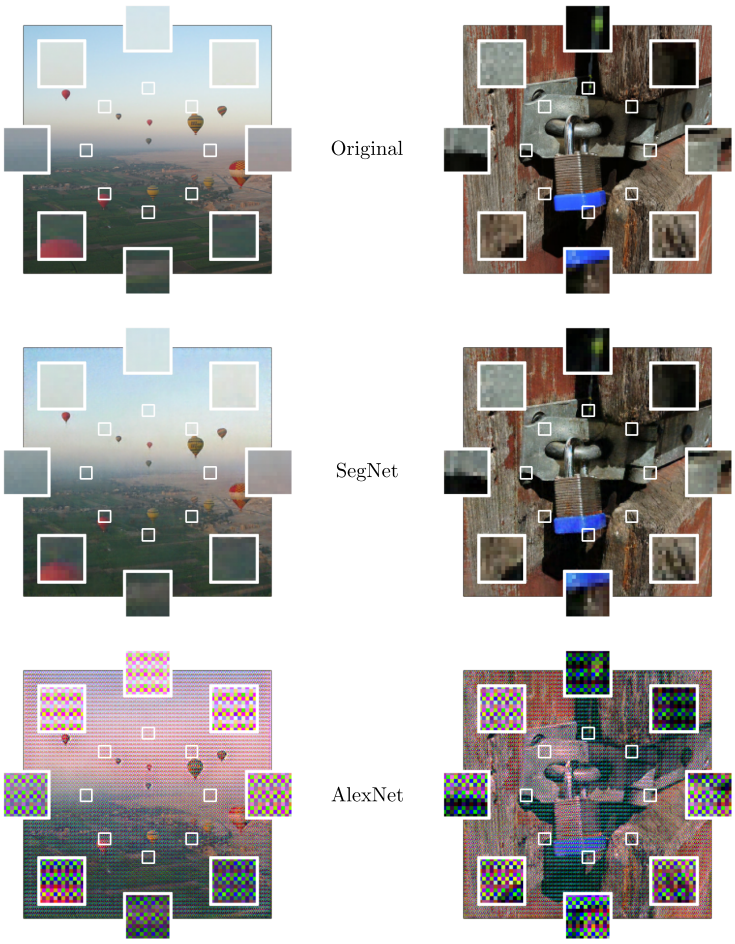


Fig. A.3.: Zooming in on different areas for the original sample, and reconstructions from SegNet and AlexNet.

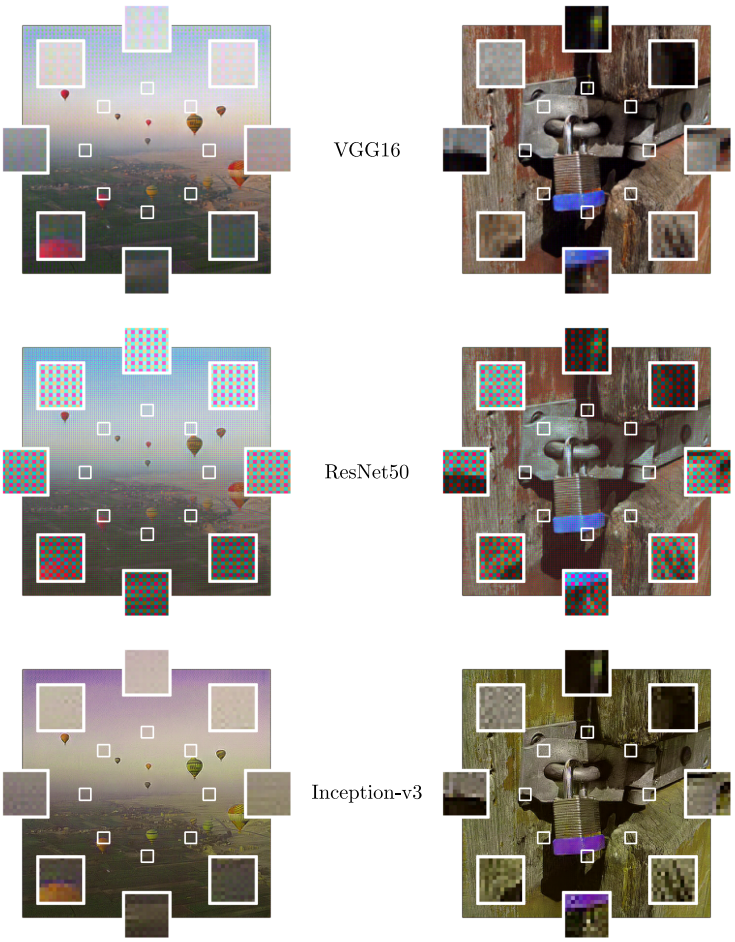


Fig. A.4.: Zooming in on different areas for reconstructions of VGG16, ResNet50 and Inception-v3.

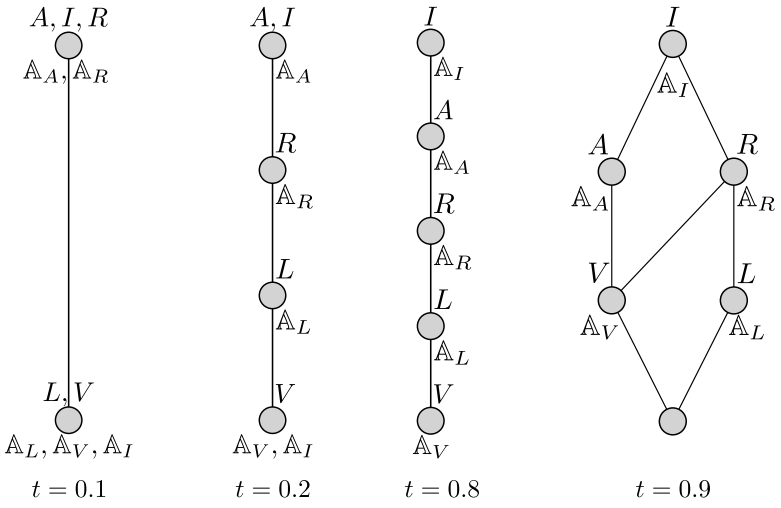


Fig. A.5.: FCA lattices based on $\mathrm{RC}_{\min}$ and different thresholds t.

A.3 Chapter 7

We can verify that the SSAL objective has an aligned inductive bias by measuring the rate of convergence when training a SSAL model. We train a baseline based on the SSAL architecture, but without a SSAL objective, similar to *CatFCNoSSAL* for 20, 50 and 100 epochs. Then, we compare these results to a model with the same architecture that uses the SSAL objective for 20 epochs worth of training. Figure A.6 (left) shows the validation accuracy of these four setups. We can clearly see that the model using the SSAL objective (in red) has achieved a higher accuracy in just 20 epochs; something that even the model trained over 100 epochs could not reach. The accelerated rate of convergence is also evident when comparing the training accuracy of a SSAL-based model against a baseline implementation with no auxiliary branches (Figure A.6 right).

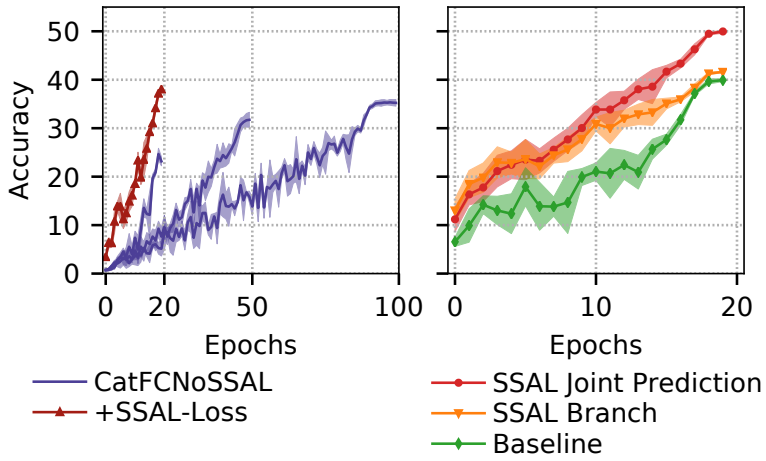


Fig. A.6.: Accuracy of a SSAL model based on Resnet18 (red) is higher than baselines using five times more training time (purple), let alone the original Resnet18 (green).

Auxiliary Architectures

Auxiliary sub-networks used to create a SSAL model vary depending on the point of attachment or the number of classes. They consist of different combinations of convolutional layers, batch-normalization and ReLU activations, Inception-like blocks, and fully connected layers. Architectural details have been determined via hyperparameter search.

Table A.2 describes auxiliary networks used for experiments on Tiny ImageNet. Table A.3 shows the architecture of SSAL branches that were added to a ResNet18 when evaluating CIFAR100. Table A.4, and A.5 present the architecture for branches attached to different

WRNs and to a DenseNet when training on CIFAR100. Finally, Table A.6 shows the architecture of the sub-networks used to train a SSAL model based on ResNet50 for ImageNet.

Tab. A.2.: Architecture of auxiliary networks for Tiny ImageNet. When attaching an auxiliary classifier to positions g_1 or g_2 of the main architecture, the AuxNet1 layout is used (Figure 7.2 in the main document). The number of output units depends on the number of groups used in the SSAL objective.

layer name	AuxNet1	AuxNet2
AuxConv1	$[3 \times 3, 128] \times 2$	—
AuxConv2	$[3 \times 3, 256] \times 2$	
GAP	Global average pool	
FC1	FC-256, ReLU	
Output	FC-200, softmax	

Tab. A.3.: Architecture of auxiliary networks on Resnet50 for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. For our main result, AuxNet1 is attached on position g_2 (see Figure 7.2 on the document), AuxNet2 on position g_3 and AuxNet3 on position g_4 of the main network. The layout of the AuxInceptionE block is shown in Figure A.7.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$[5 \times 5, 128 \text{ stride } 2]$	$[5 \times 5, 128]$	$[3 \times 3, 128]$
AuxConv2	$[3 \times 3, 128]$		
AuxConv3	AuxInceptionE		
GAP	Global average pool		
Output	FC-100, softmax		

SSAL models issue more interpretable outputs thanks to the information conveyed by the auxiliary branches. Figure A.8 shows additional examples of predictions based on SSAL models. Each sample is passed to a pre-trained ResNet50 as the baseline, and the CAM heatmap is additionally computed. Simultaneously, we also see the same image being passed through a SSAL model which gives us access to the classes from each auxiliary group and the heatmaps based on the output of each branch.

Tab. A.4.: Architecture of auxiliary networks on WRN 16-8 for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. The layout of the AuxInceptionE block is shown in Figure A.7.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$[5 \times 5, 128$ stride $2]$	$[5 \times 5, 128]$	$[3 \times 3, 128]$
AuxConv2	$[3 \times 3, 128]$		
AuxConv3	AuxInceptionE	—	AuxInceptionE
GAP	Global average pool		
FC1	—	FC-768, ReLU	—
Output	FC-100, softmax		

Tab. A.5.: Architecture of auxiliary networks on WRN 28-10 and DenseNet models for CIFAR100. The number of output units for the last layer depends on the number of groups used in the SSAL objective. For our main result, AuxNet1 is attached on position g_2 (see Figure 7.2 on the document), AuxNet2 on position g_5 and AuxNet3 on position g_6 of the original network. The layout of the AuxInceptionE block is shown in Figure A.7.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$[5 \times 5, 128$ stride $2]$	$[5 \times 5, 128]$	$[3 \times 3, 128]$
AuxConv2	$[3 \times 3, 128]$		
AuxConv3	$[3 \times 3, 256] \times 2$		—
AuxConv4	AuxInceptionE $\times 2$	—	AuxInceptionE $\times 2$
GAP	Global average pool		
FC1	—	FC-768, ReLU	—
Output	FC-100, softmax		

Tab. A.6.: Architecture of auxiliary networks on Resnet50 for Imagenet. The layout of the AuxInception blocks is shown in Figure A.7. For our main result, AuxNet1 is attached on position g_2 (see Figure 7.2 of the main document), AuxNet2 on position g_3 and AuxNet3 on position g_4 of the original network.
*The final AuxInceptionE block has twice as many channels as the first AuxInceptionE found in AuxConv4.

layer name	AuxNet1	AuxNet2	AuxNet3
AuxConv1	$[3 \times 3, 256$ stride $2]$	—	—
AuxConv2	AuxInceptionC		
AuxConv3	AuxInceptionD		—
AuxConv4	AuxInceptionE		—
AuxConv5	AuxInceptionD		
AuxConv6	AuxInceptionE*		
GAP	Global average pool		
Output	FC-100, softmax		

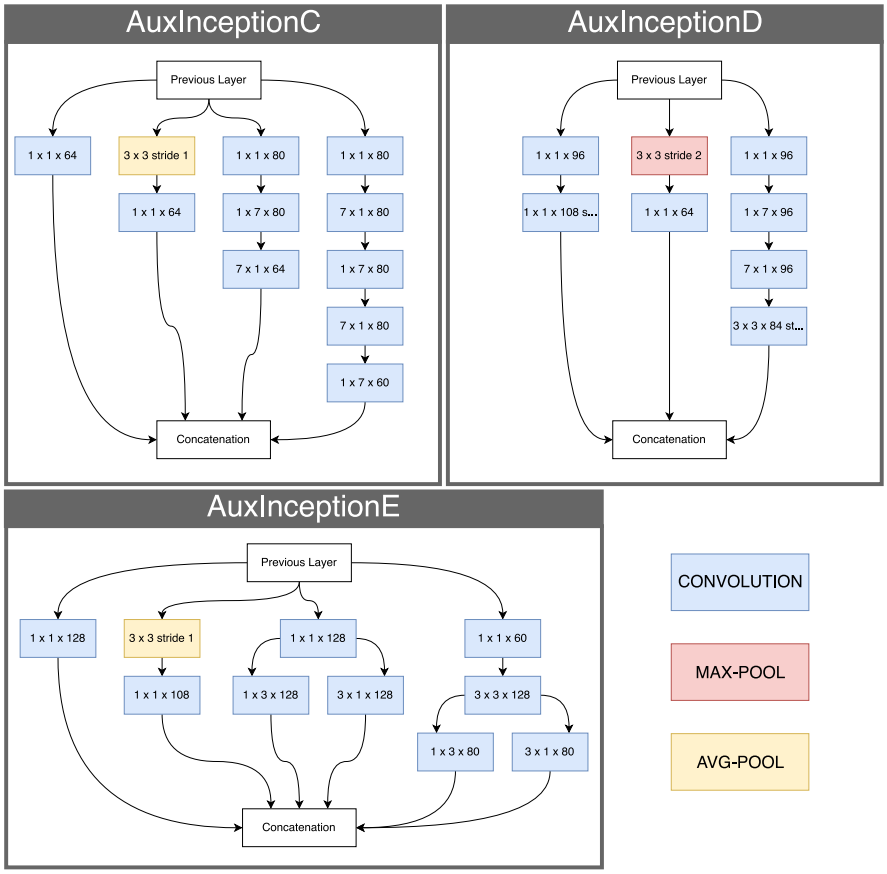


Fig. A.7.: Inception modules used for the auxiliary SSAL classifiers.

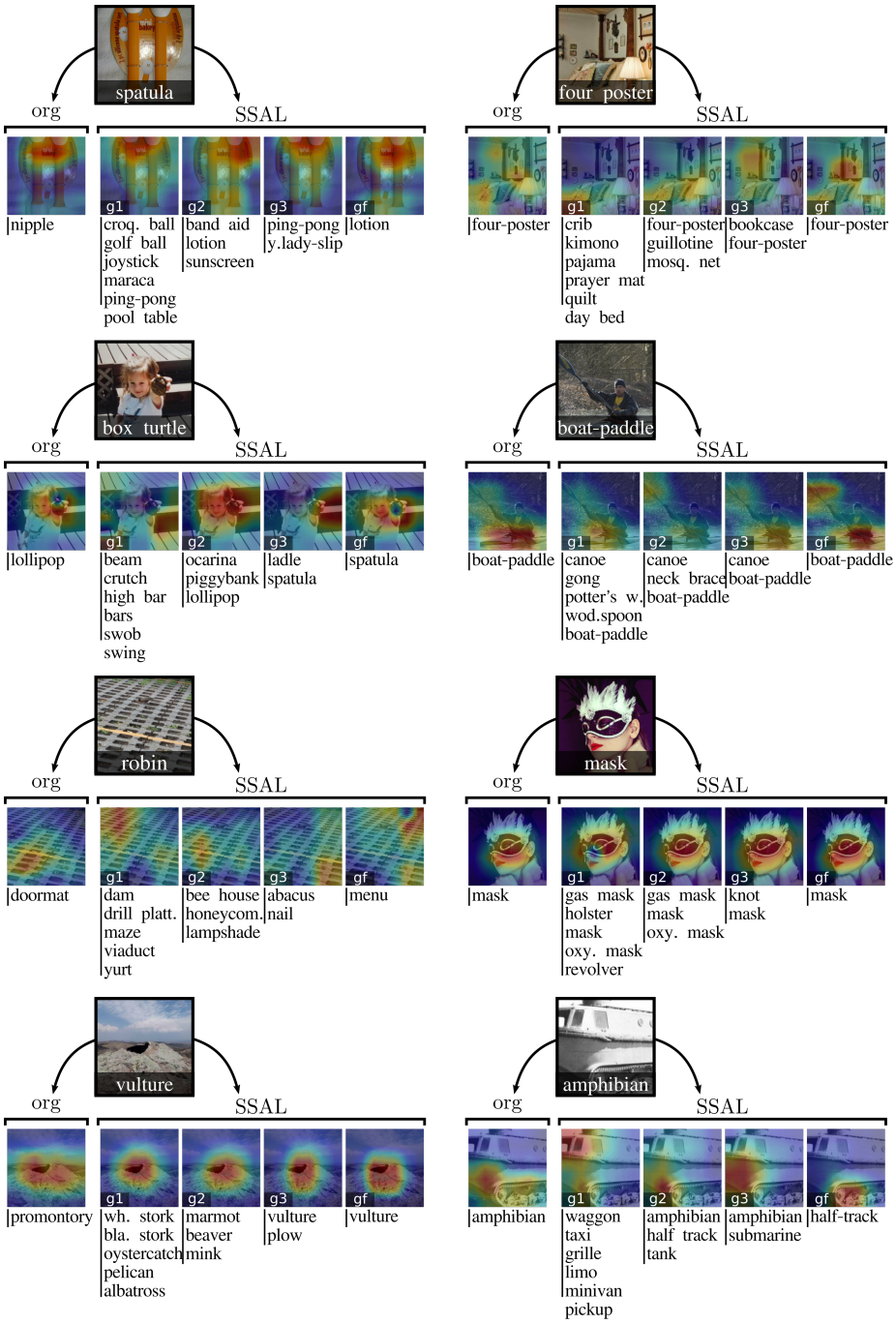


Fig. A.8.: CAM samples with and without SSAL. The left column has incorrect predictions for both models. The right column has correct predictions for both models. The last row has mixed results (one model is correct, and the other one is not).

Sebastian Palacio

Machine Learning and AI Specialist

Over 10 years of professional experience on machine learning for computer vision and language. My current research focuses on explainable AI (XAI) for Deep Neural Networks

Core Strengths

- **Scientific background**: over 12 scientific publications and patents that have led to award-winning scientific contributions in top peer-review conferences (*see Publications*). I have also been a recurrent reviewer for top conferences like ICML (recognized as top reviewer on 2020 and 2021), ICLR, ICPR, ICDAR, WACV, and ACMMM.
- **Project leading**: I am currently leading a project funded by the BMBF (German Federal Ministry of Education and Research) on explainable AI called ExplAINN.
- **Industry projects**: in agricultural, marketing and automotive sectors, bridging state-of-the-art research and industry problems.
- **Communication skills** have allowed me to develop and teach a full hands-on course on Deep Learning for university students, teams of data-scientists and project managers across multiple industry sectors.
- **Supervisory skills**: I have supervised 4 Master theses, 5 Bachelor theses, and numerous projects with students from the TU Kaiserslautern, German University in Cairo and the Universitat Politècnica de Catalunya over the last 5 years.
- **Broad experience on Digital Image Processing** with both traditional feature methods and Deep Learning.

Experience

2014 – 2022 **PhD Researcher**, *DFKI (SDS Group)*, Germany.
Developing methods on explainable AI (XAI) for Deep Neural Networks, Adversarial Attacks and unsupervised training algorithms (*see Publications*). Worked on video analysis using Adjective-Noun-Pairs and DNNs (*featured at a TEDx talk*). Developed and taught hands-on courses on Deep Learning for industry and academia.

2017 **Visiting Researcher**, *Int. Computer Science Institute (ICSI)*, Berkeley, CA.
Developed a ball-tracking prototype based on human pose cues using DCNNs for videos.

2013 – 2014 **Researcher**, *DFKI (AV Group)*, Germany.
Developed novel methods for an automatic multi-camera vehicle tracking system in Python and C++ resulting in US-Patents $9,449,258$ and $9,953,245$.

2009 – 2013 **Research Assistant**, *DFKI GmbH (former IUPR Group)*, Germany.
Feature developer for two in-house multimedia retrieval platforms: MoViMoS and SunVid

Nov 2007 – Jan 2008 **Research Assistant**, *University of Mannheim*, Germany.
Development of a prototype for panoramic image stitching in Java.

Publications

2021 **IteROAR: Quantifying the Interpretation of Feature Importance Methods**, *NeurIPS Workshop on Pre-register Science*, Online.

2021 **XAI Handbook: Towards a Unified Framework for Explainable AI**, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Online.

2020 **Contextual Classification Using Self-Supervised Auxiliary Models for Deep Neural Networks**, *International Conference on Pattern Recognition (ICPR)*, Online.

2020 **Revisiting Sequence-to-Sequence Video Object Segmentation with Multi-Task Loss and Skip-Memory**, *International Conference on Pattern Recognition (ICPR)*, Online.

2020 **P≈NP, at least in Visual Question Answering**, *International Conference on Pattern Recognition (ICPR)*, Online.

2019 **Adversarial Defense based on Structure to Signal Autoencoders**, *IEEE Winter Applications on Computer Vision (WACV)*, Aspen, CO.

2018 **What do Deep Networks Like to See?**, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, **Received the NVIDIA AI Pioneer Award**.

2017 **Classless Association using Neural Networks**, *International Conference on Artificial Neural Networks (ICANN)*, Alghero, Italy.

2016 **Symbolic Association using Parallel Multilayer Perceptron**, *International Conference on Artificial Neural Networks (ICANN)*, Barcelona, Spain.

2015 **Multi-camera vehicle identification system**, *US Patent*.
Patent Nr: US 9449258 B1

2014 **Application Exploring of Ubiquitous Pressure Sensitive Matrix as Input Resource for Home-Service Robots**, *International Conference on Robot Intelligence Technology and Applications*, Beijing, China.

2013 **Exploiting the Spatial Distribution of Interest Points Using Hierarchical Clustering for Improved Scene Retrieval**, *TU Kaiserslautern*, Kaiserslautern.
Master Thesis (Mark: 1.0)

2012 **Retrieving Objects, People and Places from a Video Collection: TRECVID'12 Instance Search Task**, *TrecVid*, Washington D.C..
Notebook Paper

Invited Talks

2021 **AMLDS Lecture: Adversarial Attacks and Links to Interpretability**, *TU Kaiserslautern*, Kaiserslautern.

2020 **ML in KL Lightning Talks: Visualization of Complex Data**, *Fraunhofer ITWM*, Kaiserslautern.

2019 **ML in KL Lightning Talks: AutoEncoders and their applications**, *Fraunhofer ITWM*, Kaiserslautern.

2019 **Understanding Deep Networks Through Properties of the Input Space**, *NVIDIA's GPU Technology Conference (GTC)*, Santa Clara, CA.

2018 **Towards Understanding Deep Network Behaviour**, *NVIDIA's GPU Technology Conference (GTC)*, Munich.

Education

2017 – 2023 **PhD Candidate**, *DFKI and TU Kaiserslautern*, Kaiserslautern - Germany.
Topic: "Towards Interpretable Models for Computer Vision"

2010 – 2013 **M.Sc. in Computer Science**, *TU Kaiserslautern*, Kaiserslautern - Germany.
Thesis: "Exploiting the Spatial Distribution of Interest Points using Hierarchical Clustering for Improved Image Retrieval". Mark 1.0

2009 – 2010 **B.Sc. in Computer Science**, *TU Kaiserslautern*, Kaiserslautern - Germany.
Thesis: "Evaluation of Parameter Influence and Dimensionality Reduction for CBIR Systems using Local Features, TF-IDF and Inverted Files."

2004 – 2009 **B.Sc. in System's Engineering**, *EAFIT University*, Medellin - Colombia.
Undergraduate. Transferred to the University of Kaiserslautern.

1990 – 2003 **High School**, *Colegio Aleman*, Medellin - Colombia.
Honorable mention for outstanding performance in sports and French

Technical Skills

Languages	Python, Bash-Scripting, C++
ML Libs	Keras, PyTorch, TensorFlow, OpenCV, SciPy, NumPy, scikit-learn
OSs	Linux, MacOS
Misc	Git, Sacred, Inkscape, GIMP, Matplotlib, LaTeX

Languages

Spanish	Native	-
English	Full professional proficiency	*85/100: MELICET*
German	Professional working proficiency	*Deutsche Sprach Diplom II/C1*
French	Limited working proficiency	*DELF A6*

Hobbies

- Playing guitar, composing music, cooking, magic tricks, traveling.

Colophon

This thesis was typeset with LaTeX 2_ε. It is based on the *Clean Thesis* style developed by Ricardo Langner and modified by Sebastian Palacio. Cover art by Sebastian Palacio. Text and grammar follows recommendations in the Chicago Manual of Style.

hoc opus est scriptum magister da mihi potum; Digitis scriptoris careat grauitate doloris.