


Doctoral Dissertation
Doctoral Program in Control and Computer Engineering (33.th cycle)

Explaining black-box deep neural models' predictions, behaviors, and performances through the unsupervised mining of their inner knowledge

Francesco Ventura
* * * * * *

Supervisor
Prof. Tania Cerquitelli, Supervisor

Doctoral examination committee
Prof. Claudia Diamantini, Università Politecnica delle Marche
Prof. Khalid Belhajjame, Université Paris-Dauphine
Prof. Elena Baralis, Politecnico di Torino
Prof. Jérôme Darmont, Université de Lyon
Prof. Genoveva Vargas-Solar, CNRS

Politecnico di Torino
2021

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Francesco Ventura

Turin, 2021

Summary

Artificial Intelligence applications are experiencing a disruptive expansion in many relevant aspects of our life thanks to the development of even better performing Deep Neural Networks (DNNs). However, along with higher performance, AI models are characterized by high complexity and opaqueness, i.e., they do not allow understanding the reasoning behind their automatic decision-making process. This widely limits their applicability and opens to a wide range of problems in many sensible contexts, e.g. health, transportation, security, and law. From one side, it is very hard to interpret the decision-making process of AI models both at the local and global levels. Furthermore, it is even harder to assess the reliability of their predictions over-time, e.g. because of the presence of concept-drift. Ignoring just one of these aspects may have very harsh consequences in real-life settings where it is supposed that users, both expert, and non-expert, should trust the decisions taken by "smart" platforms and devices. The evident complexity and relevance of these issues have led to the birth of a new branch of research, namely Explainable Artificial Intelligence (xAI).

In the literature, these challenges are faced separately and often without taking into account the latent knowledge contained in the deeper layers of the DNNs. Instead, we claim that these issues are part of the same big challenge: the Model Reliability Management. For these reasons, we aim to address these challenges by proposing (i) a unified model-aware strategy for the explanation of deep neural networks at prediction-local and model-global level, and (ii) a unified model-aware assessment framework for the management of models' performance degradation over-time.

First, the prediction-local and model-global explainability has been addressed by introducing a new explanation framework tailored to Deep Neural Networks. We introduce new unsupervised mining strategies to extract and analyze the inner knowledge contained in multiple deep layers of different models employed in different contexts, i.e. image and textual Machine Learning tasks. The explanations produced by the proposed framework consist of innovative quantitative indicators and ad-hoc qualitative visual/textual information. These explanations are developed to be easily understandable by humans enabling both expert and non-expert users to better understand and to dig into the black-box decision-making process. Also, the proposed explanation framework has been developed to adapt to as many alternative models as possible going from Convolutional Networks to complex Natural Language Models like BERT. The proposed

local and global explanations have been validated on diverse image and textual classification tasks and exploiting very different state-of-the-art deep architectures, i.e. VGG16, VGG19, InceptionV3, InceptionResNetV2, BERT, and LSTM. We show with these experiments that the proposed framework can be easily adapted to several use-cases and that we are able to effectively identify misleading patterns in the prediction process. Also, we provide a complete comparison of our methodology w.r.t. the current state-of-the-art and we quantitatively validate our solution collecting feedbacks from human users through an online survey. Our explanations have been considered more interpretable concerning the compared methods in 75% of the cases.

Then, assessing and managing the quality of the model's outcomes over-time requires dealing with the management of concept drift. To address this challenge we introduce a new Concept Drift Management framework that allows to continuously monitor the presence of drifting distributions of data that may affect the performance of the predictive models deployed in production systems. With this framework we aim to introduce a new level of model interpretability, addressing the current limitations of explanation frameworks that do not take into account over-time performance degradation. The framework is based on an unsupervised and scalable strategy that measures the drift of newly incoming data concerning the knowledge of the model at the training time. As it is unsupervised it does not require any prior knowledge about the ground-truth of newly collected data. Also, we introduced a new definition of model degradation that quantifies the per-class amount of performance reduction. Furthermore, the proposed methodology is very general and it can be applied to several different use-cases and predictive models: The proposed framework has been tested on very diverse deep models, e.g. Doc2Vec, VGG16, and BERT, tailored to unstructured data domains, i.e. images and texts. We show that our methodology is able to detect the presence of drift already when just 10% of data is drifting in the window of analysis. Finally, we demonstrate that our methodology can be horizontally distributed and it can linearly scale managing millions of input samples in the order of few minutes.

Acknowledgements

First of all, I would like to thank my Ph.D. supervisor Tania Cerquitelli who introduced me to this path and guided me in becoming a researcher.

A special thank goes to my life partner Monica. You give me every day the strength and the motivation to pursue my career in research. Together, we faced all the obstacles and challenges of this path and, even in the worst moments, you have been there for me. Thank you Monica for always giving me the right advice. Thank you for sharing with me this adventure that is our life. Thank you for your love and your sweetness. You are a wonderful person and an inspiration for me.

Then, I would like to thank my parents Elena and Vincenzo, and my brother Gabriele. Thank you, mum and dad, I know you have been always there cheering for me and this achievement is also yours. Your teachings are the ground basis for what I am now and for the person I will become in the future. I am glad to have you on my side. Also, thank you *little* brother because our passionate discussions about science are always of inspiration for me.

Finally, without the support and the friendship of the wonderful people that I had the pleasure to meet during my Ph.D. this work would not be possible.

To Monica,
my love, my life.

Contents

List of Tables x

List of Figures xii

1 Introduction 1
- 1.1 Main Achievements . . . 4
- 1.2 Other Research Topics . . . 5

2 Literature Review 7
- 2.1 Explainable Artificial Intelligence . . . 7
- 2.2 Concept Drift Management . . . 12

3 The Explanation Process: a Methodological Approach 15
- 3.1 Explanation Process Components . . . 16
- 3.2 Measuring the Influence of Interpretable Features . . . 20

4 Explaining the Convolutional Decision-Making Process 27
- 4.1 Convolutional Explanation Process Overview . . . 28
- 4.2 Interpretable feature extraction and perturbation . . . 28
- 4.3 Prediction-Local Explanations . . . 34
- 4.4 Per-Class Model Explanations . . . 37

5 Experimental Results on Deep Convolutional Models 39
- 5.1 Experimental Setup . . . 39
- 5.2 Evaluating Prediction-Local Explanations . . . 40
- 5.3 Evaluating Per-Class Model Explanations . . . 47
- 5.4 Comparisons with State-Of-The-Art Explanations . . . 48
- 5.5 Assessment of Influence Measurement . . . 53
- 5.6 Efficiency and Effectiveness of Influence Measurement . . . 56
- 5.7 Human Validation . . . 57

6 Explaining Deep Natural Language Models **61**
6.1 Deep NLP Explanation Process Overview . . . 62
6.2 Interpretable Feature Extraction . . . 63
6.3 Multi-layer Word Embedding (MLWE) . . . 64
6.3.1 MLWE Extraction in Practice. . . . 65
6.3.2 Exploiting MLWE in the Explanation Process . . . 67
6.4 Perturbation of Interpretable Features . . . 68
6.5 Prediction-Local Explanations . . . 69
6.6 Per-class Model-Global Explanations . . . 71

7 Experimental Results on Deep Natural Language Models **75**
7.1 Experiments at a glance . . . 76
7.2 Evaluating Prediction-Local Explanations . . . 78
7.3 Evaluating Per-Class Model Explanations . . . 83
7.4 Comparisons of Model-Agnostic and Model-Aware explanations . . . 87

8 Assessing and Trusting Models' Performances Over-Time **91**
8.1 A framework for Concept-Drift Management . . . 92
8.2 Evaluating model's performance over-time . . . 93
8.2.1 Embedding the model's knowledge . . . 95
8.2.2 Per-class predictive performance degradation . . . 96
8.3 Experimental results . . . 101
8.3.1 Evaluation of Descriptor Silhouette performance . . . 110

9 Conclusions **115**

Bibliography **117**

List of Tables

1.1 Misleading prediction example of a clean/toxic comment classification. The surname of a well-know politician is anonymized. 2
4.1 Predictions for Figure 4.6 with a pre-trained VGG16. 34
5.1 VGG16 (M1) predictions for Figure 5.1. 42
5.2 InceptionResNetV2 (M4) predictions for Figure 5.1. 42
5.3 VGG19 (M2) predictions for Figure 5.3 44
5.4 InceptionV3 (M3) predictions for Figure 5.3 44
5.5 VGG16 (M1) predictions for Figure 5.6 51
5.6 InceptionResNetV2 (M4) predictions for Figure 5.8 53
5.7 Pearson correlation between nPIR, nPIRP, and feature sizes. The abs_feat_size is the absolute feature size concerning the number of pixels while the rel_feat_size is the relative feature size w.r.t. the size of the input image. 55
6.1 Quantitative explanation for example in Figure 6.4. P is the positive label, N is the negative label. The (sub.) suffix indicates that the substitution perturbation has been applied. 70
7.1 Explanation of the BERT model: percentage of documents for which each feature extraction strategy produces at least one influential local explanation (i.e. with $nPIR \geqslant 0.5$), with and without combination of features. *Overall* is the percentage of documents for which at least one method found it. 77
7.2 Explanation of the CUSTOM LSTM model: percentage of documents for which each feature extraction strategy produces at least one influential local explanation (i.e. with $nPIR \geqslant 0.5$), with combination of features, for the class labels *Clean* and *Toxic*. 77
7.3 Quantitative explanation for example in Figure 7.1. T is the Toxic label, C is the Clean label. 78
7.4 Quantitative explanation for example in Figure 7.2. T is the Toxic label, C is the Clean label. 79
7.5 Quantitative explanation for example in Figure 7.3. P is the positive label, N is the negative label. 80
7.6 Quantitative explanation for example in Figure 7.4. P is the positive label, N is the negative label. 83

8.1 Summary of datasets exploited in concept drift management experiments. 102
8.2 Silhouette scores comparison between DS, SKS and Squared Euclidean Silhouette (SES). 113

List of Figures

3.1 Overview of the explanation framework. . . . 15
4.1 Local explanation process. . . . 29
4.2 Local-explanation example for class-of-interest *Pizza*. Input image (left), interpretable features (center), visual explanation (right-top), quantitative explanation (right-bottom). . . . 29
4.3 Hypercolumn extraction example. The DCNN is a VGG16 like model. The input image is $224 \times 224 \times 3$. The last 5 convolutional layers have been use to extract the hypercolumns. . . . 31
4.4 Example of interpretable feature extracted by hypercolumns' partitioning in 3 clusters. . . . 32
4.5 Feature extraction process and selection of the most informative explanation. . . . 33
4.6 *Pizza* input image . . . 34
4.7 Example of local explanations for two classes of interest. The input image is shown in Figure 4.6. Each explanation is organized with the Visual explanation (left), the map of features (center), the quantitative explanation (right). . . . 35
4.8 Class-based model explanation example for class-of-interest *Pizza*. . . . 38
5.1 *Mouse* input image (I1) . . . 42
5.2 Prediction local explanations. The input image is shown in Figure 5.1. Visual explanation (left), features (center), numerical explanation (right). 43
5.3 *Pizza* input image (I2) . . . 44
5.4 EBAnO Local explanations. The input image is shown in Figure 5.3. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right). . . . 45
5.4 (Continue) Prediction-local explanations. The input image is shown in Figure 5.3. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right). . . . 46
5.5 Class-based model explanation with class-of-interest *Dalmatian*. . . . 49
5.5 (Continue) Class-based model explanation with class-of-interest *Dalmatian*. . . . 50
5.6 Input image (I3). . . . 51

5.7 Prediction-local explanations. The input image is shown in Figure 5.6. Visual explanation (left), features (center), numerical explanation (right). 52
5.8 Input image (I4) taken from [76] for comparison. 53
5.9 Prediction-local explanations. The input image is shown in Figure 5.8. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right) 54
5.10 For each tested model the distributions of min and max nPIR values obtained by the prediction-local explanations with class-of-interest equal to the top-1 predicted class for each input image. 55
5.11 Comparison of nPIR index and *Shapley values*. 57
5.12 Survey example question. 58
5.13 Survey results. For each of the 12 images, we report the percentage of times that EBAnO, *GRAD-CAM*, and *LIME* have been selected as the best explanation result for the class of interest (multiple choice allowed). 60
6.1 Local explanation process. 62
6.2 MLWE feature extraction process. 65
6.3 BERT multi-layer word embedding feature extraction process. With: $E_{c\,[L_{id}]}^{<wp \text{ or } t>}$ such that: E is the word embedding matrix, wp and t indicate the position of the word-piece and token respectively in the input text, c is the component of the word embedding vector and L_{id} is the layer from which is extracted. 66
6.4 Examples of a *textual explanation* report. The original text was labeled by BERT as *Negative* with a probability of 0.99. The most relevant features are highlighted in red. Removed features are in squared brackets. 70
6.5 Influential set of words at model-global level for class-of-interest c_0. . . 72
7.1 Examples of *textual explanation* report for the input in Figure 7.1a originally labeled by custom LSTM model as *Toxic* with a probability of 0.98. The most relevant features are highlighted in red. 78
7.2 Examples of *textual explanation* report for the input in Figure 7.2a originally labeled by custom LSTM model as *Toxic* with a probability of 0.96. The most relevant features are highlighted in red. 79
7.3 Examples of *textual explanation* report for the input in Figure 7.3a wrongly labeled by BERT as *Negative* with a probability of 0.99. The most relevant features are highlighted in red. 80
7.4 Examples of *textual explanation* report for the input in Figure 7.4a originally labeled by BERT as *Negative* with a probability of 0.99. Features found are highlighted in red.(Continue) 82
7.4 (Continued) Examples of *textual explanation* report for the input in Figure 7.4a originally labeled by BERT as *Negative* with a probability of 0.99. Features found are highlighted in red. 83
7.5 Global explanation of toxic comment classification with LSTM. 84
7.6 Global explanation of sentiment analysis with BERT. 84

7.7 Relative size of the features extracted with MLWE strategy (i.e. size of the most influential clusters of words) for all the informative local explanations. The relative size is calculated, for each local explanation, as the number of tokens in the cluster over the number of tokens in the input text. . . . 86
7.8 Comparison of explanation between LIME and our framework for class-of-interest *Negative*. . . . 88
7.9 Comparison of explanation between LIME and our framework for class-of-interest *Negative*. . . . 89
8.1 Predictive model before and after the occurrence of concept drift. . . . 92
8.2 Concept-Drift management framework. . . . 92
8.3 Model's Performance Degradation Evaluation Over-Time. . . . 94
8.4 Visual example of distance computation among points and descriptors with $\kappa = 4$. intra-class cohesion on the left and inter-class separation on the right. . . . 99
8.5 Degradation estimation over-time. . . . 100
8.6 Data samples extracted from dataset D3. . . . 102
8.7 Concept Drift patterns. . . . 104
8.8 Dataset D1. Model degradation over time, with training on classes 0 and 1. 105
8.9 Comparison of Base DS curve at training time, and degraded DS curves at time t for dataset D1. . . . 105
8.10 Dataset D2. Model degradation over-time, with training on classes 0 (*food-drink*) and 1 (*literature*). Degradation introduced with the new class 2 (*mathematics*). . . . 106
8.11 Dataset D3. Model degradation over-time with training on classes 0 (*Cat*) and 1 (*Dog*). Degradation introduced with the new class 2 (*Fox*).) 107
8.12 Model degradation over-time measured for dataset D3 varying the size of the window of analysis. Max window size $T = 100$ and step $s = 10$. Training on classes 0 (*Cat*) and 1 (*Dog*. Degradation introduced with the new class 2 (*Fox*).) . . . 109
8.13 Dataset D4. Model degradation over-time with training on classes 0 (World), 1 (Sports), and 2 (Business). Degradation introduced with the new class 3 (Science). The window of analysis is equal to 2000 samples.) 110
8.14 Scikit-learn Silhouette and DS cost comparison on a single-node. D is the time required to compute the descriptors, DS is the time required to obtain the DS values and SKS is the time required for the SciKit-provided Silhouette, for datasets from d_1 to d_5. . . . 112
8.15 DS scalability performance. . . . 112
8.16 Silhouette curve comparison computed with DS configured with 200 descriptors and SKS. . . . 113

Chapter 1

Introduction

Artificial Intelligence applications are experiencing a disruptive expansion in many relevant aspects of our life. From Industry 4.0 to smart devices, we are surrounded by decisions taken by intelligent algorithms [65, 3, 77, 33, 57]. Deep Neural Networks (DNNs) are at the root of this revolution. DNN models are rapidly boosting the evolution of many Artificial Intelligence tasks such as visual and textual classification [38], image captioning [35], question answering [68], sentiment analysis [82]. There is no doubt that these models brought a significant improvement to the accuracy of these tasks. However, along with higher performance, AI models are characterized by high complexity and opaqueness, i.e., they do not allow understanding the reasoning behind their automatic decision-making process. This widely limits their applicability and opens to a wide range of problems in many sensible contexts, e.g. health, transportation, security, and law [60, 21, 104, 6]. From one side, it is very hard to interpret the decision-making process of AI models both at the local and global levels [69, 1, 42, 22, 99]. Furthermore, it is even harder to assess the reliability of their predictions over-time, e.g. because of the presence of concept-drift [28, 9, 103, 74]. Ignoring just one of these aspects may have very harsh consequences in real-life settings where it is supposed that users, both expert, and non-expert, should trust the decisions taken by ”smart” platforms and devices. The evident complexity and relevance of these issues have led to the birth of a new branch of research, namely Explainable Artificial Intelligence (xAI) [5, 32].

In our life, there are many examples where AI models failed in their tasks with serious consequences in some cases. As a motivating example, let us consider a social network moderated by a deep learning model that has been trained to recognize whether a new post is in contrast with the policy of the platform. The model is trained to recognize if the content of a post contains inappropriate content (e.g., insults, discriminating language). For each new post, it performs a prediction and it is eventually censored. The problem is that, in presence of predictive biases, a clean comment may be labeled as inappropriate. Table 1.1 shows an example of misleading predictions in this context. We obtained these results by training an LSTM model to recognize if a textual comment contains *Clean* or *Toxic* language. More details on this in Chapter 7. Both sentences are

Sentence	$P(Toxic)$
`Politician-1 is an awesome man`	0.17
`Politician-1 is an intellectual`	0.89

Table 1.1: Misleading prediction example of a clean/toxic comment classification. The surname of a well-know politician is anonymized.

expressing clean language. Indeed, the phrases contain the surname of a well-known politician (anonymized for privacy) and positive adjectives. However, the predictions are contradictory. As a consequence, the reliability and the fairness of the model are compromised. Similar reasoning can be extended to other domains even more critical, e.g. self-driving cars. These behaviors cause ethical and technical issues to different typologies of users [52]. Developers who are developing AI models need to understand the causes of misleading behaviors to fix them before deployment. Yet, managers, who are legally responsible for the services, may decide not to use these new technologies because of their poor interpretability avoiding strong juridical consequences. Finally, the end-users have the right to understand and trust the automatic decisions taken by the platform. Indeed, if a user does not trust a new technology he would not use it [91].

The opaque nature of the deep neural networks does not permit understanding the causes behind the decision-making process. This opens to new wide challenges associated with different levels of transparency:

- *Challenge 1 - Local Level Explainability:* Understanding the reasons that cause local decisions.
- *Challenge 2 - Global Level Explainability:* Understanding if the presence of common input patterns is affecting the prediction process at a global level.
- *Challenge 3 - Over-Time Reliability:* Assessing the performance of the model overtime in absence of ground truth labels. Indeed, even if errors are not occurring at the time of model deployment, they can still occur in the future, e.g. in presence of concept-drift.

In the literature, these challenges are faced separately. Several explanation frameworks both model-agnostic and domain-specific are available. Model-agnostic approaches [69, 47] can be applied to any predictive model and task, proposing visual and numerical explanations. However, being model-agnostic is at the same time an advantage and a limitation since the explanation process is approximated, i.e. they can not mine the knowledge contained in the inner layers of the model. On the other hand, domain-specific solutions [76, 64, 79], exploit models' knowledge but are often tailored to specific deep learning architectures and data domains. Also, they are usually able to provide prediction-local explanations only. There are few solutions for global explanations, e.g. [69, 47, 79], and often they apply to specific data domains and/or models. Furthermore,

the current trends in xAI do not include the assessment of model performance over-time, e.g. due to the presence of concept drift, as a transparency requirement. Also, the current literature on concept-drift management [28, 55, 67, 11, 30, 74] does not propose a unified framework for handling different ML/AI architectures and data domains.

We claim that these problems are part of one big challenge, namely *Model Reliability Management*. So, we aim to address these challenges by proposing (i) *a unified model-aware strategy for the explanation of deep neural networks at prediction-local and model-global level*, and (ii) *a unified model-aware assessment framework for the management of models' performance degradation over-time*.

In this work, we focus on models applied to unstructured information only, i.e. visual and textual data. All the proposed strategies in this work exploit the knowledge of the models to produce their results. We aim at providing prediction-local, class-based model-global, and over-time predictive performance explanations of AI models.
In the prediction-local explanation process, we analyze *interpretable features* extracted through a new unsupervised mining process of the inner layers of the models. Therefore, we measure the influence of the features by computing two new indexes, i.e. the *normalized* P*erturbation* I*nfluence* R*elation* (nPIR) and the *normalized* P*erturbation* I*nfluence* R*elation* P*recision* (nPIRP). The explanations are proposed to the users with both visual and numerical reports. A preliminary version of this work has been published in [98, 99].
The class-based global-explanations are then computed by aggregating the information extracted from a set of local ones. Accordingly to the data domain, the global explanations are computed and presented with different strategies. For visual data, the report summarizes the most influential visual patterns, given a set of locally explained images. Instead, for textual data, the report computes the *Global Absolute Influence* and the *Relative Absolute Influences* of each input feature, given a set of locally explained textual documents.
Instead, to assess the predictive performance over-time we introduced a new scalable unsupervised approach to measure the shifts of data distribution. Our methodology is based on a new scalable index, namely *Descriptor Silhouette*, and a new definition of model performance degradation [100, 19]. Thanks to this approach we can quantify how much the performance of a predictive model is degrading over-time. Also, we aim to achieve a solution not tailored to a specific use-case or application domain, nor to a specific data type. All the explanations are human-readable at all levels. Finally, we show how our solutions can be extended to different use cases describing the strategies that have to be exploited accordingly to the input data type.

The main contributions of this work can be summarized in:

- The design of a novel explanation process that exploits the inner knowledge of multiple layers of AI models, i.e. DCNNs and NLP models.

- The introduction of two new indexes, i.e., nPIR and nPIRP, to efficiently quantify both the *influence* and the *precision* of the input features concerning local predictions.
- The computation of informative class-based model-global explanations.
- The unsupervised extraction of *interpretable features* easily understandable by humans.
- The introduction of a novel unsupervised and scalable methodology to assess the model's performance over-time based on a new index, i.e., the Descriptor Silhouette.
- Extensive experimental validation of our solutions on state-of-the-art AI models, i.e., VGG16, VGG19, InceptionV3, InceptionResNetV2, BERT, and LSTM.

In Chapter 2 we introduce and discuss the currently available literature. Chapter 3 provides a methodological overview of the proposed prediction-local and class-based explanation process. Chapter 4 describes how the explanation process is adapted to the visual data domain by explaining Deep Convolutional Neural Networks (DCNN). In Chapter 5 we provide the experimental validation of the explanation process for DCNN. Chapter 6 describes how the explanation process is customized to address natural language processing tasks by explaining models like BERT and Long Short-Term Memory networks. In Chapter 7 we provide the experimental validation of the explanation process for NLP models. Chapter 8 describes in detail our solution to models' predictive performance assessment over-time and provides extensive experimental validation of the proposed strategy. Chapter 9 concludes the dissertation.

1.1 Main Achievements

In this Section the main achievements obtained by the Ph.D. candidate will be discussed. The candidate is co-author of 14 published articles with proceedings covering several research areas such as Applied Machine Learning, Explainable Artificial Intelligence, Data Mining, and Big Data Management and Analytics. In particular, this document is the result of some of the achievements that the candidate obtained in the three years.

A preliminary version of the explanation framework presented in this thesis has been firstly published in [99] for which a pre-print version is available in [98]. Instead, the latest results related to the explanation framework have been recently submitted to two journals for revision. The explainability of convolutional models has been treated in [97], the work is ready to be submitted to a journal for revision. While the explainability of deep NLP models is treated in [102], the work has been submitted to the "Knowledge and Information Systems" journal for revision. This explanation framework has been

also released open source [96][1]. Finally, this thesis extends the results obtained in the context of concept-drift detection and published in [100, 19, 17].

Furthermore, the research activities carried out by the candidate resulted in international achievements.

- The candidate won the BEST PAPER AWARD for the paper [3] titled *"iSTEP, An Integrated Self-Tunin g Engine for Predictive Maintenance in Industry 4.0"* published in the 16th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2018).
- The candidate won the BEST PAPER AWARD for the paper [14] titled *"Clustering-Based Assessment of Residential Consumers from Hourly-Metered Data"* published in the International Conference on Smart Energy Systems and Technologies.
- The candidate won the STUDENT SCHOLARSHIP for the paper [19] titled *"Towards a Real-Time Unsupervised Estimation of Predictive Model Degradation"* published in the "Proceedings of Real-Time Business Intelligence and Analytics (BIRTE)".

1.2 Other Research Topics

During the three years course, the candidate collaborated with national and international researchers on different research projects achieving relevant results in different areas. The candidate has been involvement in four research contracts funded by international companies and public entities plus an external research period of four months at the Technische Universität of Berlin.

The first research contract has been funded by a private company (AMADA). The objective of this research contract was to identify an effective and efficient approach to improve the design time, adapt manufacturing processes, and re-use sub-processes, by exploiting the similarity of CAD objects with a particular focus on the automatic bending process.

Then, the candidate has been involved in a research contract funded by ENEL Foundation. The objective of this research contract was to identify and characterize the consumption profiles of individual consumers through the analysis of hourly metered-data. The candidate collaborates with the design of the process and he developed the CONDUCTS engine (CONsumption DUration Curve Time Series) exploiting big-data strategies, data-stream processing and unsupervised machine learning to identify patterns of individual electricity consumption, patterns of consumers behavior and characteristics of consumer profiles over time. This contract has been carried out in collaboration with the Department of Energy (Politecnico di Torino). The results of this project have been published in [14, 15].

[1] https://ebano-ecosystem.github.io

The candidate also participated in the DISLOMAN research project funded by Piedmont Region. The candidate was charged with studying the state-of-the-art technologies employed in the Industry 4.0 domain concerning the management, storage, monitoring, and analytics of huge flows of data, generated by networks of sensors and Internet of Things devices. Moreover, the candidate developed a framework, taking advantage of different distributed technologies, to face the emerging requirements of this new field of research. The results achieved in this project have been published in [3].

Then, the candidate collaborated in the SERENA research project funded by European Commission. The candidate was charged with studying the state-of-art techniques for industrial data processing, concerning the machine learning pipeline employed in the Industry 4.0 context. The candidate engineered and developed the machine learning building blocks that fulfill the requirements of the companies involved in the project, including (i) pre-processing and classification model training service, (ii) real-time and long-term prediction service, (iii) Self-Assessment service, (iv) Self-Labelling service. Each service has been deployed in a distributed and containerized environment provided by the other partners of the project. The results of this project have been published in [3, 65, 100, 18, 16, 59].

Finally, in 2020 the candidate has been involved in an external research period of 4 months at the Technische Universität of Berlin in the Database Systems and Information Management Group (DIMA). In this period, he designed and developed an innovative query workload training data generator for ML-based data management. The new engine is characterized by a three steps data-driven white-box workload generation process which involve (i) the generation of Abstract Execution Plans starting from a small pre-existing input workload, (ii) the instantiation of new executable jobs by analyzing the user's input data, and (iii) the forecasting of jobs' cost labels by learning from the available computational resources and from a small amount of executed jobs. The results of this external research activity have been published in SIGMOD 2021 [101].

Chapter 2

Literature Review

In this Chapter we report the latest advancements in the literature related to Explainable Artificial Intelligence and Concept-Drift Management. Section 2.1 discusses the current state-of-the-art and details the major challenges that we aim to address. Section 2.2 describes the current achievements in Concept-Drift Management highlighting the corresponding open challenges.

2.1 Explainable Artificial Intelligence

A recent research trend, namely *Explainable Artificial Intelligence*, has been focused on making ML/AI models interpretable and/or explainable. In literature, there have been recognized two main large categories of approaches to achieve interpretable results, i.e. model transparency and post-hoc explainability. Model transparency is referred to all those strategies that allow understanding how a model is working internally. In this category, we can put interpretable models by nature, e.g. decision trees, and techniques that can make a model interpretable, e.g. visualization techniques that show the inner functioning of deep learning models [75, 84, 83]. Instead, with post-hoc explainability (or interpretability) we refer to all those methods that aim to explain the predictions/behavior of AI models after the training phase, e.g. [42, 31, 1].

Furthermore, the categorization of explainability techniques includes the type of data under analysis (e.g. structured data, images, text), the machine learning task performed by the model (e.g. classification, forecasting, clusterization), and the typology of explanations that should be provided (e.g. local explanations, global explanations, over-time reliability explanations). Up to now, many efforts have been devoted to explain the prediction process in the context of structured data (e.g. measuring quantitative input influence [22], by means of local rules in [61]) and of deep learning models for image classification (e.g. [76], [27], [99]), while less attention has been devoted to multi-domain explanation frameworks for unstructured data analytics.

In this work, we focus on post-hoc methodologies. The works belonging to the post-hoc category will be grouped into three further categories: model-agnostic, domain-specific, and task-specific approaches.

Model-agnostic approaches. Common techniques exploited by data scientists are model-agnostic approaches, like *LIME* [69] and *SHAP* [47]. Secificaly, *LIME* [69] produces prediction-local explanations for any predictive model independently from their architecture since it does not take into account the inner functioning. Moreover, it can be applied to both structured and unstructured data (e.g. image, text). To compute the local explanation, *LIME* exploits a surrogate linear model that performs a local approximation of the prediction made by the more complex black-box model. The surrogate model is trained with local perturbations of the input instance. Then, the explanations are produced by analyzing the coefficient of linear surrogate which are interpretable by definition. On the other side, the model agnostic explainer implemented in *SHAP* [47] is based on the idea that different input features may have a different contribution amount in the prediction process. To measure this contribution they exploit a concept coming from *Game Theory*. The per-feature contribution is measured by exploiting the *Shapley Values* [78]. In the prediction process, the model outcomes are considered as a collaborative contribution of the elements that compose the input data So, the local explanation is composed by the measure of the contribution of each feature in the prediction task. However, the feature to analyze are analyzed not taking into account the knowledge of the model, i.e. also the feature extraction process is model-agnostic. This requires a more complex analytical process to find a suitable explanation for the prediction.
Model agnostic techniques are really powerful and simple to use in many domains, but often they provide very approximate explanations, limiting their reliability in critical contexts. They are not able to analyze the prediction process taking advantage of the information contained in the model under analysis and to give specific outcomes taking into account the domain of interest. Although they can be applied in the context of NLP and image processing, they are not aware of the real knowledge of the model, i.e., they can not explain what the model has learned. They do not have access to the inner details of the model and this can lead to less accurate explanations. For example, in the case of NLP, model-agnostic techniques analyze the impact of singular words over local predictions, without taking into account the complex semantic relations that exist in textual documents (i.e., semantically correlated portions of text). However, modern models like [23] are considering those complex relations and a reliable explanation should consider that as well.

Domain-specific approaches: Image processing The literature proposes several approaches to study the behavior of DCNNs. Most of these methodologies take into account the model's knowledge [79, 27, 107, 64, 76] but they are tailored to the image

domain. The general approach is to analyze the information produced during the prediction process to highlight the portions of the input that mostly affect a specific outcome. [79] explores approaches for prediction-local and class-local explanations by exploiting the knowledge of the model. Prediction-local explanations visualize the portions of the input that mostly characterize the prediction through saliency maps. Instead, class-local explanations are computed by using the CNN model as a generator of images by maximizing the probability of a target class-of-interest. Generating an image in this way allows to shows which are the patterns that are characterizing the class-of-interest. However, this methodology requires a non-trivial adaptation of the model to convert the prediction process into a generation process. Also, [107] identifies an explanation strategy that requires modifying a traditional CNN making it self-explainable. The main idea here is that each convolutional layer should be activated only by a certain part of the input image which in turn belongs to a specific category. The explanation highlights the influential object parts by means of saliency maps. The concept of saliency maps is also exploited in [27] which proposes a paradigm that learns the minimally salient part of an image. Their goal is to find the smallest perturbation of the input that allows to reduce the brings classification score of a given class-of-interest. Similarly, RISE [64] studies the effect of perturbing randomized input samples to produce prediction-local explanations. To do so, they measure the effects of randomly the delete and insert input pixels on the outcomes of the model. Differently from other methods, they do not take into account the internals of the model. This, increase the applicability of their methodology but reduces the efficiency and effectiveness. Instead, *GRAD-CAM* [76] elaborates prediction-local explanations through a white-box saliency approach. This methodology analyzes the gradient of the last convolutional layer of a DCNN, generalizing the approach proposed in [109]. The explanations are composed of saliency maps that highlight the most influential input regions of a prediction. As we aim to do, also GRAD-CAM [76] has been human-validated to assert the clearness of the produced explanations.

We notice also that several works analyze reactions of the model when perturbing input concepts [2, 99, 47, 69, 76, 27]. Some of these techniques rely on random approaches, e.g. [64, 69]. However, a random perturbation does not guarantee that the identified features contain meaningful information for the model. This reduces the efficiency and effectiveness of those approaches. Differently from the current state-of-the-art, we aim to make also the extraction of the features to perturb an interpretable step that can help the user to understand the behavior of the model.

The main research challenges that has to be addressed in the context of image processing with convolutional neural networks are multiple:

- Extracting the inner knowledge of the DCNNs taking into account multiple layers of deep networks.
- Identifying a transparent process able to extract the interpretable features, i.e.

image segments, that should be understandable by human users and meaningful for the prediction process.

- Quantifying in an efficient way the contribution of each interpretable feature for a given local prediction to understand how much each of them is influencing the class-of-interest and to measure the precision with which they are associated to the learned concepts.
- Visualizing the explanations in the clearest way possible by including as much information as possible, overcoming the limitations of the currently exploited saliency maps.
- Computing model-global explanations to visually explain the behavior of a predictive model by aggregating and analyzing similar concepts that the models have learned at the local level.

Domain-specific approaches: Natural Language Processing Differently than model-agnostic frameworks like [69, 47], some strategies exploit the information contained in the model also in the context of NLP. Common xAI techniques for NLP models are presented in [53]. As in the image domain, many of the proposed techniques are based on feature-perturbation strategies, e.g. [2, 99, 47, 69, 44, 56]. Indeed, they analyze the model reactions to perturbed versions of the input and then they compute prediction-local explanations. Again, this approach requires a meaningful selection of the input portions to perturb, since the efficiency and the effectiveness of the produced explanations strictly depend on this process. The authors of [56] use an approximate brute force strategy to analyze the influence of each sentence in the input on the predictions made by LSTM models. Then, they compute an importance score, tailored to LSTMs, to select the sentences with a larger contribution over the prediction process. This solution has been developed for LSTM models and it is difficult to generalize to other models. Also, [2] introduces an explanation strategy tailored to structured and sequential data models that exploit a perturbation strategy. However, they introduce a semantically controlled perturbation process by exploiting *variational autoencoders*. This explanation strategy has been tailored to sequence to sequence tasks, e.g., machine translation, and it requires training an external variational autoencoder. The usage of an external model significantly reduces the applicability of the approach in real-life problems. Unlike the previous work, [41] proposes to ”train” the explainer alongside the predictor. So, they introduce an *encoder-generator* framework able to extract a subset of inputs from the original text as an interpretable summary of the prediction process. Despite the improvement, they still require training a separate model that deals with the explanation process. Similarly, [44] proposes to train a surrogate reinforcement learning model that with the aim to identify the minimal set of words to perturb to cause a change in the model outcomes. Again, this method requires the training of an external model to extract features to be perturbed. We claim that relying on a separate model is not a good

idea since it could introduce a further level of opacity and complexity which affect in turn the reliability of the explanation process.

To overcome the issues of the above-mentioned works in the NLP domain we have to face different challenges:

- Defining a common strategy to extract the inner knowledge from Deep Natural Language models taking into account multiple layers.
- Identifying a transparent process able to extract semantically meaningful features considering both the NLP domain and the concepts learned by predictive models. Of course, even in this data domain, the extracted features should be understandable by human users and meaningful for the prediction process.
- Quantifying the contribution of each extracted feature for a given local prediction. Similarly to the previous domain, we aim to understand how much each portion of the input is influencing the class-of-interest. Also, we aim to obtain the most concise and complete explanation possible in an efficient way.
- Computing model-global explanations to highlight global patterns in the input data that are biasing the decision-making process of NLP models by aggregating and analyzing similar concepts that the models have learned at the local level.

Task-specific approaches. There are use-cases in which task-specific solutions are necessary. [110] focuses on explaining the duplicate question detection task. The authors developed a task-specific model based on the attention mechanism. The explanation is provided by a visual analysis of the attention matrix. This allows inspecting the inter-words relations learned by the model. Of course, this strategy is limited to AI models that are based on the attention mechanism. Also, they require significant expertise to be interpreted. [108] proposes an explainable tag-based recommendation model. The aim is to increase the interpretability of the users' recommendations by providing a report showing the user's preferences correlated with the model's learned topics and the predicted tags. Finally, [26] exploits a linguistic explanation approach for fuzzy classifiers. The proposed explanations include reasons, confidence, coverage, and feature importance. However, the authors do not take into account the complexity of AI models.

In this work, we do not take into account specific tasks. Our goal instead is to propose a general framework that can be adapted to several tasks and which allows exploiting the inner-knowledge of a large variety of AI models.

2.2 Concept Drift Management

The current literature related to Explainable Artificial Intelligence does not include the explanation of models' performance over-time. We claim though that this is a crucial aspect while managing complex data-driven platforms. The complexity and velocity of the data analyzed by AI models can radically change-over-time. So, also the performance of models trained on old data distributions may change drastically when deployed in production systems, e.g. due to the presence of concept drift.

In [93] the phenomenon of concept-drift is defined as a change-over-time of the concepts collected in real-world applications. This work claims that the concepts often do not remain stable over-time and these changes can make a predictive model, built on old data, inconsistent with the new incoming data. Then, the drift can occur *Abruptly*, so without any prior notice or *Gradually* over-time. The result however is always the same: an old model that can not be considered reliable is not properly tested and eventually retrained periodically. Furthermore, the drift of concepts can be divided also between *Real* and *Virtual*. The real concept drift occurs when the shift of the distribution of the collected data is causing also a shift in the distributions of the predicted classes and requires the retraining of the predictive model to properly adapt to the new concepts [73]. The virtual drift instead happens when only the distribution of the incoming data changes without influencing the distribution of the target classes [93]. In this last case, the shift of the data distribution is still included in the decision boundaries of the predictive model so its performance is unaltered while the data is actually shifted. Distinguishing from the two types of drift in practice, without recurring to ground-truth labels is a very hard problem that, to the best of our knowledge, has not been solved yet and that is out of the scope of this work.

A large number of works dealing with concept drift are collected in [28] where the authors identified interesting trends in modern concept-drift management strategies. A common strategy to address the detection of drift is to monitor the changes in distributions at different time windows. [103] proposes an entropy-based technique to monitor IoT data streams (e.g., wearable devices) Specifically, this work aims to detect abrupt concept drifts due to context changes. Also, [67] compares different strategies that measure drifts in data distributions. In this empirical study, they combine dimensionality reduction techniques and statistical testing to propose a new strategy for detecting concept shifts in real-life settings. Adaptive windows techniques have been explored in [9] to efficiently monitor the performance of a predictive model over-time. This idea has been also extended by [30] which introduces a parallel adaptive windowing to manage high-velocity data streams. [37] presents a concept drift detection technique based on Support Vector Machines, validated in the case of textual data. An analysis of discrete-time Markov chains affected by concept drift has been recently proposed in [71], providing a collection of change-detection mechanisms and an adaptive learning algorithm suited to this specific context. [105] addresses the problem of imbalanced classes in online learning. Training on an imbalanced dataset can indeed cause a predictive model

to suffer from the presence of drifting concepts. They reveal a problematic absence of methodologies tailored to this problem and they carried out an empirical study identifies the principal challenges when dealing with imbalanced streams of data. Instead, [86] faces the challenge of detecting model degradation in incremental learning. To this aim, they propose a tree-based ensemble learning method that aims to exploit the knowledge learned by past models and newly collected data over-time to create an updated version of the model itself. Unlike other works, [74] addresses the problem of measuring the performance of AI models from a different perspective. Indeed, the authors propose to estimate the accuracy of an AI model by exploiting a further surrogate performance predictor. The performance predictor is trained with the specific goal of understanding if a prediction is accurate or not without knowing anything about the input ground truth.

The validation of many cited works is limited by the fact they use controlled synthetic datasets with known concept drifts to evaluate their approaches. A further challenge highlighted in [93] is the inability of most state-of-the-art approaches to identify the presence of local concept-drift, i.e. models are discarded because their performance is lowering at a global level, while in most of the cases it is just a portion of the knowledge (e.g. singular classes) that are affected by drifting concepts. Moreover, scalability has been only partially addressed by very few works [30]. We believe instead that scalability should be a mandatory requirement when dealing with models that are used on even more large-scale scenarios. Finally, some of the related works rely on the fact that ground-truth labels would be available shortly after the prediction, e.g. in stock market forecasts you have access to the correct decision shortly after the prediction. However, in most applications, continuously collecting data sets including labeled data may be very hard, or even unfeasible.

The challenges in the management of concept-drift are multiple. In particular, we aim to:

- Include new concept-drift management strategies into the Explainable Artificial Intelligence pipeline, improving the reliability of the models over-time even in absence of ground-truth values.
- Identify a general strategy to deal with concept-drift that should not be tailored to any specific use-case and easily adaptable to different predictive models, with a particular focus on deep learning models.
- Define a new quantitative unsupervised measure of concept-drift, i.e. not tailored to the availability of ground truth values at any point in time.
- Provide a horizontally scalable solution fitting the Big Data requirements of modern data applications.

Chapter 3

The Explanation Process: a Methodological Approach

The explanation process proposed in this work has been designed to fulfill the lack of transparency in modern Artificial Intelligence (AI) algorithms. Specifically, we focused on AI models applied to high-dimensional unstructured data, i.e. images and texts. The developed explanation framework has been designed as general as possible, allowing to apply it to a large variety of deep learning architectures, e.g. Deep Convolutional Neural Networks, Long Short-Term Memory, and Attention Based Architectures like BERT. Figure 3.1 shows a high-level overview of the explanation process presented in this work, i.e., the explanation of local predictions and class-global behaviors of the predictive models.

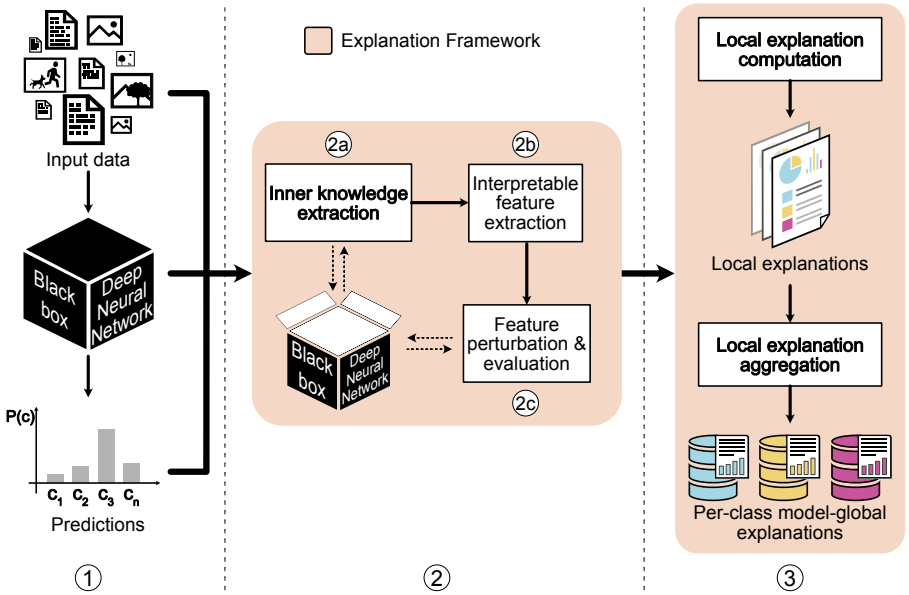


Figure 3.1: Overview of the explanation framework.

The automatic decision-making process carried out by a predictive model is usually affected by the content of the input and by the model's knowledge incorporated at training time (Step (1)). In classification tasks, the decisions of a predictive model are usually represented by a distribution of probabilities. These probabilities define which would be the most suitable class to assign to the input data. We have two main goals:

- Explaining the reasons why a specific class-of-interest has been predicted for a given input sample.
- Explaining the per-class global behavior of the predictive model, given a pool of input samples.

To produce reliable and effective explanations, the framework takes advantage of all the main components of the prediction process, i.e. the input data, the Deep Learning model, and the produced predictions for each input sample.
First, the explanation framework extracts the inner knowledge from the black-box model (Step (2a)). This step is very important making the explanation process representative of the target model. Indeed, by mining the knowledge of the model itself we optimize the final explanation to be as informative as possible for the end-user.
Then, to optimize the explanation process, we extract sets of aggregated features that have a relevant role in the prediction process. We do this by analyzing the inner model's knowledge. The framework computes a set of *Interpretable Features* (Step (2b)). We refer to these features with the term *interpretable* since they are selected to be both meaningful for the predictive model and understandable by humans.
Finally, being the prediction process potentially highly affected by the computed features, we measure how each of them is contributing to the predicted outcome. We do this by iteratively perturbing and evaluating the reactions of the AI model for each feature (Step (2c)).

The explanation process produces sets of *prediction-local* and *per-class model-global* explanations (Step (3)). For each input sample and each class-of-interest, i.e. each predicted class, a prediction-local explanation is computed. So, a user can understand the causes of a specific prediction and he can decide if to trust it or not. Also, when a pool of explanations is available they are aggregated to produce per-class model-global explanations. In this way, common input patterns can be analyzed understanding if the model had been affected by prejudices or biases in the prediction process.

In the rest of the Chapter we will introduce each component of the proposed explanation process in Section 3.1 and we will define the new indexes that we exploit to measure the interpretable features' influence and influence precision in Section 3.2.

3.1 Explanation Process Components

Inner-Model Knowledge. We claim that deep models know more than what they actually tell with predictions. It is widely recognized in the literature that the hidden

layers of deep neural networks contain valuable latent knowledge [75, 34, 84, 83]. So, the analysis of the knowledge, latently hidden in a model, can lead to very accurate results for Explainable Artificial Intelligence (xAI), other than providing accurate predictions. To produce reliable and effective explanations, we aim to take advantage of this inner-knowledge contained in multiple layers of the model.
In our design, the explanation process has to be deployed alongside the predictive model. Having access to the inner functioning of a model is one of the requirements of our methodology. However, it is very common for data scientists to have access to all the details of the model when designing, training, fine-tuning, or even just deploying a pre-trained model. The inner-knowledge extraction process has to be tailored to the architecture of the model itself. Indeed, different architectures learn concepts in different ways. Models based on the Attention mechanism [95] are very different from the ones based on convolutional layers [80]. Therefore, we propose different strategies to extract the knowledge contained in different models. We propose to analyze deep convolutional models through hypercolumns representation [34], while for attention-based models like BERT we introduced the Multi-Layer Word Embeddings (MLWE) feature extraction. More details about the inner-model knowledge extraction techniques exploited in this work are reported in Chapters 4.2 and 6.3 concerning the explanation process for convolutional models and NLP models respectively.

Interpretable Features. Machine learning models based on neural networks have the great power to learn, during the training phase, how to automatically extract the most relevant features from an input. Black-box models process complex unstructured input data by considering a small unit of analysis. For example, DCNN input units of analysis are pixels in the context of image processing. The deep network then automatically learns, at training time, how to aggregate semantically correlated pixels by means of convolutional filters. In a similar way, Deep Natural Language models analyze words or pieces of words, i.e. tokens, as the unit of analysis. Then, the inner layers learn the contextual relations among input tokens during the training process.

When models are deployed in production environments, they exploit the learned knowledge to predict the outcomes for new input samples. So, each unit of analysis included in a sample is potentially affecting the decision-making process of the model. However, in real cases, singular units may not be that influential on the outcomes of the model. Instead, often there exists a subset of input units that results much more influential than all the others. Identifying the subset of the few subsets of influential units of analysis is a challenge.

Model agnostic approaches [69, 47] face this problem with not efficient strategies. In practice, they do not exploit the knowledge of the model under the exam to find the set of influential features. From one side this increases their applicability to any model. On the other instead, this widely limits their efficiency and reliability, since the search space of possible influential subsets of units could be very large (e.g. large images, long texts). So, analyzing the whole search space to find the optimal explanations would

not be feasible. They use heuristic strategies to reduce this search space that limits, in turn, the reliability of the explanations. In the worst case, their generated explanations would highlight erroneous units of analysis as influential for the prediction process. Instead, we strongly believe that exploiting the inner knowledge of deep models is a requirement.

The objective of our explanation process is to provide results understandable by any human user. To do so we propose a feature extraction process based on the mining process of the inner-model knowledge, i.e. the output of multiple hidden layers. In this way, we can efficiently identify which are the subsets of input units mostly contributed to the model outcomes. Each subset of units of analysis identified by this mining process is called *Interpretable Features*. This name comes from the dualistic nature of the knowledge extracted at this step. With *Interpretable Features* we provide two important properties:

- They are representative of the knowledge learned by the model in the exam.
- They are easily interpretable by human users.

First of all, they are the numerical features of our explanation process. They include the knowledge extracted directly from the predictive model. Second, the extracted features have a one-to-one correspondence with the input units of analysis so that each of them is traceable to a specific portion of the input. Both the properties are guaranteed by an ad-hoc mining process of the inner-model knowledge tailored to the models' architecture. Thanks to these, our explanation process is much more efficient and reliable than the model agnostic ones. The data mining approach allows to automatically identify correlated portions of input, according to the model's knowledge. This reduces significantly the search space providing always informative explanations to the end-user.

Of course, the tradeoff between representativeness for the model and interpretability can be relaxed in favor of a higher human understandability. Accordingly to the context of the application indeed it is necessary to investigate also features more related to the expertise of the user. For example, in the context of natural language processing, other than the features extracted from the inner-model knowledge, also portions of a textual document related to Part-of-Speech (e.g. noun, verbs, adjectives) or whole sentences should be investigated. This last typology of features is less related to the knowledge of the model, while highly understandable by human beings. To fulfill also these requirements, we combine the analysis of different typologies of interpretable features not having to renounce neither of the above properties.

More details and examples about the interpretable feature extraction process in each applicative domain are reported in Sections 4.2, and 6.2 for images, and texts respectively.

Perturbation and Evaluation Process. This step allows investigating which is the contribution of each interpretable feature over the predicted outcomes. After the extraction of the interpretable features, an iterative perturbation and evaluation of the

extracted knowledge is performed.

The perturbation of a feature can be performed for instance by adding noise or removing the corresponding portion of the input. This operation of course has to be tailored to each specific input data type. For example, images can be perturbed by exploiting gaussian blur, and textual data with word removal. So, different data domains require different perturbation strategies.

Adding noise to the model input is a well-known technique adopted by different state-of-the-art approaches [2, 47, 69]. This approach aims to study the model reactions to a perturbation of a meaningful portion of the input. However, the real challenge here is to detect the input segmentation that will describe with higher accuracy the behavior of the model. The cited above works are not always able to efficiently and accurately identify multiple regions of the input to perturb. Instead, thanks to interpretable features extracted through unsupervised mining of the inner-model knowledge, our framework perturbs only the portions of input really relevant. Also, this avoids very expensive brute-force or heuristic approaches like [64, 69, 47].

Further details about the perturbation process will be discussed in Sections 4.2, and 6.4 for visual and textual data domains respectively.

Prediction-Local Explanations. At this step, the explanation process produces a human-readable report locally explaining the predictions for a specific input sample. A predictive model may produce multiple predictions with different probabilities. Potentially, each predicted class is a class-of-interest for the user, even if classes predicted with higher probability usually correspond to the most useful ones. Each class-of-interest, predicted by the black-box model, is explained through *qualitative* and *quantitative* prediction-local explanation reports. In other words, the outcome of the explanation process is a detailed report of the whole process composed by:

- A *Visual Explanation* which gives an immediate qualitative idea to the user about the most impacting portions of input on the prediction process.
- A *Numerical Explanation* that provides quantitative details about the influence and of the influence precision that each interpretable feature has on the prediction process.

Our target users are both experts and non-expert users. Both the representations can be very useful from the users' point of view. While experts, i.e. data scientists, have the background knowledge to understand the quantitative results (e.g. complex numerical indexes), non-expert users may not be interested in such deep details requiring only the more direct qualitative explanations.

We tailored the visual explanation process to the data domain to be as much effective as possible. For example, the image prediction process is visually explained through graphical explanations, while the natural language prediction process is more suited to textual explanations. Contrariwise, the numerical explanations of prediction-local

reports are agnostic about the data domain. We introduce two new indexes to efficiently quantify the contribution of each interpretable feature over the prediction process.

The indexes are (i) the normalized Perturbation Influence Relation (nPIR), and (ii) the normalized Perturbation Influence Relation Precision (nPIRP). Given a class-of-interest, the indexes are calculated for each extracted interpretable feature. The nPIR index describes the positive or negative influence of an interpretable feature over the prediction process. Instead, the nPIRP represents the precision with which a feature is influencing the class-of-interest w.r.t. the impact that it has over the whole set of possible classes. For example, a portion of the input may be positively influential for a specific class. However, the same portion of input can significantly influence other classes. This means that the current portion of the input is not precise for the class-of-interest since the model is using the same pattern that is characterizing different classes. These concepts will be deeply discussed in Section 3.2 and the indexes will be formally defined.

Further details about the prediction-local explanation reports are reported in Chapters 4.3, and 6.5 respectively for each data domain.

Per-Class Model-Global Explanations. The main purpose of this component is to identify which are the input patterns that are influencing a whole class-of-interest at a model-global level. Indeed, a model may have been trained with an unbalanced dataset containing any sort of bias. So, discovering if there are input patterns that lead the predictions to a specific class is of high relevance. Our model-global explanations can be computed for a pool of input samples. The larger is the input pool, the higher is the reliability of the per-class model-global explanation.

Model-global explanations are computed by aggregating prediction-local reports. As already introduced, a local explanation for a class-of-interest is composed of a set of interpretable features. Also, each interpretable feature is characterized by nPIR and nPIRP values. So, the joint analysis of the two indexes computed for each interpretable feature of each input sample allows obtaining a more general representation of the model's behavior for a class-of-interest. However, aggregating the local features extracted with the local explanations can be a challenge accordingly to the data domain. Indeed, different data domains have different structural and semantical characteristics. So, at the model-global level, the explanation process has been tailored to address each specific data domain.

The details of each model-global explanation process will be described in Sections 4.4 and 6.6 for visual and textual data domains respectively.

3.2 Measuring the Influence of Interpretable Features

To quantify the contribution of each interpretable feature over the prediction process we introduced two novel indexes. Let us consider the local explanation process for

a class-of-interest. The perturbation of an interpretable feature can influence the model outcome in three different ways:

- The probability of the class-of-interest *increases* → the feature is *negatively* influencing the prediction process;
- The probability of the class-of-interest *decreases* → the feature is *positively* influencing the prediction process;
- The probability of the class-of-interest *remains unchanged* → the feature is *neutral* to the prediction process.

Analyzing these effects we quantify the role of each interpretable feature in the prediction process. The indices introduced for this purpose are:

- the normalized Perturbation Influence Relation (nPIR)
- the normalized Perturbation Influence Relation Precision (nPIRP)

The normalized Perturbation Influence Relation is calculated for each interpretable feature extracted from the input and it represents the positive or negative influence that the perturbed feature has on the outcomes of the Black-Box Model. The nPIR is exploited to evaluate if a specific concept is important for the model or not, since perturbing a semantically significant region of input is like hiding concepts to the model. The index is calculated exploiting the outcomes of the network before and after the perturbation.

Unlike the state-of-the-art solutions [22, 78], our indexes measure both influence and influence precision for an input feature independently of their structured/unstructured nature and in an efficient way.

To formally define the new indexes let's consider a black-box model able to distinguish between a set of classes $c \in C$ and let be $ci \in C$ the *class-of-interest* for which the local-explanation has to be computed. Given the input sample I, the explanation process extracts the interpretable features $f \in F$. For each f the perturbation is applied and the reactions of the predictive model are evaluated. These reactions represent the contribution of f over the prediction process and they will be quantified with the nPIR and nPIRP indexes. Let be $\mathbb{P}_{o,ci}$ the output probability of the original input I (the unperturbed input) to belong to class-of-interest ci, and $\mathbb{P}_{f,ci}$ the probability of the same input, with the feature f perturbed, to belong to the same class. Also, consider the predicted classes as exclusive, i.e. $\sum_c^C \mathbb{P}_{o,c} = 1$ and similarly $\sum_c^C \mathbb{P}_{f,ci} = 1$. For instance, the output of the model is given by a SoftMax layer.

Measuring influence. We introduce a generic definition of influence relation for a feature f by combining the outcomes of the model $\mathbb{P}_{o,ci}$ and $\mathbb{P}_{f,ci}$ before and after the perturbation process. Basically, the model can react to the perturbation of f in three different ways:

- If $\mathbb{P}_{o,ci} > \mathbb{P}_{f,ci} \rightarrow$ it means that f contained a concept that was positively contributing to the prediction of ci.
- if $\mathbb{P}_{o,ci} < \mathbb{P}_{f,ci} \rightarrow$ it means that the pattern in f was negatively affecting the prediction of class ci. In other words, hiding f to the model, the probability of belonging to class ci increases.
- If $\mathbb{P}_{o,ci} \approx \mathbb{P}_{f,ci} \rightarrow f$ was not affecting the prediction process. It can be considered neutral. So, the concept contained in the feature is not relevant for the prediction process.

To measure these effects we introduced the nPIR index. We defined the nPIR to range in the $[-1; 1]$ interval. A nPIR value for f close or equal to 1 represents a positive relevance for the concept in f over the prediction of class ci. On the opposite, a nPIR value for f close or equal to -1 represents a negative impact of that feature over the prediction of class ci. Instead, the more nPIR is close to 0 the more f is neutral w.r.t. the prediction of class ci. The nPIR takes into account both the amplitude of the impact and the relative influence of the perturbation process.

The nPIR derives from the combination of two sub-indicators: the *Amplitude of Influence* ΔI and the *Symmetric Relative Influence* SRI. The ΔI for a feature f is measured by:

$$\Delta I_f = \mathbb{P}_{o,ci} - \mathbb{P}_{f,ci} \tag{3.1}$$

It ranges from -1 to 1 since the domain for probability values is included in [0,1]. A $\Delta I_f > 0$ represents a positive influence of the feature f for class ci, since the perturbation of the corresponding portion of input causes a decrease of its probability to belong to the class-of-interest. Thus, f is relevant for class ci. Similar reasoning could be made for $\Delta I_f < 0$ representing a negative influence of the feature f for ci. However, the amplitude does not reflect properly the contribution of f. In particular, the absolute distance between two values can be low if the values are small w.r.t. the probability values domain, but still, their relative distance can be significant. This effect should not be ignored as well. Thus, we consider also the relative influence of f. The ratio between the probabilities, as shown in [99], will result in an asymmetric evaluation of the relative influence: the ratio $\frac{\mathbb{P}_{o,ci}}{\mathbb{P}_{f,ci}}$ will range from 0 to 1 in case of negative influence and from 1 to ∞ in the other case. So, it will be hard to quantitatively compare positive and negative influences. To solve this problem we defined the *Symmetric Relative Influence* for a feature f:

$$SRI_f = \frac{\mathbb{P}_{o,ci}}{\mathbb{P}_{f,ci}} + \frac{\mathbb{P}_{f,ci}}{\mathbb{P}_{o,ci}} \tag{3.2}$$

This index takes into account, in a symmetric way, the relative influence that f has over $\mathbb{P}_{o,ci}$ and $\mathbb{P}_{f,ci}$. The symmetry of this relation allows measuring the relative influence of the feature f before and after the perturbation regardless of its positiveness or negativeness.

Then, combining Equations 3.1 and 3.2 we define the *Perturbation Influence Relation* for f:

$$\begin{aligned} PIR_f &= \Delta\, I_f * SRI_f \\ &= \mathbb{P}_{f,ci} * \beta - \mathbb{P}_{o,ci} * \alpha \\ \text{with } \alpha = 1 - \frac{\mathbb{P}_{o,ci}}{\mathbb{P}_{f,ci}}&, \beta = 1 - \frac{\mathbb{P}_{f,ci}}{\mathbb{P}_{o,ci}} \end{aligned} \tag{3.3}$$

The coefficient α represents the contribution of the original input concerning the perturbed one.Similarly, β represents the contribution of the perturbation of f concerning the original input. The PIR index ranges in $(-\infty, +\infty)$ in a symmetric way going from negative to positive impact. However, despite particular cases, PIR values are included in a much smaller interval. The PIR index is thus normalized exploiting the well known *Softsign* [29] transformation and redefined as the normalized Perturbation Influence Relation(nPIR):

$$nPIR = softsign(PIR) \tag{3.4}$$

The *Softsign* function allows normalizing PIR index in $[-1; 1]$ range for strong negative or strong positive impacts respectively while highlighting the differences of values close to 0 with a linear behavior.

In the normalized Perturbation Influence Relationwe have both the advantages discussed for DI_f and SRI_f represented in a more confident domain. The nPIRindex measures if the influence of f is positive or negative taking into account both the amplitude and the relative impacts in a symmetric way. It ranges in $[-1; 1]$ going from negative to positive impacts respectively.

Exploiting a human validation of our explanations (more details about human-validation in Chapter 5.7) it has been noticed that users perceive a feature to be very positively influential when its nPIR is equal or greater than 0.75 on average. On the other side, a feature is perceived as negatively influential when nPIR is lower than -0.20. Thanks to the symmetry of the index, a feature f can be generally considered:

- *Positively Influential* if $nPIR_f \geq 0.20$;
- *Negatively Influential* if $nPIR_f \leq 0.20$;
- *Neutral* if $|nPIR_f| < 0.20$.

Also, if $|nPIR_f| >= 0.75$, the influence of f can be considered strong and it can not be ignored.

Measuring influence precision. Other than influencing the class-of-interest, each interpretable feature can have a positive contribution to multiple classes. Under some circumstances, the same input pattern can actually belong to two or more classes, while other times this is not correct. The influence of a feature f on the prediction process can be categorized in:

- *Positively Precise* → the class-of-interest is the only one affected by the perturbation of f;
- *Not Precise* → the perturbation of f is equally affecting positively the class-of-interest and at least another class;
- *Negatively Precise* → the perturbation of f is affecting positively any of the other classes more than the class-of-interest.

So, it is necessary to study the precision with which each interpretable feature is influencing the decision-making process of the model and to quantify how precisely each concept has been learned. The more a concept impacts a wide range of classes the less it can be considered accurate for the class-of-interest. As a consequence, if the concept contained in a feature is impacting more than one class, it means that the model has associated that specific pattern to a wide range of outputs. This can be considered, in some cases, misleading knowledge gained by the model during the training phase, caused by the presence of bias in the dataset and/or by a bad design of the network. Thus, for a specific feature f it is necessary to study nPIR values for all the classes known by the model, computing the influence precision of the feature over the entire prediction process.
To measure the precision of the influence we introduced the normalized Perturbation Influence Relation Precision (nPIRP). The definition of normalized Perturbation Influence Relation Precision follows similar reasoning made to define Equation 3.3. The nPIRP evaluates the precision of the absolute impact of f over class ci, i.e., ξ_{ci}, w.r.t. its total positive impacts over classes $C \setminus ci$, i.e., $\xi_{C \setminus ci}$. the two components ξ_{ci} and $\xi_{C \setminus ci}$ are defined as:

$$\xi_{ci} = \mathbb{P}_{o,ci} * |nPIR_{ci}| \tag{3.5}$$

$$\xi_{C \setminus ci} = \sum_{c}^{C \setminus ci} p_{o,c} * max(0, nPIR_c) \tag{3.6}$$

As the distribution of the predicted probabilities may be very sparse, the two measurements of influence are weighted by the probability of the original input to belong to each corresponding class. So, the influences of the most probable classes are taken into greater consideration w.r.t. less probable ones.

Combining Equations 3.5 and 3.6 we defined the Perturbation Influence Relation Precision of a feature f as:

$$\begin{aligned} PIRP_f &= \Delta I_f(\xi_{ci}, \xi_{C\setminus ci}) * SRI_f(\xi_{ci}, \xi_{C\setminus ci}) \\ &= \xi_{C\setminus ci} * b - \xi_{ci} * a \\ \text{with } a = 1 - \frac{\xi_{ci}}{\xi_{C\setminus ci}}&, \; b = 1 - \frac{\xi_{C\setminus ci}}{\xi_{ci}} \end{aligned} \tag{3.7}$$

Similarly to Equation 3.4, we normalize Equation 3.7 exploiting the *softsign* function to get the normalized Perturbation Influence Relation Precision index:

$$nPIRP = softsign(PIRP) \tag{3.8}$$

The nPIRP is computed for each feature and it ranges in $[-1; 1]$. The nPIRP metric measures the precision of the influence that feature f has over the whole set of predicted classes w.r.t. the class-of-interest ci. When f is very precise in describing ci the nPIRP has a value close to or equal to 1. Instead, if the feature is impacting more on other classes than the class-of-interest the nPIRP is close or equal to -1. In the case of values close to 0 the precision of f can be considered neutral. In other words, the influence of f over ci is comparable to the influence that it has over other classes in $C - ci$.

As a general guideline, the nPIRP should be considered after analyzing the nPIR. Indeed, if the nPIR is significant, then the nPIRP carries meaningful information about the precision of the target concept. If $nPIRP = 1$ it means that the influence of f is all concentrated on the class-of-interest. In the opposite case, i.e., $nPIRP = -1$, the influence of f is irrelevant to the class-of-interest w.r.t. the influence that it has over all the other classes.

Chapter 4

Explaining the Convolutional Decision-Making Process

This chapter describes the methodological details of the explanation process carried out to explain image classification through Deep Convolutional Neural Networks (DCNNs). Deep Convolutional Neural Networks have been introduced in the last years but they already reached important milestones in computer vision tasks. However, DCNN models are considered black-boxes due to the complexity of understanding their inner functioning. In this Chapter, we define a novel explanation process applied to deep convolutional models.

Deep Convolutional neural networks [80, 89, 87] (DCNNs) are a family of deep learning models that exploit convolution filters to process data with grid-like topologies. Thanks to the convolution operation, these kind of networks can be applied to many different kinds of data such as time-series, images, texts, and audio signals. The proposed framework is able to unsupervisely mine the decision-making process of convolutional models. through the unsupervised mining of the inner knowledge contained in multiple layers of a DCNN. Differently from other explanation strategies [69, 76], it provides both prediction-local and per-class model-wise explanations. Moreover, we provide both visual and quantitative explanations. This enables wider applicability of the approach in different contexts. Both expert and non-expert users can benefit from our comprehensive explanations to better understand the reasons behind the prediction process.

In particular, this chapter includes:

- The prediction-local and per-class model-global explanation process is tailored to DCNN.
- The interpretable feature extraction process tailored to visual input data, which is easily understandable by humans and meaningful for the model under exam.
- The definition of quantitative and qualitative explanation reports, measuring the

influence of each interpretable feature on the local outcome provided by the black-box model.

- The new visual class-based model-global explanation report tailored to the image classification by aggregating quantitative prediction-local explanations.

The rest of the chapter is organized as follows. Section 4.1 gives an overview of the explanation process tailored to Deep Convolutional Neural Networks. Section 4.2 describes how interpretable features are extracted in the case of convolutional layers. Section 4.3 provides details about the local explanation process in the image classification use-case. Finally, Section 4.4 describes how the local explanations are aggregated to provide per-class model global explanations.

4.1 Convolutional Explanation Process Overview

Given an input image and a DCNN predictive model, the proposed framework provides a detailed prediction-local explanation of the black-box outcome. Figure 4.1 shows the main steps of the explanation process of a convolutional model. An image is given as input to the DCNN model (Step ①). The model produces a set of predicted class labels along with their probabilities (Section ⑤a). As the prediction process proceeds, the inner-knowledge of the model is extracted and analyzed. Hypercolumns are extracted (Step ②) and processed to obtain a set of interpretable features (Step ③). More details in Section 4.2. The contribution of each interpretable feature over the prediction process is then measured by means of an efficient iterative perturbation and prediction (Step ④). More details in Section 4.2. Finally, visual and numerical local explanations are produced (Step ⑥), as detailed in Section 4.3.
A sample of the explanation process is provided in Figure 4.2, showing an input image (left), the interpretable feature extraction result (center), and the local explanations, both visual and numerical (right).

4.2 Interpretable feature extraction and perturbation

One of the main challenges when dealing with explanations is to make sure that they are actually related to the model under analysis. To address this challenge we want to exploit the knowledge already included in the deep models during their training. Indeed, Neural Networks have the great power to learn how to automatically extract the most relevant features from an input.

Thus, our objective is to identify which are the portions of the input that mostly contribute to the prediction process. However, the contribution of single-pixels does not provide interpretable results and is computationally demanding. On the other side,

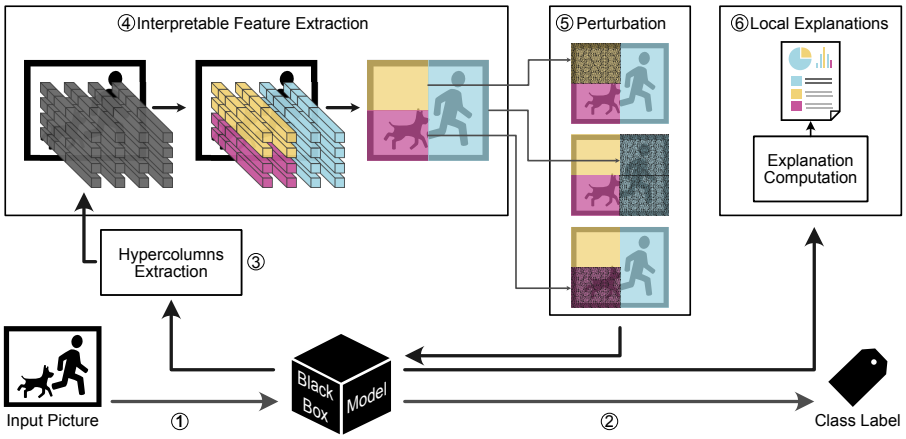


Figure 4.1: Local explanation process.

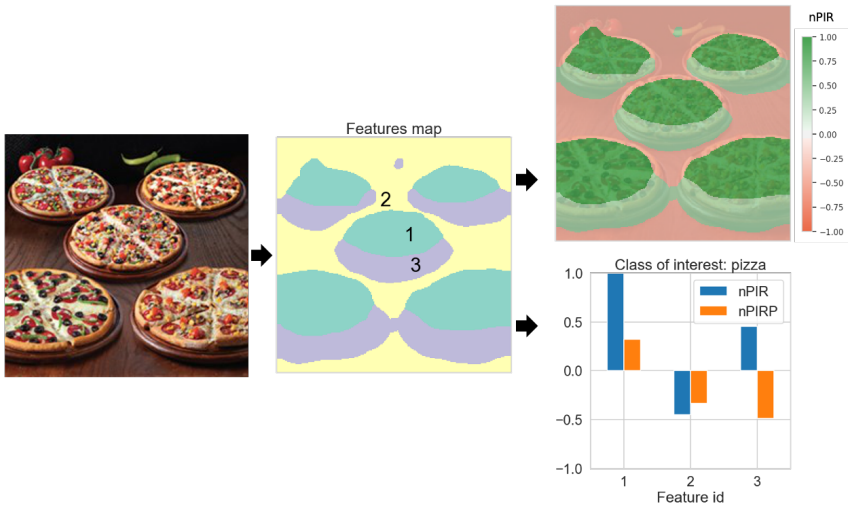


Figure 4.2: Local-explanation example for class-of-interest *Pizza*. Input image (left), interpretable features (center), visual explanation (right-top), quantitative explanation (right-bottom).

identifying groups of pixels without taking advantage of the model has multiple drawbacks. Identifying portions of the image really meaningful for the model can be computationally very expensive. This can require testing a very large number of combinations and even in this case, one can not be sure that they are really meant for the model and the end-user.

Instead, we aim to *identify the most informative partitioning of input pixels for the model under analysis and for the end-user.*. To reach this goal we exploited a mining process of the knowledge contained in multiple hidden layers of the network. In this way, we can identify sets of correlated pixels mostly influencing the outcome of the black-box model and yielding more understandable results for humans.

The proposed feature extraction strategy identifies meaningful and interpretable portions of an input image by analyzing in an unsupervised fashion the extracted *hypercolumns* [34], which are a vectorial representation of the model across all its inner levels. The image segments extracted with this strategy are called *interpretable features* because of their immediate understandability. An example of *interpretable features* is reported in Figure 4.2 (center picture). The convolutional-layers' information has been clustered, producing a set of image portions providing different semantic aspects learned by the model. Specifically, the pizzas on a table have been segmented into 3 features: i) the inner part of the pizzas (feature 1 in green), ii) the table under the pizzas (feature 2 in yellow), and iii) the border of the pizzas (feature 3 in purple). As required by the proposed local explanation process, the extracted features are both numerically quantifiable for the model under exam and interpretable by humans.

Hypercolumns extraction. *Hypercolumns* have been defined by [34] as a vectorial representation of every input pixel. This vectorial representation has the ability to collect the information related to a specific location of an input image across all the layers of a DCNN. Then the latent information under the form of *hypercolumn* can be mined, allowing us to understand if bias or knowledge has been learned by the black-box model. This allows combining feature extraction of multiple levels of a DCNN making the explanation process aware of the real knowledge learned by the model.

When an image is given as input to a DCNN (Step ① in Figure 4.1), the model is internally extracting the features necessary to understand its semantic content. In the specific case of DCNN, it is widely known that the first layers of the DCNN are able to generalize over the shape of objects, identifying corners and edges, while the last layers are more sensitive to the semantic meaning of an image [8, 51, 34]. The hypercolumns of a specific input can be extracted feeding into the black-box model the target image: each convolutional layer of the network outputs a tensor that is the result of the application of the weights learned by the model. These tensors contain all the latent information learned by the model during the training phase.

Let us provide an example in Figure 4.3. Consider a DCNN model taking as input an image with shape $224 \times 224 \times 3$ and its last 5 convolutional layers have as outcome 2 tensors with shape $28 \times 28 \times 512$ and 3 tensors with shape $14 \times 14 \times 512$. The final shape of the hypercolumn tensor, extracted from these five layers of the network, will be $224 \times 224 \times 2560$ (i.e. $512 * 5 = 2560$). The first two dimensions of the tensors extracted by the convolutional layers are upscaled with bilinear interpolation to fit the first two dimensions of the input image; then, the different layers are concatenated by the third dimension composing the hypercolumns for each input pixel.

In the case of a very deep network, using all the convolutional layers to extract the hypercolumns can produce a very deep tensor which is difficult to manage. However, our methodology focuses on the most characterizing information of the model, which is usually included in the deepest layers of the network, i.e., the deeper the layer, the

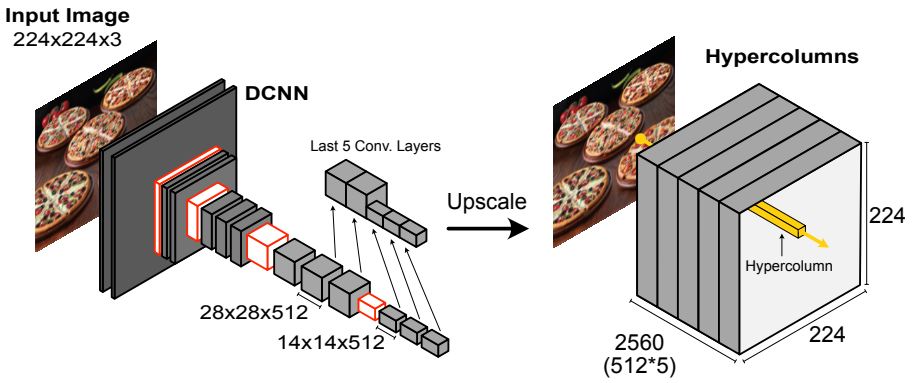


Figure 4.3: Hypercolumn extraction example. The DCNN is a VGG16 like model. The input image is $224 \times 224 \times 3$. The last 5 convolutional layers have been use to extract the hypercolumns.

more specialized it is, and the information that can be extracted from it is very task-specific. For this reason, the number of layers that should be considered is usually much lower than the total number of available layers in the network. The number of layers exploited to extract hypercolumns is a parameter that remains empirically configured, being related to target the network's architecture.

Feature extraction. Hypercolumns incorporate the information about the relations that the DCNN has learned to exist between different pixels, hence identifying the portions of an image semantically similar, as detected by the network. The interpretable feature extraction is performed through a segmentation process of the input image, aware of the knowledge learned by the model. Grouping correlated hypercolumns together can lead to meaningful explanations for humans, hence the resulting image segments have been called *interpretable features*. Moreover, it is necessary to identify correlated beams of hypercolumns to maintain the analyzed portion of input significant for the black-box model as well. The K-Means algorithm [46] has been exploited for the clustering analysis, evaluating the Euclidean distances among the hypercolumns.

Through an unsupervised clustering analysis of hypercolumns, we can identify correlated beams of vectors to subsequently detect the input areas to which they correspond. It projects the grouped beams of hypercolumns on the input image by labeling each pixel with its cluster. Figure 4.4 shows an example of hypercolumns' partitioning in 3 clusters. The clustering of hypercolumns is not lead by the position of the pixels in the image. This image segmentation strategy, differently from others, does not consider input colors or pixel location. Instead, it is strictly related to what the model has learned. Indeed, only the weights learned by the model are used during the partitioning, driving the explanation with the inner information of the model itself. If the image contains correlated hypercolumns that are distant in the original input image (pixel-wise), they will be partitioned together, through the analysis of hypercolumns

This property is desired. Also, semantically-meaningful image segmentation depends only on the training of the model. This improves the reliability and understandability of the produced explanations.

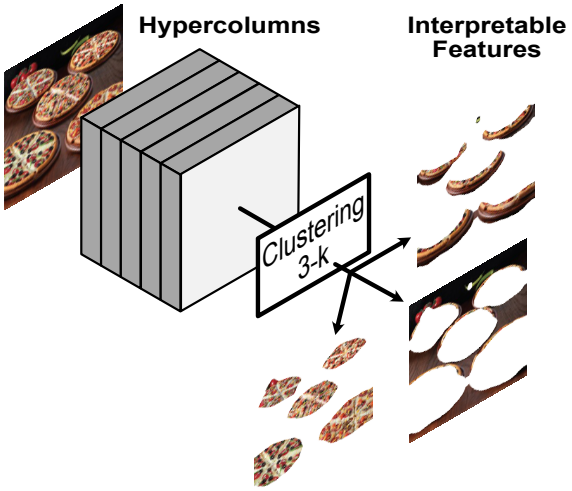


Figure 4.4: Example of interpretable feature extracted by hypercolumns' partitioning in 3 clusters.

When dealing with clustering analysis, a critical parameter that needs to be evaluated is the number of interpretable features to extract, i.e., the number of partitions k. Measuring the quality of the partitioning with well-known indexes such as SSE and Silhouette, would not be effective in our case. Indeed, these indexes do not take into account our task. They consider only the geometrical error in assigning points to each cluster. Instead, we want to choose a partitioning that contains as most information as possible for the model and for the user. In practice, we aim to present to the end-user the *Most Informative Local Explanation* possible.

Figure 4.5 shows the proposed feature extraction along with the selection of the most informative local explanation. We recall that we are measuring the influence of each interpretable feature over the prediction process through the nPIR index presented in Section 3.2. As previously explained, the target input image (Step ①) is given as input to the black-box model, and the hypercolumns are extracted (Step ②). Then, the clustering algorithm is used (Step ③) to extract a number of k partitions. The number of partitions k, which corresponds to the number of resulting interpretable features, ranges in the interval $k \in K = 2, \dots, k_{max}$. For each $k \in K$, a local explanation e is computed (Step ④). Each explanation $e_k \forall k \in K$ is characterized by $f = k$ interpretable features, and F is the set of all the interpretable features computed for e_k. The maximum number of features k_{max} has to be set so that f is small enough to be manually inspected by human users but large enough to avoid missing details and diversity. Finally, among the set of potentially useful explanations $e_k \forall k \in K$, the process identifies the most

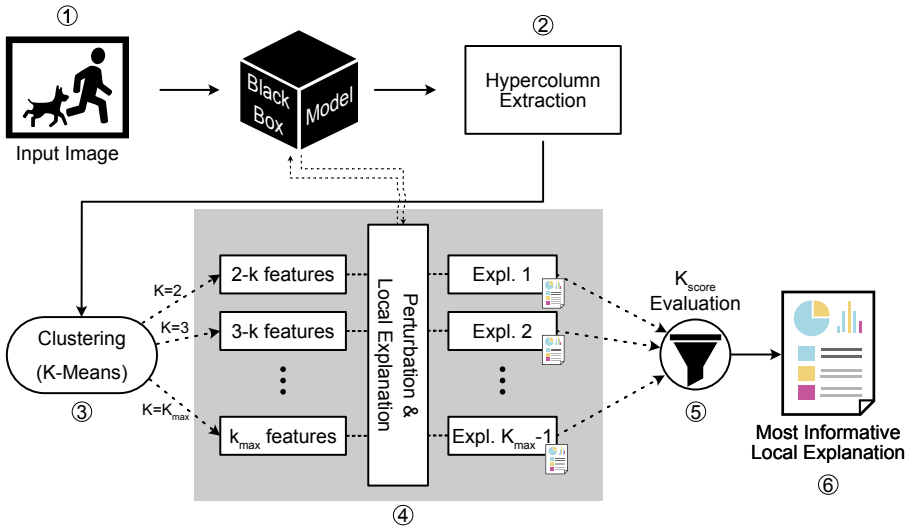


Figure 4.5: Feature extraction process and selection of the most informative explanation.

informative explanation (Step ⑤). The most informative explanation is defined as the one that maximizes the contrast among the features' influence. Each explanation e_k is characterized by the score K_{score} that measures the information contained in the explanation itself. It is defined as:

$$K_{score}(e_k) = \max_{f \in F}(nPIR_f(e_k)) - \min_{f \in F}(nPIR_f(e_k)) \tag{4.1}$$

Then, the *Most Informative Local Explanation* $\hat{e}$ is identified (Step ⑥):

$$\hat{e} = \max_{e_k \forall k \in K}(K_{score}(e_k)) \tag{4.2}$$

This allows to provide to the end-user the best explanation in terms of information and understandability: the wider the K_{score} is, the more contrast there will be among the interpretable features.

Perturbation Theoretically, each *interpretable feature* represents a relevant concept contained in the input image and its role in the prediction process strictly depends on the training process of the model. Here the challenge is to identify the most relevant portions of the input to perturb with the aim of obtaining a significant reaction from the predictive model. The perturbation of the model's input is a well-known state-of-the-art strategy adopted by several related works [2, 99, 47, 69] to study the impact of input data on the prediction outcome. Usually, the main challenge is to define relevant portions of input to perturb to obtain a valuable reaction from the predictive model. However, thanks to the proposed interpretable features extraction process, the meaningfulness of the input regions to analyze is guaranteed. This lowers the complexity of the iterative perturbation process and increases the information that can be gained

from the reaction of the predictive model when a perturbed image is given as input. Indeed, the perturbation is performed on the specific pixels of the *interpretable features* and measure the prediction difference on those concepts.

We exploited an iterative perturbation process based on Gaussian blur. A new perturbed image is produced for each blurred interpretable feature (Step 5 in Figure 4.1), and we expect the model to miss the recognition of the corresponding concept; three results are possible: (i) no change in prediction (the concepts represented by the feature were not relevant for the predicted class); (ii) stronger prediction (the probability of belonging to the predicted class increases after the perturbation, hence, removing the feature, the predicted class is better modeled); (iii) weaker prediction (the probability of belonging to the predicted class decreases after the perturbation, hence the feature was important to model the class).

4.3 Prediction-Local Explanations

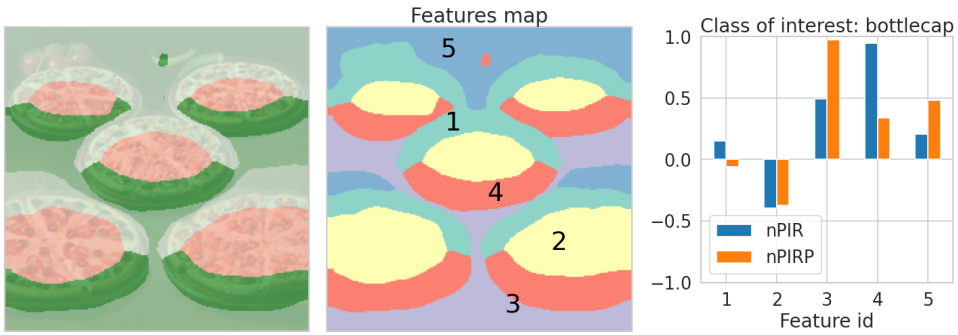
Comparing the probability distributions before and after each perturbation, our explanation framework is able to study the influence of each *interpretable feature* on a specific predicted class, producing a *local explanation* (Step 5b in Figure 4.1) with a *numerical* contribution and a *visual* part. Let introduce a running example. Figure 4.6 shows the input image for which a pre-trained VGG16 DCNN predicts the classes in Table 4.1. The model predicts the wrong *Bottlecap* class as most probable followed by the correct one *Pizza*. Figure 4.7 shows the visual and numerical local explanations proposed for the two classes-of-interest.

Figure 4.6: *Pizza* input image

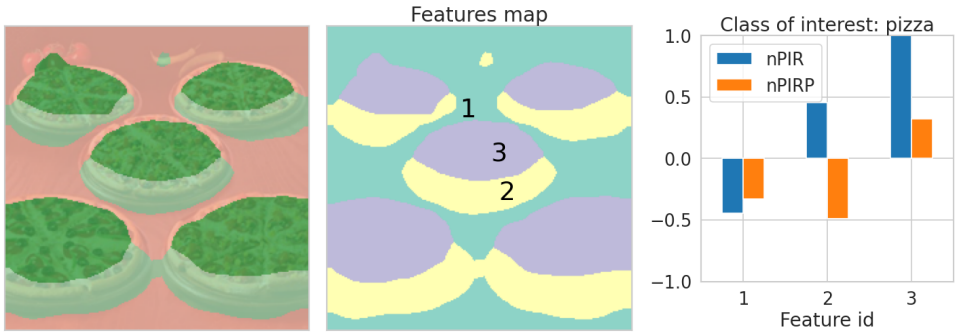
Class	$P(c)$
Bottlecap	0.42
Pizza	0.28
Bakery	0.08
Trifle	0.06
Dining table	0.03

Table 4.1: Predictions for Figure 4.6 with a pre-trained VGG16.

Quantitative explanation Providing a quantitative measure of relevance for each feature is of mandatory importance in the explanation process. It allows the user to objectively inspect the details of the prediction process. Differently from other works [47,



(a) Explanation of prediction *Bottlecap* (VGG16 in Table 4.1).



(b) Explanation of prediction *Pizza* (VGG16 in Table 4.1).

Figure 4.7: Example of local explanations for two classes of interest. The input image is shown in Figure 4.6. Each explanation is organized with the Visual explanation (left), the map of features (center), the quantitative explanation (right).

69], our indexes (i) efficiently measure the influence relation that exists between the input feature and the model outcomes in terms of neutral, positive, or negative impact, and (ii) consider the influence precision of the features for the class-of-interest in a multi-class problem.
We recall that the influence of a feature over the class-of-interest can be considered:

- *positive* if the predicted probability decreases after the perturbation, meaning that the feature was positively relevant for model;
- *neutral* if the predicted probability remains the same after the perturbation, meaning that the feature was irrelevant for model;
- *negative* if the predicted probability increases, meaning that the feature was negatively relevant for model.

Also, the distribution of the probabilities for the other classes can change accordingly to the perturbed feature. The precision of influence of a feature can be considered:

- *precise* if the class-of-interest is the only one affected by the perturbation process of the feature;
- *not precise* if the perturbation of the feature is affecting equally the class-of-interest and at least another class;
- *negatively precise* if the perturbation of the feature is affecting more any of the other classes other than the class-of-interest.

To quantify these conditions we introduced two new indexes in Section 3.2: normalized Perturbation Influence Relation (nPIR) and normalized Perturbation Influence Relation Precision (nPIRP). Analyzing together these two indexes, a detailed report about the prediction process is produced.

Considering the numerical local explanations in Figures 4.7a-right and 4.7b-right, it is possible to understand the reasons for the wrong prediction. The bottom borders of the pizzas (feature 4 in Figure 4.7a-right) are very influential for class *Bottlecap* and also the underlying table in the background (features 3 and 5) is very precise for it. On the other side, the class *Pizza* is predicted mainly because of the toppings of the pizzas (feature 3 in Figure 4.7b-right). Indeed the corresponding features resulted both positively influential with a very high nPIR value and precise with a positive nPIRP value.

Visual explanation Along with the detailed quantitative explanation, our explanation framework provides an easy-to-understand prediction-local visual explanation. In the visual explanation, each interpretable feature is colored with a red-green gradient. It goes from green (positive influence) to red (negative influence), passing through white, according to the value of nPIR. The more a green area is intense, the more the corresponding feature is positively influential for the class-of-interest. On the contrary, the more a red area is intense, the more the feature is negatively impacting the class-of-interest. White areas instead, which results almost transparent, show input portions that have a neutral impact on the prediction process, i.e. the model is completely independent of the presence of these features.
An example of a visual explanation is reported in Figures 4.7a-left and 4.7b-left for the classes-of-interest *Bottlecap* and *Pizza* respectively. The visual explanation clearly shows which are the areas of the image that have been mostly positively impacting the prediction process reflecting the results already discussed with the quantitative explanation.
Differently from other works [76, 79, 27, 107, 64], the proposed visualization is not based on saliency maps. A saliency map is a simple and clear visualization strategy that smoothly shows the relevance of contiguous areas of pixels but it does not allow to differentiate the influence of multiple input areas at the same time. Instead, our approach, as shown in the example, can highlight the impacts of more input regions simultaneously, with their positive and negative contributions, including more information in a

single visual representation. The visual explanation represents a more direct way to communicate the explanation of the user, even if losing the details related to the influence precision. Non-expert users can take advantage of this visual representation to have an immediate result, while expert users may prefer to consider also the detailed numerical report.

4.4 Per-Class Model Explanations

A model-global explanation is usually exploited to study the influence of a specific concept on the whole prediction set provided by the model, to detect possible bias, for instance. In data domains like tabular data or textual data, the explanation process takes advantage of well-defined features, i.e. columns and word tokens, that are simple to aggregate in model-global explanations. Works like [47, 69] study the behavior of the model aggregating the explanations produced for singular predictions by feature meaning.

In the case of image inputs, instead, the DCNN model processes each pixel. As previously discussed, single-pixel explanations are useless for humans. Hence, we group prediction-local explanations according to *interpretable features*. Analyzed together, nPIR and nPIRP describe the influence, in terms of both the contribution and the precision, of each *interpretable feature* of an input image on the prediction process. This information enables the explanation process to identify behavioral patterns of the model w.r.t the prediction of each class.

The model-wise challenge is to aggregate the interpretable features belonging to different images by their semantic meaning, without using another supervised model. To this aim, we provide an unsupervised class-based model explanation by aggregating the prediction-local features according to their class-of-interest. Then, each class-of-interest is described by all the features, extracted during the local-explanation process, exploiting their nPIR and nPIRP values. The features are projected on the $nPIR \times nPIRP$ space and studying their distribution allows the user to inspect the class-wise behavior of the model during the decision-making process.

Figure 4.8 shows an example of a class-based model explanation for the class-of-interest *Pizza* computed for a VGG16 model, aggregating three local-explanations of three different input images. The figure shows the interpretable features distributed in the $nPIR \times nPIRP$ space and their KDE distributions (Kernel Density Estimation) on the nPIR and nPIRP axis. The plot groups the features in the four quadrants. The global explanation report can be analyzed following:

- $nPIR \geq 0$; $nPIRP \geq 0$: the features in this section are both positively influential and precise for the class-of-interest.
- $nPIR < 0$; $nPIRP \geq 0$: the features in this section are negatively influential but precise for the class-of-interest.

- $nPIR < 0; nPIRP < 0$: the features in this section are both negatively influential and not precise for the class-of-interest.
- $nPIR \geq 0; nPIRP < 0$: the features in this section are influential but not precise for the class-of-interest.

On the top and right axis, the KDE distributions of the features considering nPIR and nPIRP are reported.

The *optimal distribution of features* for a model would be when all the image segments that are representative for the class-of-interest are positioned on the top-right corner with $nPIR = 1, nPIRP = 1$, and all the other features are close to the center with $nPIR = 0, nPIRP = 0$ meaning that the contextual features are not influencing the decision-making process. The presence of features spread around the plot means that the model can be considered uncertain about the role of them in the prediction process. The plot easily enables human experts to quickly drive their evaluation towards specific features for a semantic assessment of the model behavior.

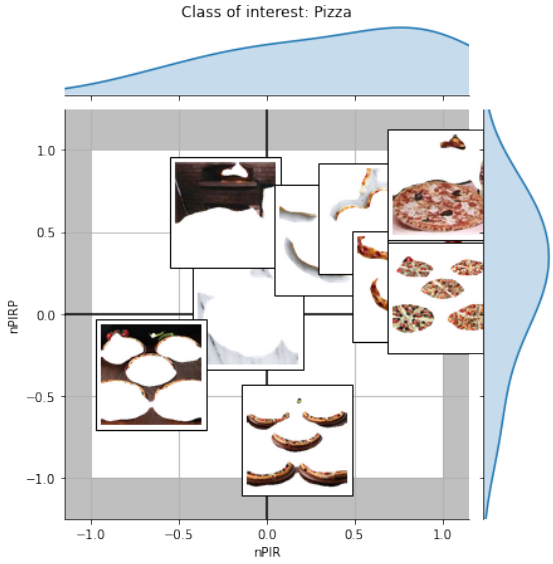


Figure 4.8: Class-based model explanation example for class-of-interest *Pizza*.

Chapter 5

Experimental Results on Deep Convolutional Models

In this Chapter, we provide experimental evidence of the effectiveness of our explanation techniques when applied to DCNN. We provide a quantitative validation of the proposed approach, producing almost 10,000 explanations for four state-of-the-art DCNNs on 250 input images. We compare our explanation framework with state-of-the-art explanation tools, i.e., *LIME* [69], *GRAD-CAM* [76], and *Shapley values* [85]. Then, we provide a qualitative human validation of the effectiveness and easiness of understanding of our explanation framework. This last result has been obtained by developing an online survey and collecting users' feedbacks.

Section 5.1 introduces the experimental setup. Section 5.2 describes relevant prediction-local explanations. Section 5.3 shows the results of the per-class model-global explanation process. Section 5.4 provides a performance comparison between us and the state-of-the-art. Section 5.5 discusses the performance of the nPIR index across all the tested images and models, and then Section 5.6 compares our index with the state-of-the-art *Shapley values*. Finally, Section 5.7 reports the human-validation process assessing the interpretability of the explanations concerning the state-of-the-art.

An interactive library of explanations produced by the proposed framework is available online[1] as well as the online survey proposed to the users[2].

5.1 Experimental Setup

We tested the explanation process on four different pre-trained DCNN models[3]:

[1] https://ebano-ecosystem.github.io/#explanation-library

[2] https://ebano-ecosystem.github.io/#ebano-survey

[3] The models are available in the Keras deep learning library [20], version 2.2.4

- (M1) VGG16 [81],
- (M2) VGG19 [81],
- (M3) InceptionV3 [90],
- (M4) InceptionResNetV2 [88].

All the models are pre-trained on 1000 classes of the ImageNet [72] dataset. The explanation process has been applied to explain a set of 250 input images collected from different datasets, i.e. Coco [45], ImageNet [72], Caltech[43], and web scraping. The set of tested images contains 54 different classes. The top-10 predicted classes of each image represent the classes-of-interest analyzed, for a total of 10,000 prediction-local explanations. The input images have never been used during the training phase of the corresponding models.

The hypercolumns are extracted from multiple inner convolutional layers of each model. The number of convolutional layers analyzed for each model has been experimentally set. Models M1 and M2 are relatively small DCNNs. We extracted the last 5 and 8 convolutional layers for M1 and M2 respectively. Instead, models M3 and M4 have a more deep and complex structure. For those models, we considered the last 34 and 24 convolutional layers. These settings represent a fair trade-off between feature interpretability and execution complexity.

We experimentally set the number of interpretable features to extract from each input image to range between 2 and 10. We set the upper limit to avoid too small features with no semantic meaning for the user. Each explanation is characterized at most by 10 features. The *most informative explanation* (Section 4.2) is automatically proposed to the user.

5.2 Evaluating Prediction-Local Explanations

We show here the explanation reports obtained for two representative input images I1 and I2. Figure 5.1 shows the input image I1 representing a mouse over a tailed surface. Input I1 is analyzed by models M1 and M4. Model M1 outcomes are shown in Table 5.1. Model M4 predictions are shown in Table 5.2.

Figure 5.3 shows input image I2. Input I2 shows a pizza with an uncommon heart shape. Models M2 and M3 are used to predict the classes for this image. Outcomes of the models are reported in Tables 5.3 and 5.4 respectively. In this case, the two models are apparently not influenced by the shape of the pizza. Indeed, M2 and M3 correctly predict class *Pizza*. However, to assess the reliability of models M2 and M3 a deeper investigation is required.

We aim to unwrap the black-box models by analyzing detailed explanations to answer the following questions:

- **Q1.** "Why is Figure 5.1 representing a *Toilette seat* for model M1?"
- **Q2.** "Why is Figure 5.1 not a *Mouse* for model M1?"
- **Q3.** "Why is Figure 5.1 a *Mouse* for model M4?"
- **Q4.** "Why Figure 5.3 is a *Pizza* for model M2?"
- **Q5.** "Why Figure 5.3 is a *Pizza* for model M3?"

A larger number of prediction-local explanations for the four models is publicly available and it can be explored through an interactive web-based tool[4].

Answering Q1. Model M1 wrongly predicts the class for I1. The most probable class is indeed *Toilet seat*, while the correct class *Mouse* has a much lower probability. Analyzing the probability distribution, it is possible to guess that model M1 confused the mouse for a toilette object. However, we want to answer the question "Why" this is happening.

Figure 5.2a shows the explanation for model M1 and the class-of-interest *Toilet seat*. The explanation identifies the 9 features in Figure 5.2a-center for the most informative explanation. Figure 5.2a-left shows the visual explanation and Figure 5.2a-right shows the quantitative one. We recall that the visual explanation highlights in green and red the features that are positively and negatively influential respectively. From the explanation, it is clear that the prediction *Toilet seat* is caused by the presence of the horizontal lines between the tiles. Hence, the prediction *Toilet seat* has been taken because of contextual information and not because of the subject itself.

The numerical explanation in Figure 5.2a-right shows the values of PIR and PIRP for each extracted feature. It confirms that the lines between tiles, i.e., feature 6, are the most positively influencing. Also, the corresponding PIRP value is close to 0. This means that the feature is not precise: it is a contextual feature influencing other classes. Moreover, inspecting the body of the mouse, i.e. feature 7, we noticed that it is neutral for the class-of-interest *Toilet seat* with a PIR value near 0. its PIRP value is very negative, meaning that it has a much larger influence on other classes than *Toilet seat*.

Answering Q2. Figure 5.2b shows the explanation for the class-of-interest *Mouse* predicted by model M1. The most informative explanation is composed of 4 interpretable features in Figure 5.2b-center. From the visual explanation in Figure 5.2b-left the *Mouse* is correctly positively represented by feature 1. However, the PIRP of feature 1 is negative, as highlighted by the numerical explanation in Figure 5.2b-right. The case of positive PIR and negative PIRP clearly characterize the prediction process. Even if

[4] https://ebano-ecosystem.github.io/#explanation-library

feature 1 is positive for the class *Mouse*, other classes are more influenced so this prediction is confirmed to be not reliable. Instead, we can consider correct the negative influence of the background tiles, i.e. feature 4 with negative PIR and PIRP.

Thanks to the explanations we discover that the prediction process has been heavily affected by the context of the subject. Model M1 correctly distinguishes among the different concepts in the input. However, it takes into account more the context than the subject.

Answering Q3. Model M4 predicts correctly the class *Mouse* for input I1. The most informative prediction-local explanation, with 5 features, is shown in Figure 5.2c. The body of the mouse, i.e., feature 1, is the only one affecting the prediction process with $nPIR \approx 1$ and $nPIRP = 1$. All the other features are not influential. Contrary to the previous model, M4 is more focused on the subject. All the features surrounding the mouse, i.e., the context, are not influential showing $nPIR = 0$. Its predictions can be considered generally reliable and it can be trusted with higher confidence than M1: it is not by chance that the prediction is correct.

Also, comparing the explanations obtained for models M1 and M2, the differences in the prediction processes emerge allowing the user to decide which is the most reliable one and explaining why it can be trusted.

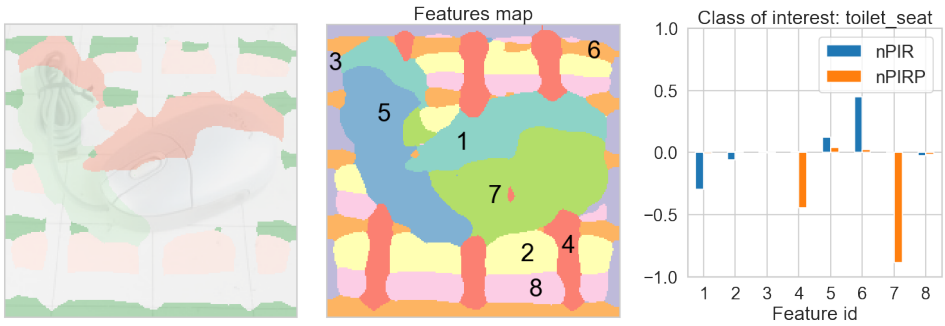
Figure 5.1: *Mouse* input image (I1)

Class	$P(c)$
Toilet seat	0.23
Mouse	0.15
Soap dispenser	0.11
Washbasin	0.11
Can opener	0.06

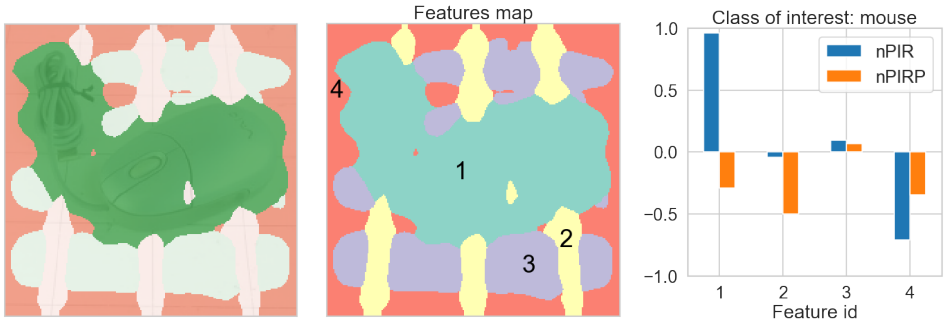
Table 5.1: VGG16 (M1) predictions for Figure 5.1.

Class	$P(c)$
Mouse	0.99
Mousetrap	0.00
Toucan	0.00
Joystick	0.00
Computer keyboard	0.00

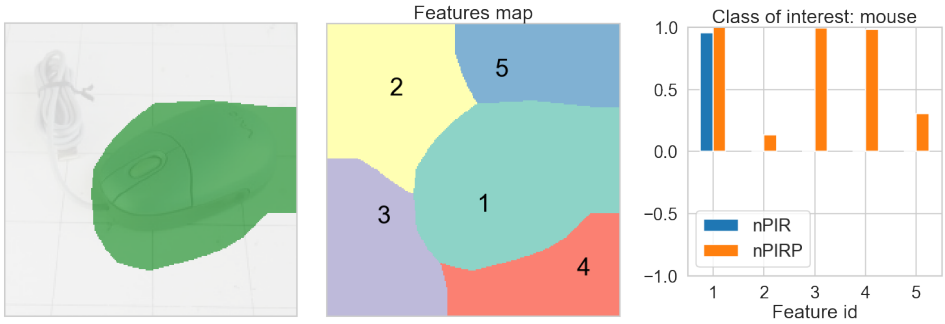
Table 5.2: InceptionResNetV2 (M4) predictions for Figure 5.1.



(a) Explanation of prediction *Toilette seat* (VGG16 in Table 5.1).



(b) Explanation of prediction *Mouse* (VGG16 in Table 5.1).



(c) Explanation of prediction *Mouse* (InceptionResNetV2 in Table 5.2).

Figure 5.2: Prediction local explanations. The input image is shown in Figure 5.1. Visual explanation (left), features (center), numerical explanation (right).

Answering Q4. Let us consider input I2, Figure 5.3 analyzed with model M2, Table 5.3. Apparently, M2 took the correct decision, but with some uncertainty. Indeed, the predictions for input I2 are class *Pizza* with $P = 0.48$ and class *Bagel* with $P = 0.16$. Figure 5.4a reports the explanation that answers to Q4, i.e., "Why is I2 a *Pizza*?". The visual explanation in Figure 5.4a-left shows the reasons for the model uncertainty. Through the explanation, we discover that most of the interpretable features are positively influencing the prediction of *Pizza*. The toppings of the pizza, i.e., feature 2, are the most positively influential, but also the underneath table, i.e., feature 1, is taken into great

Figure 5.3: *Pizza* input image (I2)

Class	$P(c)$
Pizza	0.48
Bagel	0.16
Corn	0.05
Pretzel	0.05
Meat loaf	0.04

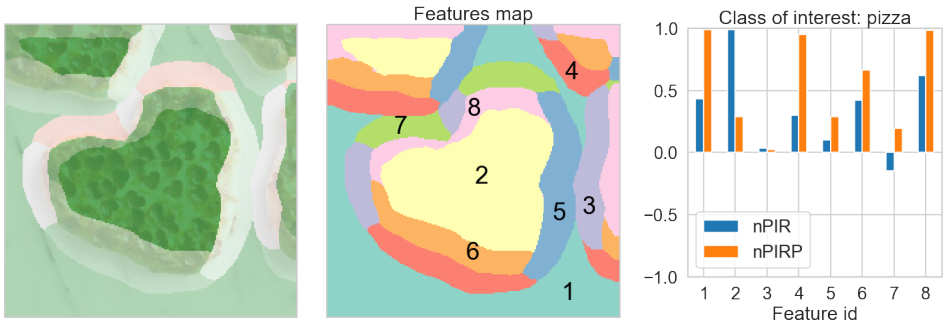
Table 5.3: VGG19 (M2) predictions for Figure 5.3

Class	$P(c)$
Pizza	0.76
Honeycomb	0.24
Dutch oven	0.00
Custard apple	0.00
Starfish	0.00

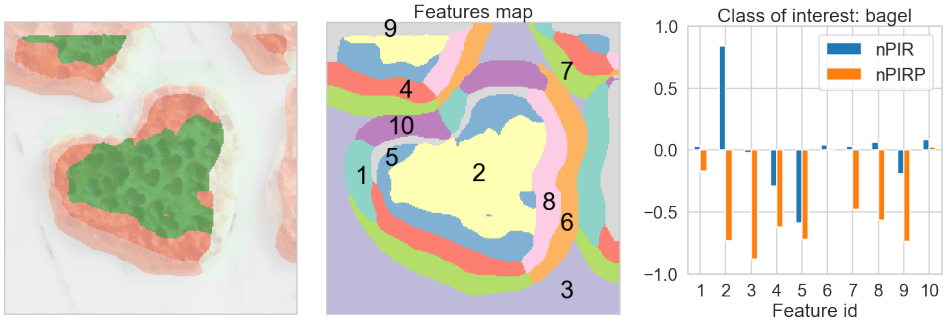
Table 5.4: InceptionV3 (M3) predictions for Figure 5.3

consideration by the model. The numerical explanation details that feature 2 has a very positive PIR but a low value of PIRP. This means that the pizza itself has a low precision for the class-of-interest *Pizza*. Instead, feature 1 has a $\boldsymbol{PIR} < 0.5$ and it is very precise with $\boldsymbol{PIRP} \approx 1$. So, in this case, the table is more important for class *Pizza* than the pizza itself.

Furthermore, we can answer the opposite question as well, i.e., "Why is I2 NOT a *Pizza*?". Figure 5.4b explain the reasons why class *Bagel* is the second most probable class for I2. The toppings on the pizza, i.e., feature 2 in Figure 5.4b, are confusing the prediction process. Indeed, the feature is positively influential for class *Bagel* with $\boldsymbol{nPIR} \approx 0.9$, but it is also negatively precise with $\boldsymbol{nPIRP} \approx -0.75$. This means that the inner part of the pizza is more influential for other classes than for *Bagel*. So, model M2 gets probably confused by the uncommon shape and texture of the subject during the prediction of class *Pizza* that, in this case, are recognized to be correlated also to the concept of *Bagel*. However, the class *Pizza* is confirmed to be influenced mainly by the texture of the underneath table. This strongly calls into question the reliability of the predictions. Without our detailed explanation process, these behaviors would remain hidden.



(a) Explanation of prediction *Pizza* (VGG19 Table 5.3). The number of interpretable features suggested by the engine equal to 8

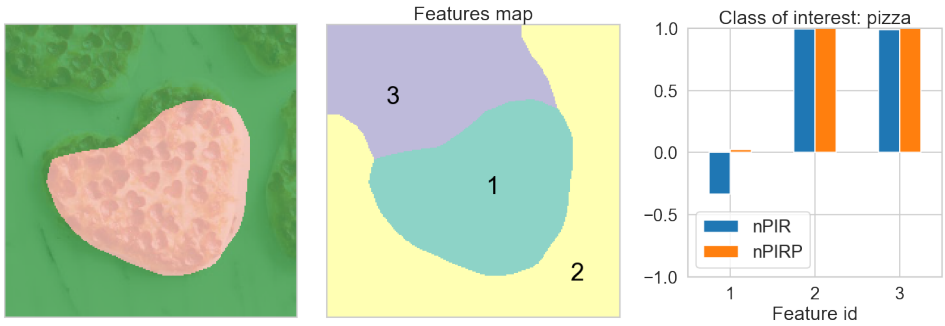


(b) Explanation of prediction *Bagel* (VGG19 Table 5.3). The number of interpretable features suggested by the engine equal to 10

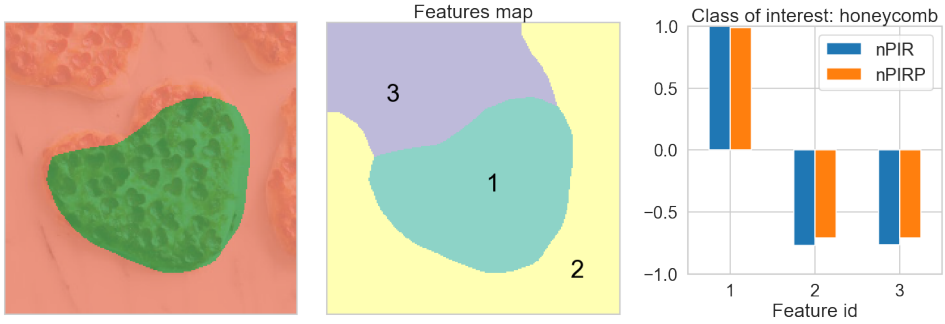
Figure 5.4: EBAnO Local explanations. The input image is shown in Figure 5.3. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right).

Answering Q5. Let us consider now the predictions of model M3 for input I2. The model's predictions are reported in Table 5.4. In this case, model M3 predicts the correct class *Pizza* with high probability $P = 0.76$. While the second relevant predicted class is *Honeycomb* with $P = 0.24$.

We want to assess the predictions of model M3 by exploiting our explanation process answering question Q5, i.e. "Why is I2 a *Pizza*?". Figure 5.4c reports the explanation. The explanation highlighted 3 informative features showing that only the context of the image is positively influencing the prediction *Pizza*. Indeed, features 2 and 3, in Figure 5.4c-center, are highlighted in green and they clearly represent the table, completely excluding the main subject. Moreover, the pizza itself, i.e. feature 1, is colored in red describing a slightly negative influence on the class *Pizza*. This behavior is even more clear by analyzing the quantitative results in Figure 5.4c-right. Indeed, PIR and PIRP values confirm that this prediction is completely biased by the texture contained in the context more than the subject being not only influential but also precise. So, thanks to the explanation we can say that the prediction process is completely biased since the concepts influencing the model outcomes are misleading and do not correspond to the



(c) Explanation of prediction *Pizza* (InceptionV3 Table 5.4). Number of interpretable features suggested by the engine equal to 3



(d) Explanation of prediction *Honeycomb* (InceptionV3 Table 5.4). Number of interpretable features suggested by the engine equal to 3

Figure 5.4: (Continue) Prediction-local explanations. The input image is shown in Figure 5.3. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right).

predicted class.

It is now important to further understand the reasons for the erroneous decision-making process by answering also in this case to the opposite question, i.e. "Why is I2 NOT a *Pizza*?". To do so we analyze the prediction process of the second most probable class, i.e., *Honeycomb*. The explanation for the prediction *Honeycomb* with model M3 is reported in Figure 5.4d. Interestingly, we note that the class *Honeycomb* is predicted considering mainly the pizza. Instead, in this case, the context of the input is negatively influencing the prediction. This behavior is highlighted in the visual explanation in Figure 5.4d-left and supported by the quantitative indexes in Figure5.4d-right. Even in this case the uncommon structure and texture of the pizza are leading prediction to a wrong result. It is true that the most probable predicted class is the correct one, i.e. *Pizza*, however, we demonstrated through our local explanations that the results of the model are based on completely misleading concepts.

5.3 Evaluating Per-Class Model Explanations

In Section 4.4 we showed how prediction-local explanations can be aggregated to describe the behavior of the predictive model at the global level. The local explanations are aggregated and analyzed by class-of-interest to show the most influential input patterns in terms of nPIR and nPIRP. Figure 5.5 shows the per-class model explanation reports computed on a pool of 50 images belonging to class *Dalmatian* with models M1, M2, M3, and M4.

As a first analysis, interpretable features for M1 and M2 are sparsely distributed in the whole $nPIR \times nPIRP$ space in Figures 5.5a and 5.5b. Contrariwise, models M3 and M4 show more dense patterns in Figures 5.5c) and 5.5d). Note that M1 and M2 belong to the VGG family of DCNN, thus they have a common architecture and we expect common behaviors.

Let us start with the results obtained for model M1. The main features correctly representative for class *Dalmatian* are distributed in the $nPIR \geq 0$ area. Also, they are characterized by $-1 \leq nPIRP \leq 1$. This means that the models learn correctly the features that are representative of the class being very influential. However, the precision of the learned concept varies a lot. They are mostly distributed in the first quadrant with $0 \leq nPIRP \leq 1$, i.e. between the neutral and precise edges. However, few samples are more influential for other classes than the one in the exam. Instead, the other features, representing contextual concepts, are distributed in the $nPIR < 0$ area, despite one outlier showing clearly a dalmatian dog positioned on the left bottom of the area. Then, to have a detailed explanation about the reasons why a specific feature is in a specific region the user can look at the corresponding local explanation. For instance, we note the presence of an outlier in the bottom left corner of the model explanation, showing a feature representative for a dalmatian dog which is both negatively influential and negatively precise. Also, comparing global explanations in Figures 5.5a and 5.5b, we can confirm that M1 and M2 behaviors are similar being characterized by a common distribution of *interpretable features*. So as a first result, according to the *optimal distribution of features* introduced in 4.4, we note that the class-based explanations of models M1 and M2 highlight a significant uncertainty in the prediction process of class *Dalmatian*.

The situation is very different for models M3 and M4 instead. Interpretable features extracted from M3 and M4 are concentrating along the edges of $0 \leq nPIR \leq 1$ and $nPIRP \approx 1$. We notice that features related to the background are mostly distributed in the $0 \leq nPIRP \leq 1, nPIR \approx 0$ area. This means that the features used to predict class-of-interest *Dalmatian* are characterized by a very high precision even if the influence itself may below for some samples. So, the explanations highlight a much more reliable decision-making process carried out by M3 and M4 for the class-of-interest. Indeed, these models associate a much more defined role to each feature. However, as in the previous case, we note the presence of an outlier feature on the left bottom corner of the plot that shows a dalmatian dog that is again negatively influential and not precise. The user can further inspect the details of the prediction process for the corresponding

input sample by looking at its prediction-local explanation. Furthermore, models M3 and M4 show similar decision-making patterns.

Comparing the two behaviors, the user is able to decide which model to trust the most. For instance, we can verify that model M1 is indeed more uncertain about the role of each feature while model M4 has a very clear differentiation of the features that are influential and the ones that are neutral. So, we can say that in this case models M3 and M4 are showing a more reliable decision-making process w.r.t. M1 and M2. This result is also coherent with the state-of-the-art knowledge: M4, i.e. InceptionResNetV2 is known in the literature as a much more accurate and reliable model w.r.t. M1, i.e. VGG16.

Finally we highlight also that our explanation process is able to highlight the model uncertainty following a completely unsupervised approach. Indeed, the proposed framework does not require the ground truth labels to compute any of the explanations. Hence, class-based model explanations are widely applicable, and they can improve the understandability of the black-box decision-making process.

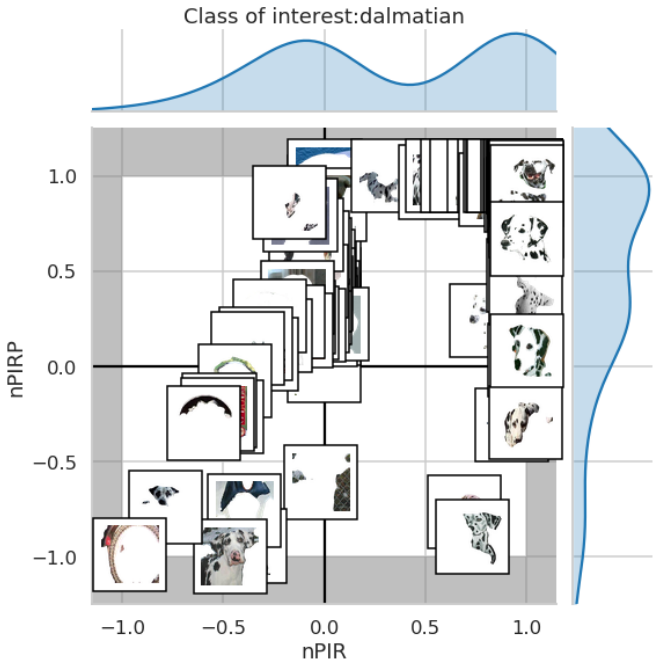
5.4 Comparisons with State-Of-The-Art Explanations

In this section, we compare our local explanation process with the ones available in the state-of-the-art, i.e.m *LIME* [69] and *GRAD-CAM* [76]. We provide a comparison with two input samples, i.e. I3 and I4 in Figures 5.6 and 5.8 respectively.

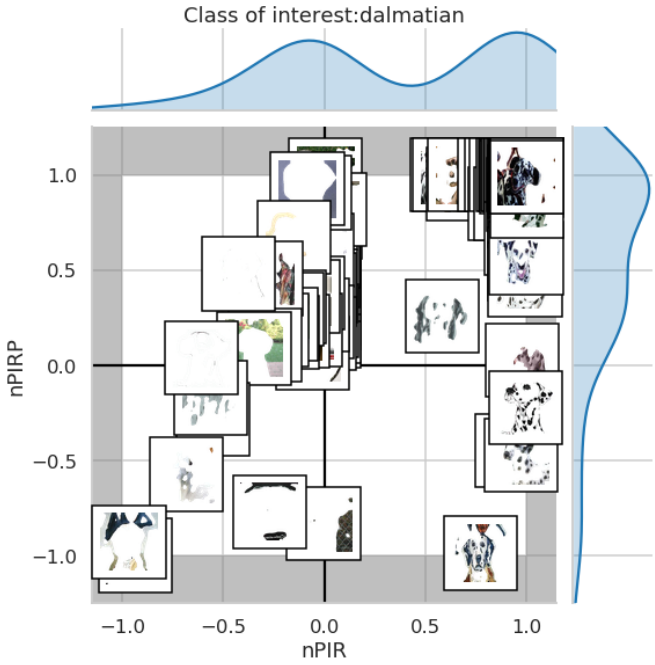
The comparison takes into account the input in Figure 5.6 for which M1 predicts *Acoustic Guitar* with $\boldsymbol{P} = 0.22$ in Table 5.5). Image I3 has been extracted from [69] as a comparison. The explanation proposed by *LIME* is reported in Figure 5.7a. While the *GRAD-CAM* explanation is reported in Figure 5.7b. *LIME* highlights in green the areas that are relevant for the class-of-interest, while in red those negatively influencing the prediction. On the other hand, *GRAD-CAM* exploits a color gradient going from hot to cold colors, i.e. from red to blue, to highlight important and not important input areas.

The explanation provided by *LIME* is quite confusing because of the feature extraction process based on a model agnostic segmentation strategy. For instance, the presence of many small sparse green areas makes it difficult to interpret the real model behavior. Also, it may be possible that the network is confused by the presence of some background noise but, using *LIME* we can not be sure of this since this method is not taking into account the model knowledge.

GRAD-CAM, on the other hand, is more precise. It identifies with greater precision the area around the neck of the guitar that is influencing the prediction process and it ignores the background portions that were identified by *LIME* as influential. Nevertheless, *GRAD-CAM* loses the information about the positive and negative influence of the highlighted input area. Indeed, *GRAD-CAM* creates the saliency map by analyzing the information extracted directly by the gradient of the last convolutional layer of the network. Also, it is not able to highlight the information related to multiple regions of the

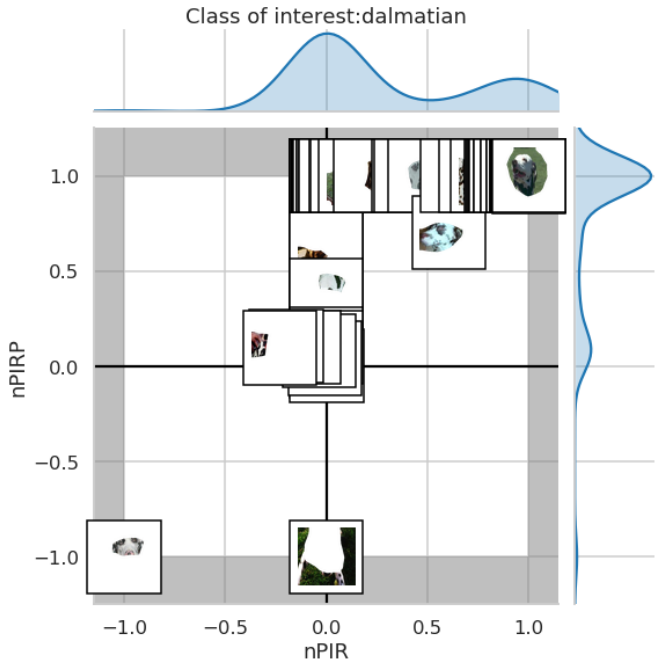


(a) Model M1 (VGG16).

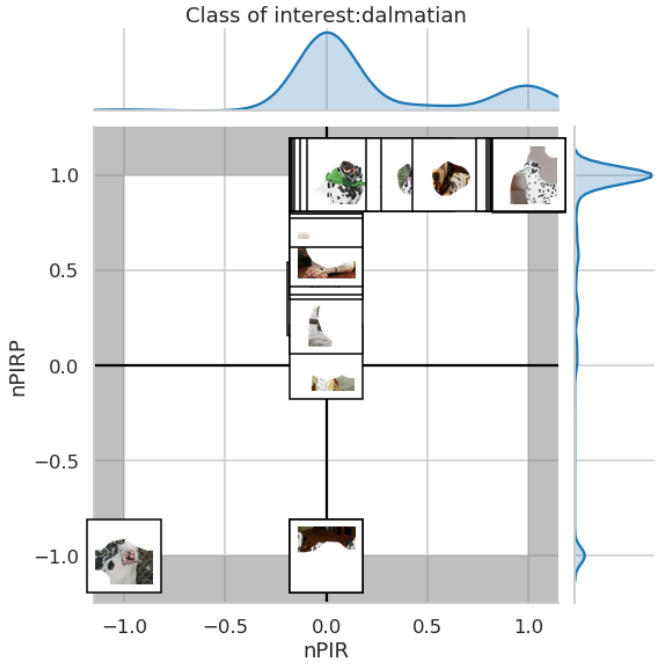


(b) Model M2 (VGG19).

Figure 5.5: Class-based model explanation with class-of-interest *Dalmatian*.



(c) Model M3 (InceptionV3).



(d) Model M4. (InceptionResNetV2)

Figure 5.5: (Continue) Class-based model explanation with class-of-interest *Dalmatian*.

Figure 5.6: Input image (I3).

Class	$P(c)$
Acoustic guitar	0.22
Electric guitar	0.09
Golden retriever	0.06
Stage	0.04
Sussex spaniel	0.03

Table 5.5: VGG16 (M1) predictions for Figure 5.6

image separately. It can just discriminate between important and not important input areas.

Also, as already introduced, none of the two state-of-the-art methods propose a human-readable numerical explanation of the prediction. Instead by analyzing our explanation for input I3 in Figure 5.7c we can highlight the following advantages:

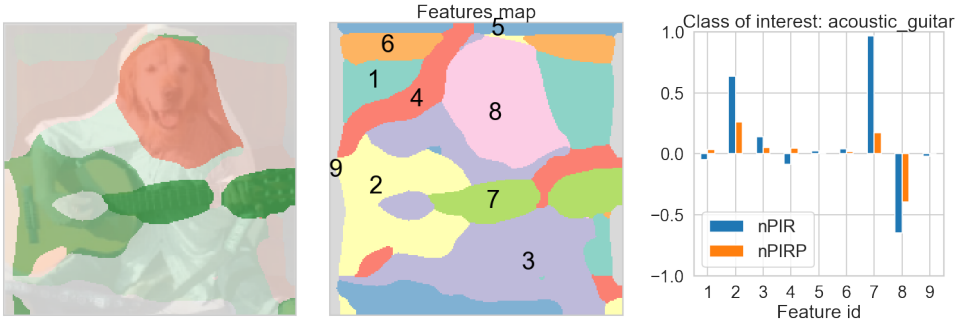
- The feature map in Figure 5.7c-center accurately identifies concept-wise portions of input responsible for the model's outcome.
- The visual explanation in Figure 5.7c-left highlights both the positive and negative influencing features for each class-of-interest.
- The quantitative explanation in Figure 5.7c-right details the influence of each feature and measures the precision of each image portion.

Indeed, our explanation identifies the input portions that correspond to the guitar as very influential with a high value of nPIR values and positive nPIRP values. At the same time, the face of the dog has been correctly identified as negatively impacting *Acoustic Guitar*. We also provide here the explanation for class *Golden Retriever* in Figure 5.7d. It shows that the guitar has in this case a very negative impact and the dog face is positively influencing the decision-making process.

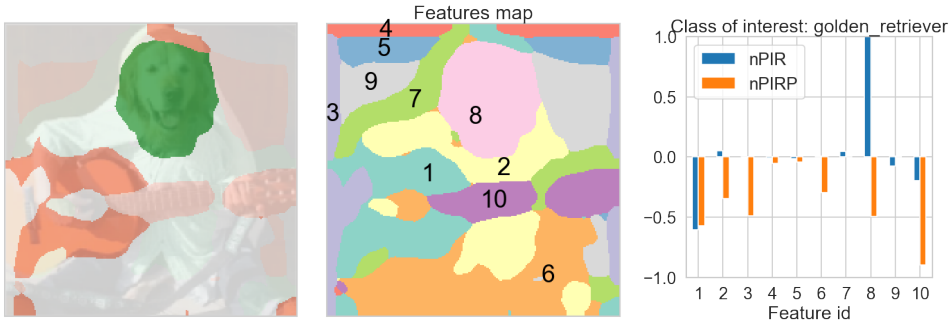
We exploit input I4 in Figure 5.8, which is extracted from [76], as a further comparison. For input I4, model M4 predicts *Bull Mastiff* with $P = 0.63$ in Table 5.6. Let us consider the class-of-interest *Bull Mastiff*. *LIME* highlights in Figure 5.9a the top area of the image, including much of the background, as positive for the prediction. *GRAD-CAM*, instead is much more specific, highlighting only the head of the dog. Even in this case, both methodologies show the same issues highlighted in the previous example: *LIME* is not precise in identifying influential input portions, while *GRAD-CAM* does not provide any evidence of evaluation of areas different than the influential one.

(a) Explanation of the prediction *Acoustic guitar* with *LIME* (in green).

(b) Explanation of prediction *Acoustic guitar* with *GRAD-CAM*.



(c) Explanation of prediction *Acoustic guitar* with EBAnO.



(d) Explanation of prediction *Golden retriever* with EBAnO.

Figure 5.7: Prediction-local explanations. The input image is shown in Figure 5.6. Visual explanation (left), features (center), numerical explanation (right).

Our explanation in Figure 5.9c instead clearly shows which is the contribution of each input region.

We show that the head of the dog is very positively influential and precise with both nPIR and nPIRP equal to 1. Also, we show that the other features are negatively impacting this prediction. For instance, the cat, feature 2, is the most negative feature in

Figure 5.8: Input image (I4) taken from [76] for comparison.

Class	$P(c)$
Bull mastiff	0.63
Tiger cat	0.11
Tiger	0.04
Tabby	0.02
Boxer	0.01

Table 5.6: InceptionResNetV2 (M4) predictions for Figure 5.8

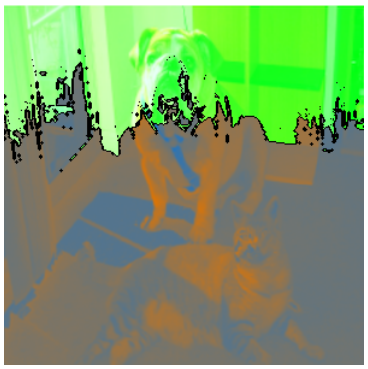
the input. We report also the explanation of the class-of-interest *Tiger Cat* in Figure 5.9d. It confirms that the feature containing the cat, i.e., feature 5 is positively influencing the prediction process with $nPIR \approx 1$. Also, its nPIRP is very low and it can explain why the prediction for class *Tiger Cat* shows a low $P = 0.11$ for model M4.

5.5 Assessment of Influence Measurement

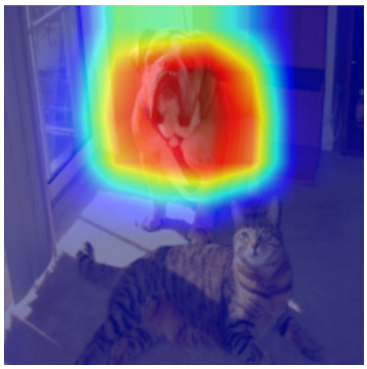
We analyze here how the proposed nPIR index is distributing over the whole set of 250 images exploited to test the framework. This analysis has the purpose of showing the general behavior of the index when exploited with different models and to check if it correlated to the meta-characteristic of the analyzed features, e.g. feature size in pixels.

We show the distributions of the minimum and maximum nPIR values in Figure 5.10 for the four studied models. The distributions take into account only the most informative explanations computed with class-of-interest equal to the top-1 prediction. The large shift between the minimum and the maximum nPIR distributions shows that the index is actually able to discriminate between influential and not influential features. This is positive since we want our explanations to contain as much contrast as possible among feature influence. Also, despite the limited number of examples commented on in this experimental section, we can guarantee that our methodology is able to provide informative explanations independently by the model.

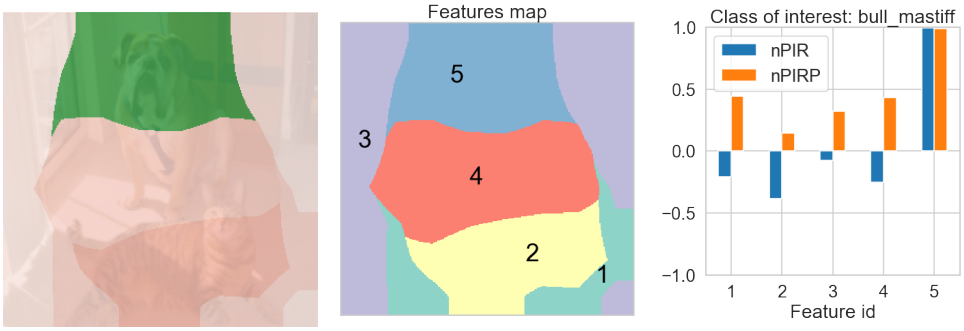
In detail, we note that the top influential features, with maximum nPIR, are mainly included in the $[0.8, 1.0]$ range. Only models M3 and M4 have some features of the top influential ones falling in the $[0.0, 0.2]$ range. Furthermore, the maximum nPIR distribution never goes below 0 for any model. So, we are always able to identify a feature positively influencing the prediction process independently by the model.



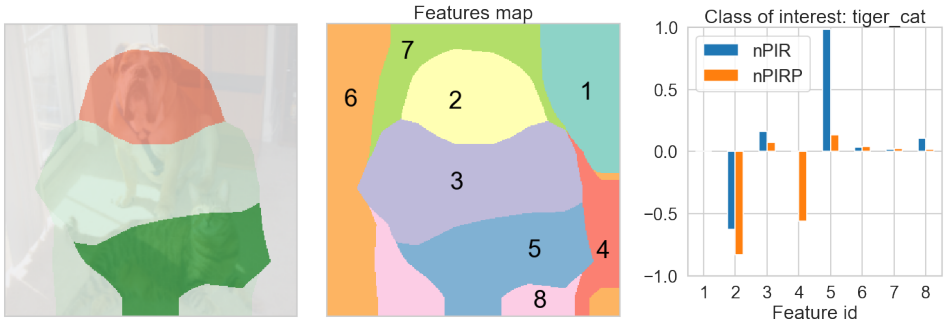
(a) Explanation of prediction *Bull mastiff* with LIME. Green segments are the explanation.



(b) Explanation of prediction *Bull mastiff* with GRAD-CAM.



(c) Explanation of prediction *Bull mastiff* with EBAnO. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right)



(d) Explanation of prediction *Tiger cat* with EBAnO. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right)

Figure 5.9: Prediction-local explanations. The input image is shown in Figure 5.8. Visual explanation (left), Interpretable features (center), nPIR and nPIRP (right)

On the other hand, the minimum nPIR distribution is predominantly ranging in $[-0.2, 0.2]$. Consequently, most of the less influential features can be considered almost neutral for the prediction process. Minimum values higher than 0.2 are very rare.

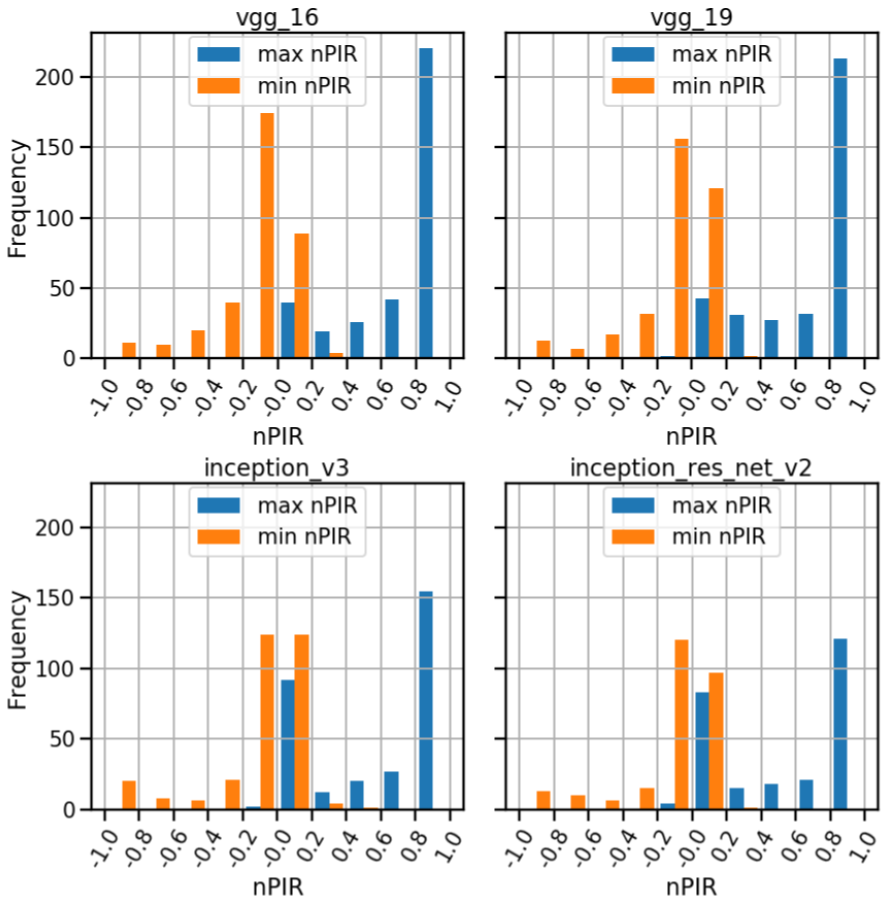


Figure 5.10: For each tested model the distributions of min and max nPIR values obtained by the prediction-local explanations with class-of-interest equal to the top-1 predicted class for each input image.

Table 5.7: Pearson correlation between nPIR, nPIRP, and feature sizes. The abs_feat_size is the absolute feature size concerning the number of pixels while the rel_feat_size is the relative feature size w.r.t. the size of the input image.

	Abs. Feat. Size	Rel. Feat Size
nPIR	0.17	0.19
nPIRP	0.13	0.07

This confirms the large distance between the minimum and the maximum nPIR distributions. This confirm also the validity of the strategy to choose the most informative explanations.

We want to understand also if the nPIR and nPIRP are influenced by the size of the extracted interpretable features. To this aim, we compute the correlation between the two indexes and the size of the extracted features, in terms of absolute size (number of

pixels) and relative size (number of pixels of the feature over image size). The results are shown in Table 5.7. It is important to check this correlation because we want to measure the influence of interpretable features independently of their sizes. The results show that the influence measured with nPIR and the precision measured with nPIRP are not significantly influenced by the size of the features. Their correlation with both absolute, i.e. in pixels, and relative sizes is always lower than 0.2.

5.6 Efficiency and Effectiveness of Influence Measurement

We compare in this section the performance of our nPIR against the state-of-the-art *Shapley values* [85, 48].

The *Shapley Values* [85, 48] are used to explain the relevance of the features exploited by the decision-making process. The concept of *Shapley Values* comes from *Game Theory* in statistics. The prediction process is considered a game and the input features are the players. The purpose of this index is to measure how much each player has contributed to an outcome of the predictive model. So, it can be computed by removing and analyzing the effects of every possible combination of input portions. Each combination of input features is a *Society* of features. Accordingly to the number of features extracted by the explanation process, this index can become very onerous in terms of computational power. The standard definition of *Shapley Values* requires evaluating the contribution of all the possible societies which results in a complexity close to $O(|F|!)$. Moreover, *Shapely Values* does not consider the precision of the influence for each feature. Also, works that implement this index, e.g. SHAP [47], as quantitative explanation are using in practice approximated versions because of its high complexity.

Instead, our nPIR index is computed by analyzing only the features extracted from the knowledge of the model. So, we can obtain very informative results by just considering the interpretable features extracted. In our case, the complexity of analyzing the effects of the features is linear $O(|F|)$. Then, to compare then the effectiveness of the two indexes we use the standard definition of *Shapley Values* since it is the most accurate.

In Figure 5.11 we show the comparisons between nPIR and *Shapley values* for the input in Figure 5.1. We compare (i) model M1 predicting class *Toilet seat* in Figure 5.11a, (ii) model M1 predicting class *Mouse* in Figure 5.11b, and (iii) model M4 predicting class *Mouse* in Figure 5.11c. Figure 5.11a shows that nPIR better emphasizes the importance of each feature. The *Shapely values* show an almost flat trend. Only feature 1 has been identified as slightly negative. Instead, nPIR better emphasizes the contributions of each feature showing a larger contrast among features. Indeed, it puts in evidence the negative influence of feature 1 and it shows more clearly the positive influence of features 5 and 6. Also, in Figure 5.11b our nPIR is more effective than *Shapley Values* by better identifying which are the real impacting features for the prediction of class *Mouse*.

(a) Model M1 with class-of-interest *Toilet seat*.

(b) Model M1 with class-of-interest *Mouse*.

(c) Model M4 with class-of-interest *Mouse*.

Figure 5.11: Comparison of nPIR index and *Shapley values*.

Even if, feature 1 is considered by both as positive, nPIR better emphasize it. Then, nPIR correctly identifies feature 4 as negative, while *Shapely values* do not identify this contribution. Lastly, Figure 5.11c shows that both indicators have the same trend. Feature 1 is considered very positive by both of them, while all the other features are neutral.

The proposed nPIR index is more efficient and effective w.r.t. the state-of-the-art *Shapely Values* in identifying the contributions of each input feature. Indeed, in all the performed experiments, nPIR has been always more clear in the results showing equal or better results.

5.7 Human Validation

Human beings are the main beneficiary of explanation frameworks. So, to assess the quality and the interpretability of our explanations we take into account users feedbacks. To this aim, we developed a publicly-available online survey[5]. The survey allowed us to assess the effectiveness and the easiness of understanding of our prediction-local explanation process. The survey's goal is to validate (i) the understandability of our explanations and (ii) to validate the quality of our framework against the explanations proposed by state-of-the-art strategies, i.e., *LIME* [69], and *GRAD-CAM* [76].

The survey is composed of 12 predictions with their local explanations. The local explanations have been chosen to be appreciated by both expert and non-expert users. So, we proposed only images analyzed by all the four models considered in this chapter and for which models correctly predicted the class. In this way, we avoid misleading behaviors of the models or complex concepts that would require detailed analysis or expert skills.

[5] https://ebano-ecosystem.github.io/#ebano-survey

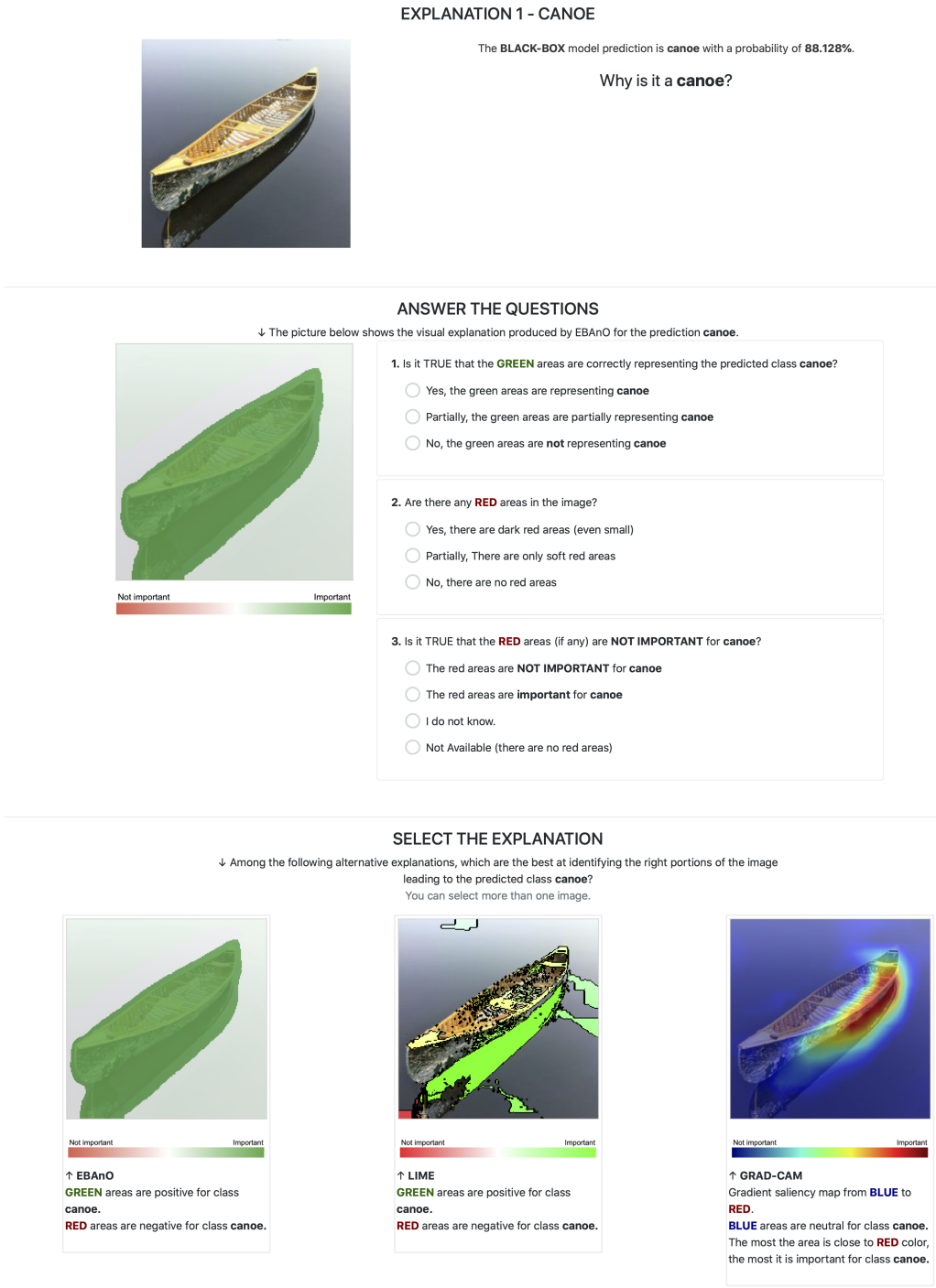


Figure 5.12: Survey example question.

A sample question, extracted from the survey is reported in Figure 5.12. Each question is divided into three sections:

1. On the top, we show the input image along with details about the model's prediction.

2. Then we show the prediction-local explanation and we ask a few sub-questions about the relevance/irrelevance of green and red areas.

3. Lastly, we propose to the user the visual explanations computed with our framework (i.e. EBAnO), *LIME* and *GRAD-CAM* and we ask to select the ones that better are representing the prediction of the target class-of-interest.

The user is asked to verify if he considers relevant the portions of the visual explanation in terms of positive (green) and negative (red) influence. Then, we asked to select, allowing multiple selections, which of the explanation strategies is representing the class-of-interest with higher precision. Since all the predictions are correct and we select the explanations more representative for them, this last question aims to understand if the average user would actually understand the explanation itself and which of the three methodologies is more effective from a human point of view.

The survey has been completed by 60 people to which we asked anonymous information about their background. The instruction level is distributed as follows: 25% bachelor, 38% master, 32% Ph.D., 5% other. Also, respondents are belonging to different age ranges: 18% between 19 and 24, 51% between 25 and 29, 30% with more than 30 years.

The green areas of our visual explanations have been selected 55% as *Important*, 40% as *Partially Important*, and only 5% have been considered *Not Important*. So, in 95% of the cases, the influential areas of the input have been considered correctly characterizing the class-of-interest. In turn, red areas have been correctly considered *Not Important* for the target class-of-interest in the 69% of the answers, when present.

Thus, also the areas negatively impacting on the prediction process, have been interpreted correctly by the majority of the users. Just 23% of the interviewed get confused about the role of the red portions of the explanations and only the 8% of the time the user was not able to answer specifying *I do not know*.

Then, analyzing the results from the third section of the survey it is possible to understand which of three tested explanation frameworks is more able to identify the portions of the input that are representing the predicted class-of-interest accordingly to human users. Figure 5.13 shows, for each of the 12 images, the percentage of times in which *EBAnO*, *GRAD-CAM*, and *LIME* have been selected.

Our explanation strategy has been chosen w.r.t. the other two methods in 67% of the cases. In particular, *EBAnO* has been considered more interpretable concerning *GRAD-CAM* in 75% of the cases. While more interpretable than *LIME* on 75% of the cases. Overall, considering all the 720 answers, i.e. 12 questions times 60 respondents, our framework has been chosen 439 times, while *GRAD-CAM* 242 times, and *LIME* 156 times.

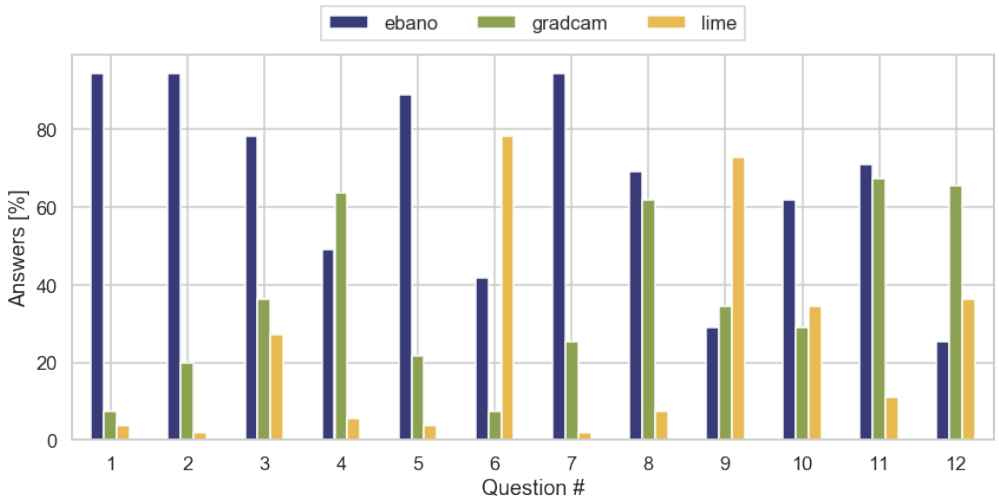


Figure 5.13: Survey results. For each of the 12 images, we report the percentage of times that EBAnO, *GRAD-CAM*, and *LIME* have been selected as the best explanation result for the class of interest (multiple choice allowed).

We collected also feedbacks after answering the survey. As result, we note that users, especially non-expert ones, exploit the quality of the segmentation as an implicit metric to evaluate the quality of the explanation. Unlike the other methods, our *interpretable features* correlate the image portions to the knowledge of the model, so the quality of the segments can be exploited as a soft metric to decide if rely on the model predictions or not. Indeed, in our case, a precise segmentation of the input is a sign that the model correctly learned the concepts related to the class-of-interest.

Chapter 6

Explaining Deep Natural Language Models

This chapter describes the details of our explanation process tailored to Deep Natural Language Models. As in the previous use-case, i.e. image classification, the general approach to the explanation process presented in Chapter 3 remains the same. However, to fully take advantage of the inner-model knowledge of Deep NLP models, the proposed framework includes new domain-specific tools such as the Multi-Layer Word Embedding (MLWE) feature extraction. The study proposed in [92] supports this choice. Indeed, textual embeddings have been recognized to have interesting interpretable properties. Also, [25] states that modern natural-language models incorporate most of the context-specific information in their inner layers.

Our explanation framework produces both prediction-local and model-global explanations, which, in the case of NLP tasks, consist of textual and numerical human-readable reports.
In particular, this chapter describes:

- The prediction-local and model-global explanation process is tailored to NLP.
- The interpretable feature extraction process is tailored to textual input documents which include a set of model-wise strategies to mine the inner-model knowledge.
- The definition of quantitative and qualitative explanations, measuring the influence of each set of features on the local outcome provided by the black-box model.
- The innovative model-global explanation process tailored to the NLP domain by introducing two new domain-specific indexes, the *Global Absolute Influence*, and the *Global Relative Influence* scores.

The following Chapter is organized as follows. Section 6.1 introduces an overview on the explanation process when applied to NLP tasks. Section 6.2 describes how the interpretable are extracted in the context of NLP. Then, Section 6.3 provides the details

of the new Multi-Layer Word Embedding feature extraction technique. Section 6.4 introduces the perturbation process exploited in this context. Section 6.5 explains how the prediction local explanations are produced in this case. Finally, Section 6.6 provides the details of the per-class model-global explanation process tailored to textual input data.

6.1 Deep NLP Explanation Process Overview

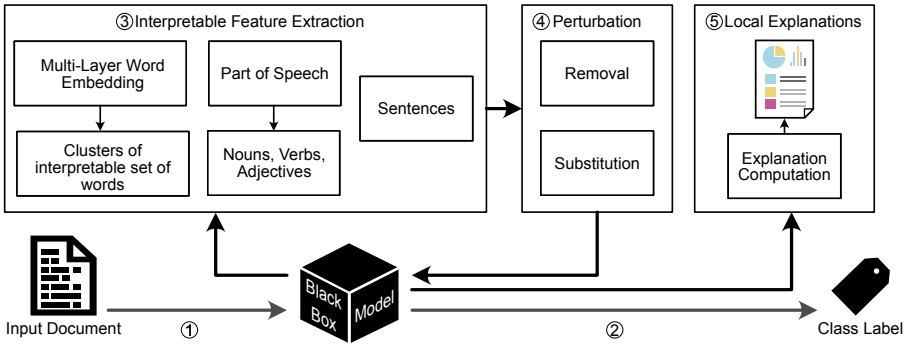


Figure 6.1: Local explanation process.

Figure 6.1 shows an overview of the explanation process tailored to Deep NLP models.

Let's consider a deep neural network, which is trained to solve a classification task, and an input document given input to the model (Step ①). The black-box model predicts the class label along with the probability distribution for each class label (Step ②).

Then, the explanation process for a class-of-interest begins. First, the *interpretable features* are extracted (Step ③). This step extracts both *model-agnostic* (i.e., part of speech, sentences) by exploiting common NLP techniques and *model-aware* (i.e., multi-layer word embeddings) features.

The interpretable features are iteratively perturbed and the outcomes of the model are evaluated (Step ④) measuring the impact of each perturbation w.r.t. the original predictions. As already introduced in previous chapters, also in this case the perturbation of a specific feature can affect the outcomes of the model by: (i) increasing, (ii) decreasing, or (iii) keeping unchanged the probability of the class-of-interest.

Accordingly to this, the framework combines the effects of all the analyzed features through the nPIR and nPIRP indexes (introduced in Chapter 3). Thus, the framework computes the explanation producing the *local explanation* report (Step ⑤) and the results are showed through an informative dashboard.

Finally, this explanation process can produce model-global explanations by combining a pool of local explanations computed to a number of input documents. To do so, we introduced two new indexes, namely *Global Absolute Influence* (GAI) and *Global*

Relative Influence (GRI), that analyze relevant inter- and intra- class semantic concepts that are influencing the black-box decision-making process. Section 6.6 provides the details.

6.2 Interpretable Feature Extraction

The interpretable feature extraction process is one of the most critical components to provide meaningful explanations in the context of NLP tasks. The challenge is to identify portions of the input text that are meaningful for both the model under exam and the end-user.

A feature is considered meaningful for the model if it has a relevant role in the prediction process. Even singular words may be considered meaningful if their perturbation is causing huge changes in the predicted outcomes. Instead, a feature is meaningful for the end-user if it can simply understand its semantical meaning.

To find the correct set of words to perturb however requires an optimized approach. Indeed, identifying a set of meaningful words by using a brute force approach may requires huge computational resources and the quality of the results is still not guaranteed. On the other side, the understanding of the end-user is correlated to the semantical meaning of the analyzed features.

Finding a trade-off between these two requirements is necessary. We developed the interpretable feature extraction process by integrating both NLP techniques, which are model-agnostic and a model-aware strategy called Multi-Layer Word Embedding. Thanks to the combination of multiple techniques, we are able to explore different aspects of the prediction process. The input text is first split in tokens, i.e. singular words. Then, the features are elaborated. Specifically, we adopted three different *interpretable feature extraction* strategies:

- *Part-of-Speech* (PoS). This strategy analyze the influence of singular part-of-speech, i.e. adjectives, nouns, verbs, adverbs, on the prediction process. Also, we record the position where each PoS token is located in the text, since different positions in the text may have different impacts. Each set of tokens belonging to a specific Pos are considered a singular interpretable feature. This representation is also very clear for a human-user since he can early evaluate the role of each word in a text knowing its role (e.g., adjective or verb).

- *Sentence-based*. Each sentence in the text is analyzed separately measuring the influence of each of them. Indeed, modern NLP models consider the whole context in which words are used. Thus, this is a necessary interpretable feature to consider while explaining the prediction process. Also, this is another technique that is clearly understandable by the user: the mining of whole sentences can be easily understood by humans.

- *Multi-layer Word Embedding* (MLWE). This is the most powerful strategy. It allows to extract knowledge about the prediction process by mining directly the output of the inner-layers of a network during the prediction process. To apply this technique it is necessary to access the predictive model. However, it can ease the extraction of meaningful patterns of correlated words. The extracted patterns of words would be the ones most representative of the model itself. MLWE feature extraction is detailed in Section 6.3.

The first two techniques are completely model-agnostic and allow to analyze sets of features that are very meaningful also for the end-user (e.g. adjectives have a specific role in the logical grammatical structure, sentences usually contain complete concepts). The way in which the features are extracted, through these techniques, allows to intrinsically maintain their contextual information (e.g. adjectives have a specific role in the logical grammatical structure, sentences usually contain complete concepts) and this information can be used by the end-user to assign a significance to the extracted features. Instead, the MLWE features are very meaningful for the model itself since they are extracted directly from the parameters of the network under exam. However, they may lack interpretability. For example, few words may be correlated for the model but not for their semantical meaning and they can be very influential for the prediction process.

In case a prominent influential feature is not identified, separately for each extraction method, we test pairwise combinations of features to create larger groups of tokens corresponding to more complex concepts. For example, if after the analysis of the features extracted by PoS, any impacting feature is found, we combine, for instance, *Adjectives* with *Verbs* or with *Nouns* and so on. The same strategy is applied also for all the other methods by combining, for instance, pairs of sentences or pairs of clusters of words (exploiting the MLWE).

6.3 Multi-layer Word Embedding (MLWE)

When dealing with such models, the objective of the training phase is to approximate the knowledge contained in the training data into a numerical model that can be applied to perform predictions for new incoming data. Specifically, the hidden layers of a deep neural network have the role of automatically extracting the most relevant features of a given input. The outcomes of the internal layers of a neural network can be often exploited as an embedded numerical representation of the input words in a document.

We want to extend this concept by including the knowledge of even multiple layers in the explanation process. Indeed, the more the explanation is related to the knowledge of the model the more it would be reliable. To do so, we introduce the Multi-layer Word Embedding (MLWE) feature extraction.

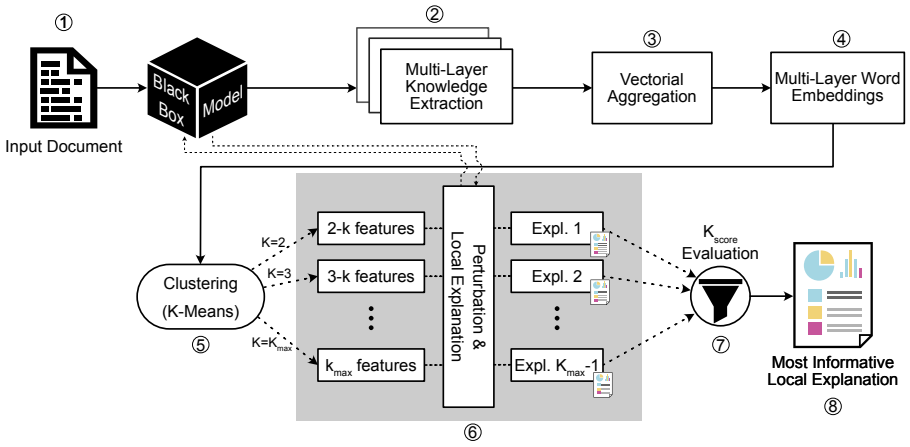


Figure 6.2: MLWE feature extraction process.

Figure 6.2 provides the main steps of this feature extraction process. The first step is the extraction of the output from the model's hidden layers for a given input (Step ①) which produces a tensor of layers' output (Step ②). Then, the tensor containing the output of each layer is aggregated (Step ③). This operation reduces the dimensionality in a similar way in which a pooling layer would do (e.g. exploiting average or sum). Additionally, the current representation is further reduced through Principal Component Analysis (Step ③)). The process up to here produces the *Multi-layer Word Embedding* of the input document (Step ④).

This embedding process allows highlighting correlated words from the model point of view, i.e. with a similar MLWE. Thus, it is possible to mine the MLWE knowledge to identify the key input concepts, that most probably are influencing the current prediction. The MLWE feature extraction, and in particular the extraction of the aggregated word embeddings from multiple layers, has to be achieved in different ways depending on the neural network architecture under the exam.

6.3.1 MLWE Extraction in Practice.

We report here two practical examples of MLWE feature extraction on two entirely different architectures: (i) a Long Short-Term Memory (LSTM) network with a single static embedding layer derived from GloVe [63], and (ii) a BERT model which is based on transformers and presents a contextualized word embedding on multiple layers. These two models will be exploited also in Chapter 7.

Long Short-Term Memory. LSTM networks are robust architectures that can learn from both from time dimension and context of words. The LSTM taken into account for this practical example includes an embedding layer of 300 dimensions, two bidirectional LSTM layers (with 256 units for each direction), and a dense layer with 128 hidden units.

This LSTM model exploits one embedding layer that works with full tokens (i.e. the input words are not split in smaller tokens) and two bidirectional LSTM layers. This is the simplest use-case, where there is only one layer that can be exploited for the MLWE extraction. So, if an input text is composed of t tokens (i.e. words) the MLWE extracts a tensor of shape $(t \times 300 \times 1)$. Then, a Principal Component Analysis is used to reduce the embedding matrix shape to $(t \times c)$, where c is the number of required principal components. This last matrix is *multi-layer word embedding* representation for the LSTM model.

BERT. A much more complex process has to be carried out in the case of a BERT model. Let us consider the BERT base (uncased) model [23]. This model is composed of 12 transformer layers [94].

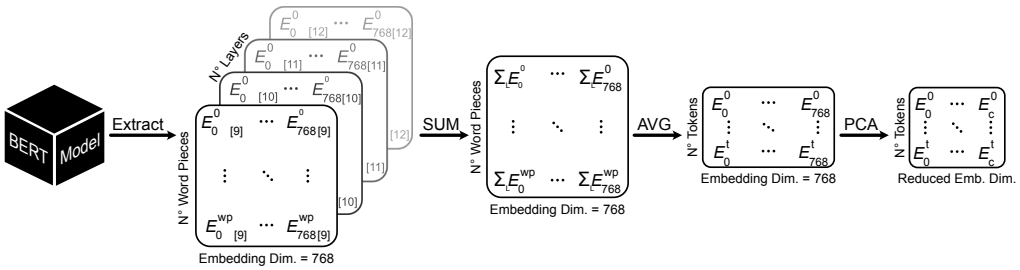


Figure 6.3: BERT multi-layer word embedding feature extraction process. With: $E_{c\,[L_{id}]}^{<wp \text{ or } t>}$ such that: E is the word embedding matrix, wp and t indicate the position of the word-piece and token respectively in the input text, c is the component of the word embedding vector and L_{id} is the layer from which is extracted.

Figure 6.3 shows the MLWE feature extraction process exploited in this case. The BERT model requires a pre-processing phase to split each input word in smaller tokens, called word pieces wp. So, each transformer layer produces a tensor of shape (wp×768). We want to exploit the information contained in as much layer as possible. However, it has been motivated in literature [25] that modern MLP models incorporate most of the context-specific information in the last and deepest layers. So, we consider the extraction of the last four transformer layers.

The MLWE feature extraction process begins by collecting the output of the last four transformer layers, i.e., L_9, L_{10}, L_{11}, L_{12} (Figure 6.3-left). The output is a tensor of shape $wp \times 768 \times 4$ where each row represents the embedding of a word piece in each layer. Then, accordingly to [25] these embeddings can be aggregated by summing the values over the layer axes (Figure 6.3-center-left). This results in a matrix of shape $wp \times 768$. Since the smallest part of the input that we consider are full words, but BERT works with word pieces, their embedding has to be aggregated as well. So, the embeddings of word pieces belonging to the same word are averaged over the word-piece axes (Figure 6.3-center-right) The result is a new matrix of shape $t \times 768$, where t

corresponds to the number of input words. Finally, the Principal Component Analysis is exploited to reduce the sparsity of the previous representation (Figure 6.3-right). The final MLWE representation for the BERT model is a matrix of shape $(t \times c)$, where each input token is represented through a vector of c dimensions.

6.3.2 Exploiting MLWE in the Explanation Process

Discovering groups of input words by analyzing the contributions of singular tokens to the prediction process is often not sufficient to produce exhaustive explanations. Because of the high dimensionality characterizing NLP tasks, models tend to be affected by many features at the same time. However, testing all the possible combinations of input tokens to identify the most impacting ones is often unfeasible, leading to huge search space. Multi-Layer Word Embeddings can help to overcome this issue. The main challenge is to find the smallest group of words that impacts the prediction, and that will produce the most informative explanation possible.

Thanks to their properties, the MLWE word representations can be unsupervisely mined to identify groups of correlated meaningful words for the target model. Thus a cluster analysis based on the well-known K-Means [46] algorithm is exploited (Step ⑤). A critical parameter in clustering analysis the desired number of groups K to extract. In our case, this corresponds to the number of concepts to analyze when producing the explanation. A fine partitioning, with large K, would fine very specific concepts that however can be too specific for the explanation purpose. On the other hand, gross partitions, with small K, can aggregate everything in few broad concepts that would make the understanding of the corresponding explanation more complex. For these reasons, we explore a number of partitions ranging in $\{2, \dots, K_{max}\}$, where the max number of clusters K_{max} is a function of the number of words N_{words} in the input. It has been empirically set to:

$$K_{max} = \sqrt{N_{words} + 1} \tag{6.1}$$

This definition prevents the number of clusters to be as large as the number of words in the input. Indeed, it would not be useful nor feasible to consider partitioning with K as large as N_{words}. The root of N_{words} provides a good trade-off between input size, performance required to perform the clustering analysis, and quality of the output explanations.

To identify the best partitioning, all the corresponding prediction-local explanations are computed (Step ⑥). Each partitioning is composed of K interpretable features, i.e., groups of correlated words. Thus, each $K \in \{2, \dots, K_{max}\}$ is evaluated by iteratively perturbing the content of the interpretable features and the explanations are computed. Next Sections 6.4 and 6.5 will provide more details about these steps. Once all the local explanations are available, we analyze how much information is contained in each of them. Potentially. a large number of local explanations is produced in the previous step. However, only the most informative explanation has to be provided to the end-user. We

define the *most informative local explanation* as the one with the highest impact on the prediction process. For each partitioning $K \in \{2, \dots, K_{max}\}$, the informativeness score K_{score} is computed (Step ⑦) We recall that the influence of an interpretable feature over the prediction process is measured with the *normalized Perturbation Influence Relation* (nPIR) index (Chapter 3). Similarly to what we showed in Section 4.2 we define the informativeness of an explanation as:

$$K_{score} = \max_{\kappa \in K} \left(\frac{nPIR_{\kappa}}{|\kappa|} \right) - \min_{\kappa \in K} \left(\frac{nPIR_{\kappa}}{|\kappa|} \right) \tag{6.2}$$

Where κ is the current partition, $|\kappa|$ is the number of words inside the partition and $nPIR_{\kappa}$ is the $nPIR$ value of the current partition κ, which measures the positive or negative influence of perturbing the tokens in κ.

Finally, the selected set of features, i.e. the partitioning producing the most informative explanation, is the one with $\max(K_{score})$. The K_{score} tends to assign a high influence to small clusters. The K_{score} ranges in [0,2]. The most informative local explanation possible would have a score of 2 which means that at least one feature in the partitioning, composed of 1 word, has $nPIR = 1$ and at least another, composed of 1 word as well, has $nPIR = -1$.

On the way around, the least informative local explanation has a score of 0. This happens when the algorithm finds K clusters of words all being neutral for the prediction of the class-of-interest, hence with $nPIR_{\kappa} = 0$. The output of the whole process represents the *most informative local-explanation* (Step ⑧).

6.4 Perturbation of Interpretable Features

After the extraction of the interpretable feature sets a perturbation phase is performed to introduce noise in the traditional phases of the predictive analytics to assess the impact of the perturbed features in the model outcomes. In the case of textual data, the perturbation can be performed by either *feature removal* or *feature substitution*.

The first considered strategy is the *feature removal* perturbation. All the extracted interpretable features are iteratively removed from the input text, producing new perturbed variations of it. Multiple blank spaces are merged into one and spaces before punctuation are removed to produce a new text that is written like humans. The perturbed variations of the input are then fed back into the model under analysis and its predictions are collected and analyzed to produce the *local explanation* report. Examples of explanations produced by feature removal perturbation are shown in Figures 6.4b, 6.4c and 6.4d. In these cases, the extracted features are actually removed from the input text. In Figure 6.4b shows Adjective-POS removal, Figure 6.4c shows sentence removal, and Figure 6.4d shows the removal of words identified by MLWE.

The *feature substitution* perturbation has been also experimented with. This strategy differs from the previous, the substitution perturbation introduces a new related concept that can cause a change in the prediction. The feature substitution perturbation

requires an additional step to select new words that will replace the ones in the current feature. We substitute words with their *antonyms*. The substitution with antonyms turned out to be very powerful in some specific cases (e.g. Adjective-POS perturbation), but in general it has several limitations like: words can have several antonyms, antonyms do not exist for some words (e.g. nouns) and, especially, the choice of the new words to insert in the substitution of the feature is task-specific (e.g. antonyms work with opposite class labels like *Positive* and *Negative* in sentiment analysis, but are not suited with independent class labels as in topic detection). Thus, the effectiveness of this perturbation strategy is highly affected by these limitations. Figures 6.4e and 6.4f show two examples of explanation performed using this technique. For the Adjective-POS features it is straightforward to find meaningful antonyms while for Verb-POS the result is very difficult to evaluate since verbs like {was, have} are substituted with {differ, lack}. This feature perturbation strategy remains an open task left for further inspection in future works in that new research issues emerged, such as:

- How to choose one antonym of a given word when multiple different antonyms exist?
- How substitute a word for which there are no antonyms?
- How to generalize the choice of the new words regardless of the task?

6.5 Prediction-Local Explanations

As introduced, to produce local explanations we analyze together the original outcome of a model against its perturbed versions. A local explanation consists of: a *textual explanation* and a *quantitative explanation*. Figure 6.4 and Table 6.1 show an example of a local explanation. In the example a BERT model has been trained to detect the sentiment of a textual document, either positive (P) or negative (N). Figure 6.4a shows the original input text.

Textual explanation. The *textual explanation* reports the most relevant sets of interpretable features for the model under analysis. This allows the user to understand the context in which they appear. Many sets of features can be extracted for each interpretable feature extraction technique. Figure 6.4 shows an example of textual explanations. Given the input document in Figure 6.4a, the model outputs a negative sentiment. So, the user can inspect the highlighted features (in red) in the textual explanations. Figures 6.4b, 6.4c, 6.4d, 6.4e, and 6.4f shows the most important concept exploited by the model while producing the predictions. For completeness, we reported also features that are not influent in Figures 6.4c and 6.4f to show how they would appear.

This film was very awful. I have never seen such a bad movie.

(a) Original text

This film was very awful. I have never seen such a bad movie.

(b) EXP1 : Adjective - POS feature extraction with removal perturbation.

This film was very awful. I have never seen such a bad movie.

(c) EXP2 : Sentence feature extraction with removal perturbation.

This film was very awful. I have never seen such a bad movie

(d) EXP3 : Multi-layer word embedding feature extraction with removal perturbation.

This film was very [awful] nice. I have never seen such a [bad] good movie

(e) EXP4 : Adjective-POS feature extraction with substitution perturbation.

This film [was] differ very awful. I [have] lack never seen such a bad movie.

(f) EXP5 : Verb-POS feature extraction with substitution perturbation.

Figure 6.4: Examples of a *textual explanation* report. The original text was labeled by BERT as *Negative* with a probability of 0.99. The most relevant features are highlighted in red. Removed features are in squared brackets.

Explanation	Feature f	L_o	L_f	$nPIR_f(N)$
EXP1	POS-Adjective	N	P	0.998
EXP2	Sentence	N	N	0.000
EXP3	MLWE	N	P	0.984
EXP4	POS-Adjective (sub.)	N	P	0.999
EXP5	POS-Verb (sub.)	N	N	0.000

Table 6.1: Quantitative explanation for example in Figure 6.4. P is the positive label, N is the negative label. The (sub.) suffix indicates that the substitution perturbation has been applied.

Quantitative explanation. The *quantitative explanation* shows the influence of each set of extracted features separately for each prediction by exploiting the nPIR introduced in Section 3. As already explained it allows to quantify the contribution of an interpretable feature over the prediction process sor a specific class-of-interest.

Table 6.1 shows the *quantitative explanations* for the *textual explanations* in Figure 6.4. Columns L_o and L_f reports the labels assigned by the model before and after their perturbation for each *interpretable feature*. Column nPIR the influence values calculated for the class-of-interest *Negative* (N). Perturbing the POS adjectives in Figure 6.4b (EXP1) or the MLWE cluster in Figure 6.4d (EXP3) the nPIR is very close to 1. This means that these sets of features are very relevant for the model outcome: *removing* one of these features will cause completely different outcomes from the model, changing the prediction from negative (N) to positive (P). Instead, the perturbation of the sentence in Figure 6.4c (EXP2) is not relevant at all for the model, showing a value of nPIR equal to 0. Thanks to the quantitative explanation is possible to understand that

features {awful, bad} and { was, awful, bad, movie } are the real reason why the model is predicting the negative class. The information contained in the sentence { This film was very awful } instead does not justify the model outcome alone, like the rest of the text that is also contributing to the prediction.
The *quantitative explanations* obtained through substitution instead (EXP4 and EXP5) have been also reported in Table 6.1. Even from these results, it is evident that the substitution perturbation has great potential in expressiveness when it is possible to find suitable antonyms. In the case of Adjective-POS substitution (EXP4), the quantitative explanation shows a $nPIR$ value close to 1. On the contrary, in the case of EXP5, verbs are replaced with semantically incorrect words (not antonyms) in the context of the phrases, showing no impact in the prediction process with a $nPIR$ equal to 0.

6.6 Per-class Model-Global Explanations

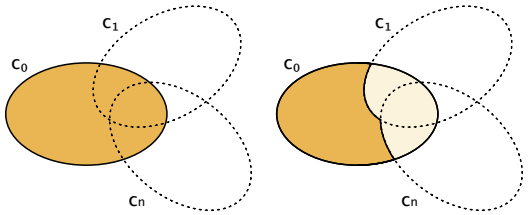
To evaluate the concepts that are globally affecting the prediction process we introduced a per-class *model-global* explanation process. A pool of local explanations computed for a corpus of documents can be aggregated to highlight misleading behaviors of the predictive model.

The per-class explanation process is carried out computing two new indexes: (i) the *Global Absolute Influence* (GAI), and (ii) the *Global Relative Influence* (GRI). The GAI score measures the global importance of *all* the words impacting on the class-of-interest, without distinction concerning other classes (Figure 6.5a). On the other hand, the GRI score evaluates the relevance of the words influential *only* (or mostly) for the class-of-interest, differently from other classes (Figure 6.5b).

Analyzing GAI and GRI scores, the user can easily understand which are the most influential inter- and intra- class concepts that are leading the prediction process at a global level. For instance, if a word is influential for the whole set of available classes, it will be characterized by a very high GAI value and a low GRI value close to 0. On the other side, a word might have a high GRI score for a specific class-of-interest, while having a very low GAI. This means that it is relevant for that specific class only.

The global explanations are computed for each available class-of-interest $c \in C$, analyzing the set of local explanations E. The local explanations $e_{d,f} \in E$ are computed for each document $d \in D$ and for each interpretable feature f.

Global Absolute Influence. The Global Absolute Influence value is computed following the process described in Algorithm 1. Given a set of local explanations E generated for a corpus of documents D, the algorithm computes the global score for each possible class-of-interest and for each lemma (base form of a word) contained in the most informative local explanations. Only MLWE explanations are used by the algorithm (line 2) since it is the only one exploiting the inner model knowledge. Then, given the MLWE explanations related to a document e_d, only the most influential one



(a) Absolute influential words. (b) Relative influential words.

Figure 6.5: Influential set of words at model-global level for class-of-interest c_0.

$\hat{e}_d$, i.e., the one with the highest $nPIR$, is selected (line 5) and the lemmas $L_{\hat{e}_d}$ are extracted from the tokens contained in the corresponding interpretable feature (line 6). The algorithm analyzes *lemmas* instead of *tokens* (words) in order to group together with their inflected forms, obtaining more significant results. Finally, the GAI score for the corresponding class-of-interest c and lemma l is updated (line 8) by summing the $nPIR$ score of the explanation $\hat{e}_d$, only if it is positively impacting the prediction (i.e., if $nPIR > 0$). The output of the algorithm is the set of *Global Absolute Influence* scores.

The GAI score will be 0 for all the lemmas that have always brought a negative influence on class c, and it will grow proportionally to the frequency and to the positive influence of each lemma positively influencing class c. The higher the GAI score, the most positively influential a lemma is for the model under analysis with respect to class c.

Algorithm 1: Global Absolute Influence.

Input: Local explanations E, Classes C .
Output: GAI scores for each class label and lemma.

```
1  GAI ← initHashMap(0);
2  E_MLWE ← getExplanationsMLWE(E);
3  for c in C do
4      for e_d in {E_MLWE : 0 ≤ d ≤ |D|} do
5          ê_d ← getMax_nPIR_Explanation(e_d,c);
6          L_ê_d ← lemmatizeTokens(ê_d.featureTokens);
7          for l in L_ê_d do
8              GAI(c,l) ← GAI(c,l) + Max[0, ê_d.nPIR];
9          end
10     end
11 end
12 return GAI;
```

Global Relative Influence. The Global Relative Influence score highlights the most influential and differentiating lemmas for each class-of-interest, discarding lemmas with multiple impacts on other classes. The GRI for a class-of-interest c and for a specific lemma l is defined as:

$$GRI(c, l) = Max[0, GAI(c, l) - \sum_{c_i \neq c}^{C} GAI(c_i, l)] \tag{6.3}$$

The GRI score is 0 when a lemma is more relevant for other classes than for the one under exam, while $GRI > 0$ if its influence is higher for class c than all the other classes. The higher the GRI value, the more specific the lemma influence is with respect to the class-of-interest.

Chapter 7

Experimental Results on Deep Natural Language Models

In this Chapter, we present experimental validation of our explanation framework in providing useful and human-readable insights on the decisions taken by black-box NLP models. We applied our explanation process to two classification use cases with two state-of-the-art NLP models, specifically LSTM and BERT.

To show the effectiveness of the proposed framework even in the case of deep NLP models we consider two different tasks addressed by two models with very different architectures.

Use case 1. We consider as a first use-case the binary *toxic comment classification*. In this task, we explain the reasons why a textual input comment has been classified as *clean* or *toxic*. A document is defined *toxic* if it contains for instance obscenity, threat, insult, identity attack, and explicit sexual content. For this task, we exploit an LSTM model trained on a civil comments dataset [10]. The LSTM model is composed of an embedding 300-dimensional layer, two bidirectional LSTM layers (with 256 units for each direction), and finally, a dense layer with 128 hidden units. We used for the embedding layer the GloVe [63] (with 300-dimensional vectors) pre-trained weights and we apply transfer learning. The fine-tuned LSTM model shows an accuracy of 90% on this binary problem.

Use case 2. We exploited as a second use-case a binary *sentiment analysis* task. Sentiment analysis consists of predicting if an input text is expressing positive or negative sentiment. In this case, we exploited a state-of-the-art BERT base (uncased) pre-trained model [23]. We fine-tuned the model on the *IMDB dataset* [49], which is a reference dataset for sentiment analysis. To fine-tune the BERT model we added a classification layer on top of the last encoder-transformer layer. The fine-tuned BERT model shows an accuracy of 86% on this sentiment analysis problem.

With the results reported in this chapter, we want to demonstrate that:

- Our explanation framework can be successfully applied to different deep learning models, and NLP tasks.
- the proposed feature extraction strategies are able to provide different kinds of explanations and they can be used in a complementary manner.
- We can extract informative explanations from both long and short text documents without limiting the interpretability.
- We provide to the end-user meaningful prediction-local and per-class model global explanations that can be analyzed to judge the quality of the model outcomes.

Section 7.1 shows a glance over the experimental results. Section 7.2 describes significant explanations produced at the prediction-local level. Then, Section 7.3 introduces a detailed analysis of explanations computed for each class-of-interest at a model-global level. Finally, Section 7.4 concludes the Chapter by comparing our solution in the NLP context with the state-of-the-art LIME [69].

7.1 Experiments at a glance

In this section, we provide an overview of the results we achieved exploiting the proposed explanation process in the context of NLP classification. We exploited all the feature extraction strategies described in Chapter 6 for each prediction-local explanation. Also, in the context of binary classification, only the nPIR is evaluated. Indeed, the nPIRP would not add any further information since the model can decide only between two classes and thus it is straightforward to understand from the nPIR if a target feature is influencing other classes. We consider, in this case, a local explanation to be informative when having a $nPIR \geq 0.5$. As already explained, the engine identifies and proposes automatically the most informative local explanation to the user. All the other explanations remain available to the users for further investigations.

We want to demonstrate the effectiveness of our feature extraction strategies. So, we measured, for each feature extraction separately, the percentage of documents for which we found at least one informative local explanation for the target class-of-interest. This analysis has been done first without combining the different features. Then, the analysis of the pairwise combination of features.

Table 7.1 shows the result for the first use-case. In the first use-case, we evaluate our explanation process on 400 textual documents: 202 of class *Positive* and 198 of class *Negative*. In total, we analyzed almost 100,000 prediction-local explanations, with 250 local explanations explored for each input document, on average. We note that the MLWE feature extraction is outperforming the other methods producing 75% of explanations with $nPIR \geq 0.5$. This confirms that exploiting the models' knowledge can improve the informativeness of the explanation process. Then, also Part-of-Speech is able to identify

Feature Extraction Type	No Comb	2 Comb
Part-of-speech	33%	70%
Sentence	22%	30%
MLWE	75%	86%
Overall	80%	90%

Table 7.1: Explanation of the BERT model: percentage of documents for which each feature extraction strategy produces at least one influential local explanation (i.e. with $nPIR \geqslant 0.5$), with and without combination of features. *Overall* is the percentage of documents for which at least one method found it.

Feature Extraction Type	Clean	Toxic	Clean/Toxic
Part-of-speech	8%	98%	53%
Sentence	2%	76%	39%
MLWE	12%	98%	55%
Overall	15%	99%	58%

Table 7.2: Explanation of the CUSTOM LSTM model: percentage of documents for which each feature extraction strategy produces at least one influential local explanation (i.e. with $nPIR \geqslant 0.5$), with combination of features, for the class labels *Clean* and *Toxic*.

meaningful explanations, especially when features are combined creating features representing more complex concepts. . For instance, the joint analysis of *adjectives+nouns* creates features composed of words like {bad, film} that better express a sentiment.

Table 7.2 reports the results for the second use-case. In the toxic comment use-case we analyzed 2250 documents: 1121 of class *Toxic* and 1129 of class *Clean*. In total we produced almost 160,000 prediction-local explanations. For class *Toxic*, we identify at least one informative explanation in the vast majority of cases. The MLWE is still, with the Part-of-Speech, the most suitable technique both reaching 98% of explained documents with $nPIR \geq 0.5$. Only the sentence-based feature extraction has a lower percentage of informative explanations. This is probably because toxic words are sparse, i.e. not concentrated in single sentences. On the other side, we note that finding informative explanations for the *Clean* class is harder. In this case, none of the features can explain more than 15% of the predictions. However, the explanation for this is intrinsic to the nature of the problem. Usually, standard documents can be considered clean. A document then becomes toxic because of the presence of specific linguistic expressions. Our, hypothesis that the LSTM model was not able to find any pattern of words that represents the *Clean* class. A further discussion on this will be provided in Section 7.3.

Criticize a black man and the left calls you a racist. Criticize a woman and you are a sexist. Now I will critcize you as a fool and you can call me intolerant.

(a) Original text

Criticize a black man and the left calls you a racist. Criticize a woman and you are a sexist. Now I will critcize you as a fool and you can call me intolerant.

(b) EXP1 : Adjective & Noun - POS feature extraction

Criticize a black man and the left calls you a racist. Criticize a woman and you are a sexist. Now I will critcize you as a fool and you can call me intolerant.

(c) EXP2 : Multi-layer word embedding feature extraction

Figure 7.1: Examples of *textual explanation* report for the input in Figure 7.1a originally labeled by custom LSTM model as *Toxic* with a probability of 0.98. The most relevant features are highlighted in red.

Explanation	Feature f	L_o	L_f	$nPIR_f(N)$
EXP1	POS-Adj&Noun	T	C	0.839
EXP2	MLWE	T	C	0.883

Table 7.3: Quantitative explanation for example in Figure 7.1. T is the Toxic label, C is the Clean label.

7.2 Evaluating Prediction-Local Explanations

In this section, we discuss a few meaningful examples of local explanations for both the use-cases. All the reported explanations are considered informative by the framework. We provide four different examples of prediction local explanation, two for use-case 1 and two for use case 2. Thus, to evaluate the reliability of the explanation process for the four input documents we try to answer the following questions:

- (Q1) "Why input text in Figure 7.1a is *Toxic* for the LSTM model?"
- (Q2) "Why input text in Figure 7.2a is *Toxic* for the LSTM model?"
- (Q3) "Why input text in Figure 7.3a is *Negative* for the BERT model?"
- (Q4) "Why input text in Figure 7.4a is *Negative* for the BERT model?"

Answering Q1. In the first example, the LSTM model for the input in Figure 7.1a predicts *Toxic*. The most influential features are shown in Figures 7.1b and 7.1c. We find that the most positive features for class *Toxic* are {black man, left, racist, woman, sexist, fool, intolerant}. Table 7.3 shows the quantitative explanation. The most informative explanations are extracted by combining adjectives and nouns (i.e., EXP2), and through MLWE (i.e., EXP2). It is interesting that the combination of *adjectives* and *nouns* is very relevant for this model. For instance, the word `black` alone does not make a comment toxic. Instead, the combination of words `black man` does. Furthermore, the POS feature extraction and the MLWE highlighted very similar sets of words. Also, thanks to the

They are nuts that have the Liberal government by the short and curlies, the good news is we see them a d they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(a) Original text

They are nuts that have the Liberal government by the short and curlies, the good news is we see them a d they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(b) EXP1 : Noun - POS feature extraction

They are nuts that have the Liberal government by the short and curlies, the good news is we see them a d they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(c) EXP2 : Multi-layer word embedding feature extraction

Figure 7.2: Examples of *textual explanation* report for the input in Figure 7.2a originally labeled by custom LSTM model as *Toxic* with a probability of 0.96. The most relevant features are highlighted in red.

Explanation	Feature f	L_o	L_f	$nPIR_f(N)$
EXP1	POS-Noun	T	C	0.608
EXP2	MLWE	T	C	0.999

Table 7.4: Quantitative explanation for example in Figure 7.2. T is the Toxic label, C is the Clean label.

explanation we can note that the model learned features like `black man` and `woman` to be positively influential for class *Toxic*. Also, despite the different methodology, EXP1 and EXP2 highlight the same set of words as influential. This confirms once again that exploiting the inner-model knowledge is a reliable strategy that can lead to correct and more interesting results.

Answering Q2. The LSTM predicts class *Toxic* also for the input document in Figure 7.2a. The most influential features are shown in Figures 7.2b and 7.2c. The *quantitative explanation* is in Table 7.4. In this case, the POS analysis identifies the *Nouns* as influential with a $nPIR \approx 0.61$(i.e. EXP1). Moreover, the MLWE finds the set {nuts, pigs, freak, suited, circus} as very important for the prediction with $nPIR \approx 0.99$ (i.e., EXP2). Also in this case, the two features are quite similar. However, the MLWE can find a smaller number of words that together cause the prediction of the target class. Moreover, the MLWE features appear to be significant for the performed prediction, enforcing the trustfulness of the model outcome. Thus, thanks to the provided explanations it is straightforward to confirm the correctness of the decision-making process performed by the LSTM model even in this case. Indeed, the explanation process identifies as influential specific words that are usually correlated to a toxic language.

How many movies are there that you can think of when you see a movie like this? I can't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. In fact, you'll probably find yourself battening down the hatches and riding the storm out. Why? ...Costner and Kutcher played their characters very well. I have never been a fan of Kutcher's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner's ability. I don't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just don't expect a smooth ride. 7 of 10 but almost a 6.

(a) Original text

How many movies are there that you can think of when you see a movie [...] I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT ... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. [...] I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner 's ability. [...] But some of these characters were used too much. [...] Just do n't expect a smooth ride. 7 of 10 but almost a 6.

(b) EXP1 : Adjective - POS feature extraction

How many movies are there that you can think of when you see a movie like this? I can't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of [...]

(c) EXP2 : Sentence feature extraction

How many movies are there that you can think of when you see a movie like this? [...] See the movie despite its downfalls there is enough to make it interesting and maybe make it appear clever. [...]

(d) EXP3 : Multi-layer word embedding feature extraction

Figure 7.3: Examples of *textual explanation* report for the input in Figure 7.3a wrongly labeled by BERT as *Negative* with a probability of 0.99. The most relevant features are highlighted in red.

Explanation	Feature f	L_o	L_f	$nPIR_f(N)$
EXP1	POS-Adjective	N	P	0.884
EXP2	Sentence	N	P	0.663
EXP3	MLWE	N	P	0.651

Table 7.5: Quantitative explanation for example in Figure 7.3. P is the positive label, N is the negative label.

Answering Q3. In this case, the BERT model wrongly predicts the sentiment of the input in Figure 7.3a as *Negative*. Indeed, the expected label is *Positive*. We want to understand what led to this prediction by exploiting our explanation process. Figure 7.3 reports the *textual explanations*. Table 7.5 shows the corresponding *quantitative explanations*. We find three main explanations for the *Negative* class. The top informative feature is represented by Adjectives-POS in the sentence in Figure 7.3c and MLWE in Figure 7.3d).

The perturbation of adjectives in Figure 7.3b shows that general words like {many, other, big, ..., smooth} are considered influential for the prediction process with $nPIR \approx 0.88$. They have a strong impact on this prediction and this may lead to not trustful results. Also, the perturbation of the singular sentence in Figure 7.3c is leading to a *Positive* prediction with a $nPIR \approx 0.66$. Finally, MLWE features in Figure 7.3d reports a set of words composed of just two instances of a single general word, {there}. By removing these two instances the prediction changes from *Negative* to *Positive* showing a $nPIR \approx 0.651$. For this reason, we could say that the prediction process for the class *Negative* performed for this text can not be trusted. As a general rule, if the perturbation of non-meaningful small portion of input leads to a change in the prediction, then the process itself must be questioned. In this case, the model was very uncertain about the prediction for the document. This uncertainty can not be discovered either having the ground-truth for comparison since one could not tell which is the problem without a detailed exploration of the result. Instead, our explanation process has been able to identify this uncertainty by putting in evidence that just a couple of words made the difference between a result or another. Then, also the global behavior of the model should be questioned. In Section 7.3 we will further discuss the global behavior of this BERT model.

Answering Q4. The BERT model correctly predicts class *Negative* for the input in Figure 7.4a. For comparison, we discuss here both informative and non-informative explanations. The reported explanations are the adjective in Figure 7.4b, verb in Figure 7.4c, adjective and verb in Figure 7.4d, sentence in Figure 7.4e, and MLWE in Figure 7.4f. Their nPIR values are reported in Table 7.6. We note that only the EXP3, EXP4, and EXP5 provide meaningful information. Instead, the EXP1 and EXP2, adjective and verb respectively, led to uninformative explanations.

From EXP1 and EXP (Figures 7.4b and 7.4c respectively), we note that the different parts-of-speech, taken separately one at a time, are not influential for class *Negative*. Indeed, Table 7.6 shows a very low nPIRof about 0.003 and 0.137 for the corresponding explanations. In these cases, the explanation process starts exploring pairwise combinations of features to analyze the contribution of more complex concepts. As result, the combination of adjectives and verbs produced EXP3 in Figure 7.4d a new very influential feature with $nPIR \approx 0.915$. On the other hand, EXP4 in Figure 7.4e shows a single influential sentence as an explanation with $nPIR \approx 0.638$. Finally, the MLWE analyzes a partitioning composed of 15 different sets of words and highlight EXP5 in Figure 7.4f as the most influential, reaching a $nPIR \approx 0.899$.

Analyzing the content of EXP3, EXP4, and EXP5 explanations, it might seem that the word {trivialized} is the main responsible for the prediction. Although, also EXP2, which is considered not informative, contains the same word. So, we could say that this prediction is not influenced by single words. Instead, it is the combination of different tokens that lead the prediction process. From EXP3 instead, we note that the joint perturbation of adjectives and verbs affects the outcomes of the model. However, the joint

There were so many classic movies that were made where the leading people were out-and- out liars and yet they are made to look good. I never bought into that stuff. The "screwball comedies" were full of that stuff and so were a lot of the Fred Astaire films. Here, Barbara Stanwyck plays a famous "country" magazine writer who has been lying to the public for years, and feels she has to keep lying to keep her persona (and her job). She even lies to a guy about getting married, another topic that was always trivialized in classic films. She's a New York City woman who pretends she's a great cook and someone who knows how to handle babies, etc. Obviously she knows nothing and the lies pile up so fast you lose track. I guess all of that is supposed to be funny because lessons are learned in the end and true love prevails, etc. etc. Please pass the barf bag. Most of this film is NOT funny. Stanwyck was far better in the film noir genre. As for Dennis Morgan, well, pass the bag again.

(a) Original text

There were so many classic movies that were made where the leading people were out-and- out liars and yet they are made to look good. I never bought into that stuff. The "screwball comedies" were full of [...] plays a famous "country" [...] getting married, another topic that was always trivialized in classic films. [...] she's a great cook and someone [...] supposed to be funny because lessons are learned in the end and true love [...] bag. Most of this film is NOT funny. Stanwyck was far better in the film noir genre. [...]

(b) EXP1 : Adjective - POS feature extraction

There were so [...] that were made where the leading people were out-and- out liars and yet they are made to look good. I never bought into that stuff. The "screwball comedies" were full of that stuff and so were a lot [...] Barbara Stanwyck plays a famous "country" magazine writer who has been lying to [...] she has to keep lying to keep her persona (and her job). She even lies to a guy about getting married, another topic that was always trivialized in classic films. She 's a New York City woman who pretends she's a great cook and someone who knows how to handle babies, etc. Obviously she knows nothing and the lies pile up so fast you lose track. I guess all of that is supposed to be funny because lessons are learned in the [...] Please pass the barf bag. Most of this film is NOT funny. Stanwyck was far [...] well, pass the bag again.

(c) EXP2 : Verb - POS feature extraction

There were so many classic movies that were made where the leading people were out-and- out liars and yet they are made to look good. I never bought into that stuff. The "screwball comedies" were full of that stuff and so were a lot of the Fred Astaire films. Here, Barbara Stanwyck plays a famous "country" magazine writer who has been lying to the public for years, and feels she has to keep lying to keep her persona (and her job). She even lies to a guy about getting married, another topic that was always trivialized in classic films. She 's a New York City woman who pretends she 's a great cook and someone who knows how to handle babies, etc. Obviously she knows nothing and the lies pile up so fast you lose track. I guess all of that is supposed to be funny because lessons are learned in the end and true love prevails, etc. etc. Please pass the barf bag. Most of this film is NOT funny. Stanwyck was far better in the film noir genre. As for Dennis Morgan, well, pass the bag again.

(d) EXP3 : Adjective & Verb - POS feature extraction

Figure 7.4: Examples of *textual explanation* report for the input in Figure 7.4a originally labeled by BERT as *Negative* with a probability of 0.99. Features found are highlighted in red.(Continue)

perturbation can be considered a good measure of robustness for this prediction. Moreover, EXP5 is characterized by an apparently random pool of relevant words. However, we could say also that MLWE features are more precise w.r.t. EXP3 with just a small penalty on the nPIR score. The MLWE strategy exploits the model's knowledge to find a concise set of words independently from their part-of-speech and sentence. So, also the resulting explanations can be considered more reliable by the end-user.

[...] She even lies to a guy about getting married, another topic that was always trivialized in classic films. [...]

(e) EXP4 : Sentence feature extraction

[...] I never bought into that stuff. The "screwball comedies" were full of that stuff and so were a lot of the Fred Astaire films. Here, Barbara Stanwyck plays a famous "country" magazine writer who has been lying to the public for years, and feels she has to keep lying to keep her persona (and her job). she even lies to a guy about getting married, another topic that was always trivialized in classic films. she's a new york city woman who pretends she's a great cook and someone who knows how to handle babies, etc. Obviously she knows nothing and the lies pile up so fast you lose track. I guess all of that is supposed to be funny because lessons are learned in the end and true love prevails, etc. [...] Most of this film is not funny. Stanwyck was far better in the film noir genre. as for Dennis Morgan, well, pass the bag again

(f) EXP5 : Multi-layer word embedding feature extraction

Figure 7.4: (Continued) Examples of *textual explanation* report for the input in Figure 7.4a originally labeled by BERT as *Negative* with a probability of 0.99. Features found are highlighted in red.

Explanation	Feature f	L_o	L_f	$nPIR_f(N)$
EXP1	POS-Adjective	N	N	0.003
EXP2	POS-Verb	N	N	0.137
EXP3	POS-Adj&Verb	N	P	0.915
EXP4	Sentence	N	P	0.638
EXP5	MLWE	N	P	0.899

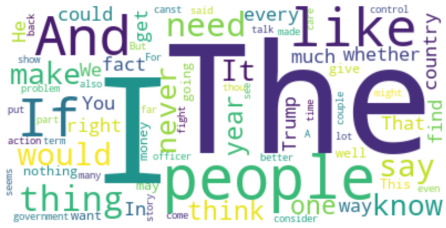
Table 7.6: Quantitative explanation for example in Figure 7.4. P is the positive label, N is the negative label.

7.3 Evaluating Per-Class Model Explanations

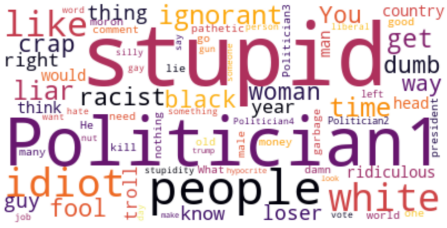
Exploiting the prediction-local explanations computed for all the input documents we provide per-class model-global insights about the leading input patterns. For each use-case we report, separately for each class-of-interest, the GAI and GRI scores of each input token under the form of word-clouds. The sizes of words in the word-clouds are proportional to the GAI or GRI scores computed separately for each class-of-interest. Also, note that the relative size of the word size is local to each word-cloud. Two words with the same size in different word-clouds do not necessarily have the same score. Also, two words with the same size in the same word-cloud have a similar score.

Use case 1. Figure 7.5 reports the GAI and GRI score for the toxic comment classification task. Class *Toxic* is reported in Figures 7.5a and 7.5c), while class *Clean* is reported in Figures 7.5b and 7.5d)respectively. The GAI word-clouds (Figures 7.5a and 7.5b) show that the two classes are influenced by a non-overlapping set of words. So, if a word is characterizing toxic language it would not be associated to clean language. We also note that the model learned to distinguish toxic comments when input documents include terms that are strongly related to discrimination, or racism. Instead, there is no specific pattern of words that characterize clean comments. This confirms the hypothesis for which the model is able to recognize toxic words while all the others are

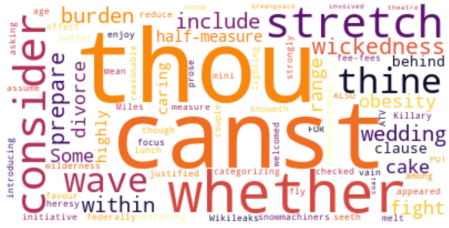
(a) GAI *Toxic* class.



(b) GAI *Clean* class.



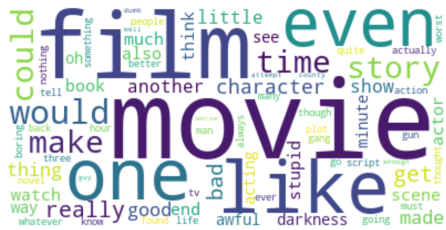
(c) GRI *Toxic* class.



(d) GRI *Clean* class.

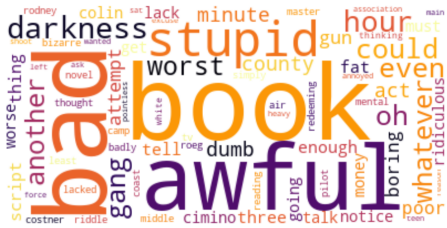
Figure 7.5: Global explanation of toxic comment classification with LSTM.

(a) GAI *Positive* class.



(b) GAI *Negative* class.

(c) GRI *Positive* class.



(d) GRI *Negative* class.

Figure 7.6: Global explanation of sentiment analysis with BERT.

correctly considered non-toxic.

The ***GRI*** word-clouds (Figures 7.5c and 7.5d) show then which are the concepts that the model is using to differentiate the two classes. They highlight that class *Toxic* is influenced by general insults like `stupid`, `idiot`, and `ignorant` and terms related to minorities and genders like `woman`, `black`, `white`, `gay`. This means that the model learned to distinguish toxic comments when these terms are present. Also, we discovered that politician family names, anonymized as `Politician1`, `Politician2`, etc.,

are meaningfully related to toxic comments by the model. These last result highlights the necessity of exploiting detailed explanations when dealing with black-box models. Indeed, if models are not carefully trained, they can learn from sensible content including prejudices and various forms of bias that should be avoided in critical contexts. For instance, associating a person's family name to a class raises ethical issues.

Use case 2. We analyzed the prediction-local explanations for 400 input texts in the *sentiment analysis* use-case. Doing so we extract per-class global insights about the BERT model. Figure 7.6 shows the GAI and GRI word-clouds for classes *Positive* and *Negative*.

In this case, the GAI word-clouds for classes *Positive* in Figure 7.6a and the *Negative* in Figure 7.6b report that several words like `story`, `movie`, `film`, `like` are impacting at the same time on both classes. Thus, the model learned overlapping concepts that do not express directly a sentiment. However, if these concepts are taken together to the context, they can actually express a sentiment (e.g. `This film is not as good as expected`).

The GRI of class *Positive* in Figure 7.6c reports also that words like `movie` and `film` are still very relevant for the model. Instead, these words disappear in class *Negative* in Figure 7.6d. Interestingly, it emerges that this class is highly characterized by the word `book`. Investigating this in the dataset, we discovered that negative comments are often related to movies extracted from books. This is a form of bias that the model has learned. From one side, the model learned what the dataset contained. On the other, though the automatic evaluation of a movie should not take into account the fact that it is inspired from by book or not. Otherwise, good movies inspired by books may be classified erroneously as *Negative* just because of the comparison with the book.

The GRI highlight that other influential words for class *Positive* are words strongly related to the corresponding sentiment, e.g. `good`, `great`, `best`, `love`. Similarly, the negative class *Negative* is characterized by concepts like `worst`, `bad`, `awful`. In this case, the model behaves as expected.

We can conclude that per-class model-global explanations are useful tools to better understand the decision-making process of a model. In our experiments, it highlighted the presence of prejudices and/or biases. Also, the data scientist responsible for the quality of the model may also use them to decide if and which corrective actions have to be taken to make the decision-making process more reliable.

Further Global Insights. Contrariwise to the short input text classified by LSTM, the BERT predictive model has been exploited to analyze much more complex and longer inputs. The average number of tokens in the texts classified by the LSTM model is 51, whereas the average for BERT is almost 225 tokens per input text.

Considering the more complex classification task of the BERT model, we analyze which are the characteristics of the input data that are influencing the prediction process. We do this by exploiting the information extracted during the explanation process.

From Table 7.1 we notice that the perturbation of a single part-of-speech (POS) is causing a change in the predicted class only in 33% of the documents, of which: 14% due to adjectives, 10% verbs, 10% conjunctions, 8% adverbs, 7% by pronouns, and 6% nouns (the percentages do not sum to 33% since several part-of-speech features can characterize each textual document).

Regarding the analysis of the features extracted by MLWE, BERT showed the tendency to be influenced by the perturbation of singular words (example of this in Section 7.2), which is typically an undesired behavior. Figure 7.7 shows the relative size of the features extracted through MLWE for each local explanation, computed as the number of tokens in the cluster exploited in each local explanation over the number of tokens in the corresponding input text. This measures the percentage of input text that should be perturbed to change the model prediction. We notice that there are few documents for which the perturbation of 2 or 3 words causes a change in the predicted outcome, whereas most documents usually require at least 30% of the text to be perturbed before changing the outcome. We can conclude that the fine-tuned BERT model is on average particularly sensitive to the context of whole phrases since there is no specific part-of-speech or singular word influencing the decision-making process at a model-global level.

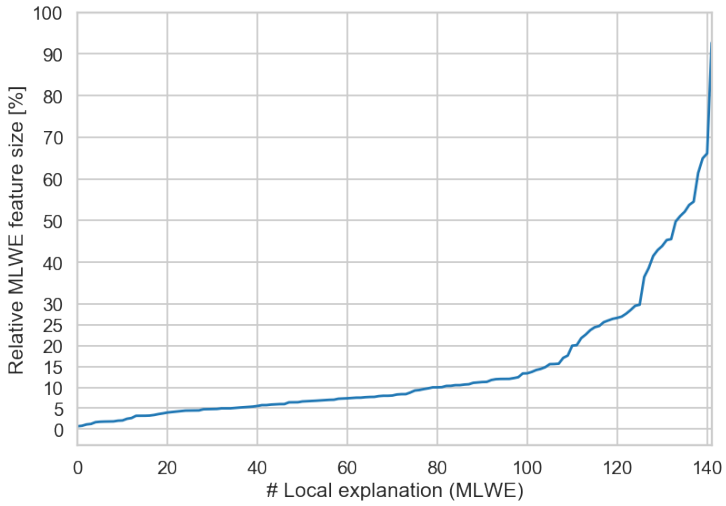


Figure 7.7: Relative size of the features extracted with MLWE strategy (i.e. size of the most influential clusters of words) for all the informative local explanations. The relative size is calculated, for each local explanation, as the number of tokens in the cluster over the number of tokens in the input text.

7.4 Comparisons of Model-Agnostic and Model-Aware explanations

In this section, we compare our explanation process with the state-of-the-art LIME [69]. The main difference between the two methods is that we consider the knowledge of the model to produce the explanations. Instead, LIME's explanations are extracted by exploiting a model-agnostic approach.

To better understand the comparison process, let us describe briefly how the local explanations are computed by LIME. Given an input text, LIME randomly perturbs the tokens in the input text and produces a number of local variations. The labels for the local variations are predicted with the target black-box model. The local variations along with the predicted labels constitute a new dataset describing the local behavior of the model. The local dataset is used to train a linear surrogate model which is interpretable. The coefficients of the surrogate model are exploited as prediction-local explanations.

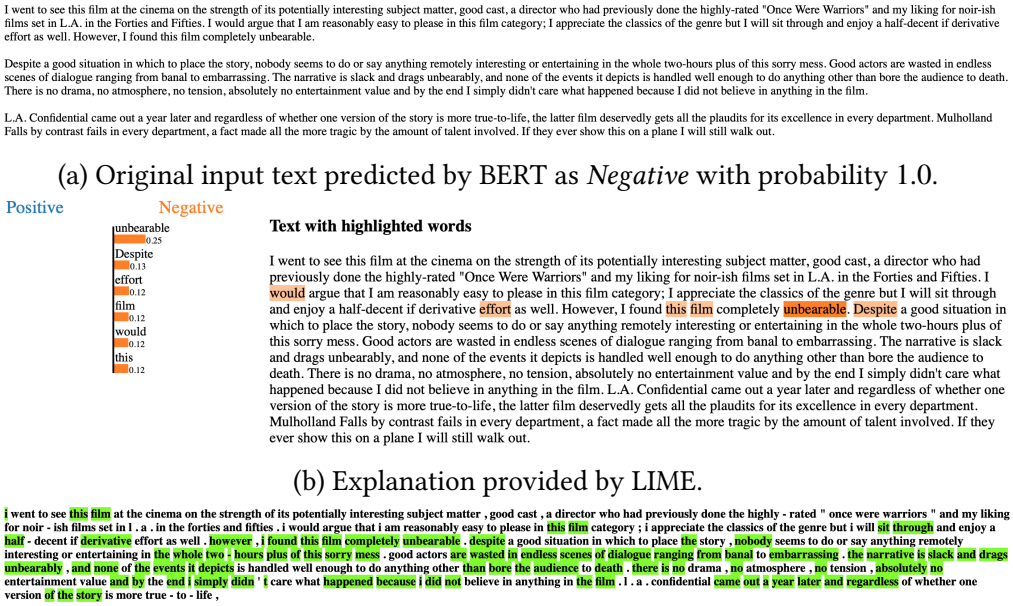
Figures 7.8 and 7.9 show the comparison of two prediction-local explanations produced by LIME and by our framework. The two examples represent two predictions extracted from use-case 2, i.e. the sentiment analysis task performed with BERT. The predictions for input texts in Figures 7.8a and 7.9a are *Negative* and *Positive* and they are both correct.

From a first analysis, LIME's explanations highlight a very small set of influential words. Figure 7.8b identifies the word `unbearable` as the main responsible for the class-of-interest. Similarly, in Figure 7.9b LIME highlight the set {`what`, `and` , `I`, `personally`, `found`, `very`} as influential for the prediction *Positive*. Both LIME's explanations are very easy to read and if not analyzed properly they may seams correct and trustworthy.

On the other side, our explanation framework highlights a much larger number of tokes as influential in Figures 7.8c and 7.9c for the two predictions respectively. The reported explanations are extracted with the MLWE strategy to fully exploit the knowledge of the model. The influential tokens have been colorized accordingly to their nPIR values, i.e. 0.99 and 0.82 respectively.

So, now the question is "which of the two frameworks is providing the more reliable explanation?". To answer this question we took the explanations provided by LIME and we removed them from the input text. Then, this newly perturbed version of the input is given in input to the BERT model which produces a new prediction. If the model is actually affected by the concepts found by LIME, then the new outcomes of the model would show a variation. Interestingly, we discovered that the perturbation of the tokens explained by LIME is not affecting at all the prediction process in both examples. The probabilities of belonging to the classes-of-interest do not change before and after the perturbation. This demonstrates that approaching the explanation process with model agnostic strategies can lead to not reliable results. Indeed, LIME's explanations are suffering from a large approximation introduced by the surrogate model.

Instead, our explanation process is intrinsically optimizing the informativeness of the explanations by using the black-box model itself. The sets of influential features found with our approach are larger w.r.t. the ones from LIME. However, they are more reliable since, if perturbed they drift the predictions to the opposite class. In particular, perturbing features in Figure 7.8c are lowering the probability of class *Negative* from 1.0 to 0.002. Similarly, the perturbation of features in Figure 7.9c is lowering the probability of class *Positive* from 0.999 to 0.186. This shows once more that our model-aware explanation process is more reliable, even if the produced explanations may seamless readable. Moreover, we can say that our explanation process is reliable by definition, since it does not relies on any other knowledge than the one coming from the target prediction process, i.e. the input data, the model's knowledge, and the model's outcomes.


I went to see this film at the cinema on the strength of its potentially interesting subject matter, good cast, a director who had previously done the highly-rated "Once Were Warriors" and my liking for noir-ish films set in L.A. in the Forties and Fifties. I would argue that I am reasonably easy to please in this film category; I appreciate the classics of the genre but I will sit through and enjoy a half-decent if derivative effort as well. However, I found this film completely unbearable.

Despite a good situation in which to place the story, nobody seems to do or say anything remotely interesting or entertaining in the whole two-hours plus of this sorry mess. Good actors are wasted in endless scenes of dialogue ranging from banal to embarrassing. The narrative is slack and drags unbearably, and none of the events it depicts is handled well enough to do anything other than bore the audience to death. There is no drama, no atmosphere, no tension, absolutely no entertainment value and by the end I simply didn't care what happened because I did not believe in anything in the film.

L.A. Confidential came out a year later and regardless of whether one version of the story is more true-to-life, the latter film deservedly gets all the plaudits for its excellence in every department. Mulholland Falls by contrast fails in every department, a fact made all the more tragic by the amount of talent involved. If they ever show this on a plane I will still walk out.

(a) Original input text predicted by BERT as *Negative* with probability 1.0.

(b) Explanation provided by LIME.

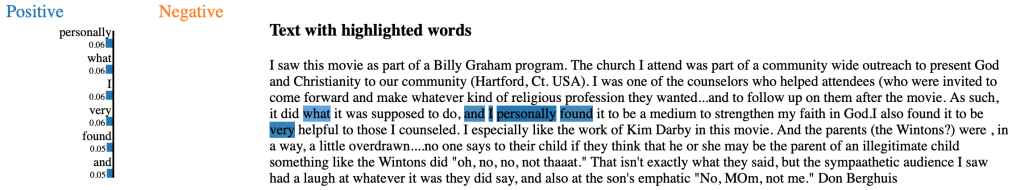
i went to see this film at the cinema on the strength of its potentially interesting subject matter , good cast , a director who had previously done the highly - rated " once were warriors " and my liking for noir - ish films set in l . a . in the forties and fifties . i would argue that i am reasonably easy to please in this film category ; i appreciate the classics of the genre but i will sit through and enjoy a half - decent if derivative effort as well . however , i found this film completely unbearable . despite a good situation in which to place the story , nobody seems to do or say anything remotely interesting or entertaining in the whole two - hours plus of this sorry mess . good actors are wasted in endless scenes of dialogue ranging from banal to embarrassing . the narrative is slack and drags unbearably , and none of the events it depicts is handled well enough to do anything other than bore the audience to death . there is no drama , no atmosphere , no tension , absolutely no entertainment value and by the end i simply didn ' t care what happened because i did not believe in anything in the film . l . a . confidential came out a year later and regardless of whether one version of the story is more true - to - life ,

(c) Explanation provided by our framework with MLWE feature extraction. The nPIRis 0.99.

Figure 7.8: Comparison of explanation between LIME and our framework for class-of-interest *Negative*.

I saw this movie as part of a Billy Graham program. The church I attend was part of a community wide outreach to present God and Christianity to our community (Hartford, Ct. USA). I was one of the counselors who helped attendees (who were invited to come forward and make whatever kind of religious profession they wanted...and to follow up on them after the movie. As such, it did what it was supposed to do, and I personally found it to be a medium to strengthen my faith in God.I also found it to be very helpful to those I counseled. I especially like the work of Kim Darby in this movie. And the parents (the Wintons?) were , in a way, a little overdrawn....no one says to their child if they think that he or she may be the parent of an illegitimate child something like the Wintons did "oh, no, no, not thaaat." That isn't exactly what they said, but the sympaathetic audience I saw had a laugh at whatever it was they did say, and also at the son's emphatic "No, MOm, not me." Don Berghuis

(a) Original input text predicted by BERT as *Positive* with probability 1.0.



(b) Explanation provided by LIME.

i saw this movie as part of a billy graham program . the church i attend was part of a community wide outreach to present god and christianity to our community (hartford , ct . usa) . i was one of the counselors who helped attendees (who were invited to come forward and make whatever kind of religious profession they wanted . . . and to follow up on them after the movie . as such , it did what it was supposed to do , and i personally found it to be a medium to strengthen my faith in god . i also found it to be very helpful to those i counseled . i especially like the work of kim darby in this movie . and the parents (the wintons ?) were , in a way , a little overdrawn . . . . no one says to their child if they think that he or she may be the parent of an illegitimate child something like the wintons did " oh , no , no , not thaaat . " that isn ' t exactly what they said , but the sympaathetic audience i saw had a laugh at whatever it was they did say , and also at the son ' s emphatic " no , mom , not me . " don berghuis

(c) Explanation provided by our framework with MLWE feature extraction. The nPIRis 0.82.

Figure 7.9: Comparison of explanation between LIME and our framework for class-of-interest *Negative*.

Chapter 8

Assessing and Trusting Models' Performances Over-Time

In this chapter, we address the problem of managing the reliability of a predictive model over-time. It is widely recognized that the performance of a predictive model can degrade over time due to factors like the presence of concept drift [28, 103, 37, 11, 30]. To this aim, we introduce a new general framework tailored to Concept-Drift Management which enables the unsupervised monitoring of predictive performance over-time. With this framework we aim to introduce a new level of model interpretability, addressing the current limitations of explanation frameworks that do not take into account over-time performance degradation. In particular, our contributions can be summarized in:

- Our approach has been developed to address different use-cases and ML/AI architectures, e.g. standard ML pipelines, DCNN, Word2Vec, BERT.
- We consider also modern Big-Data use cases where scalability is a requirement. So, we introduced a new scalable index, namely Descriptor Silhouette, that we exploit to measure the degradation of input data distribution.
- We introduced a new quantitative definition of model degradation that measures the presence of concept-drift separately for each class-of-interest.
- We validate our approach on unstructured data analyzed by modern state-of-the-art models, i.e., DCNN, Word2Vec, and BERT. Furthermore, we deeply analyze the scalability performance of our approach.

A preliminary version of this work has been published in [100, 19].

The chapter is organized as follows. Section 8.1 introduces the overall methodology to automatically provide up-to-date predictive models when required. Section 8.2 presents the new concept-drift detection strategy proposed to explain how the predictive performance degrades over time, while Section 8.3 discusses some experimental results that validate the proposed approach.

8.1 A framework for Concept-Drift Management

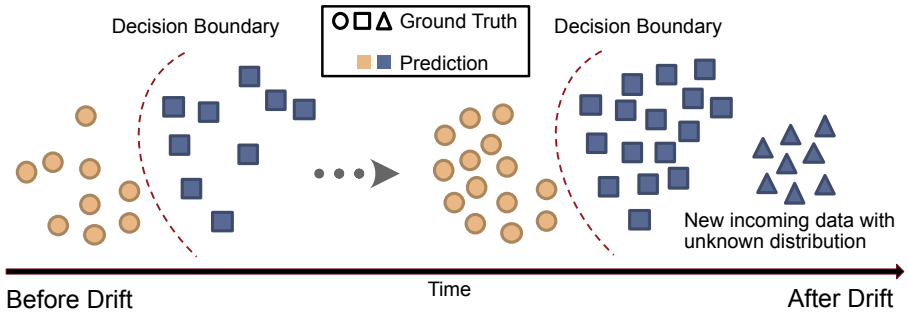


Figure 8.1: Predictive model before and after the occurrence of concept drift.

The performances of predictive models are subject to degradation over-time because of the occurrence of new data distributions, that are unknown at training-time. This phenomenon is known as concept-drift. A symbolic example of this phenomenon is shown in Figure 8.1. In the example, before the arrival of drifting data, the *Decision Boundary* of the predictive model is able to distinguish between *Circles* and *Squares*. However, as soon as a new *Triangle* data arrives, the model has to decide a label among the ones that he has learned at training-time. Unfortunately, this dummy model has never learned the concept of *Triangle*, thus it wrongly assigns the new data to the squares class.

Assessing and managing the quality of the model's outcomes over-time requires dealing with the presence of concept drift. To this aim, it is necessary to introduce further steps in the standard online analytical pipeline.

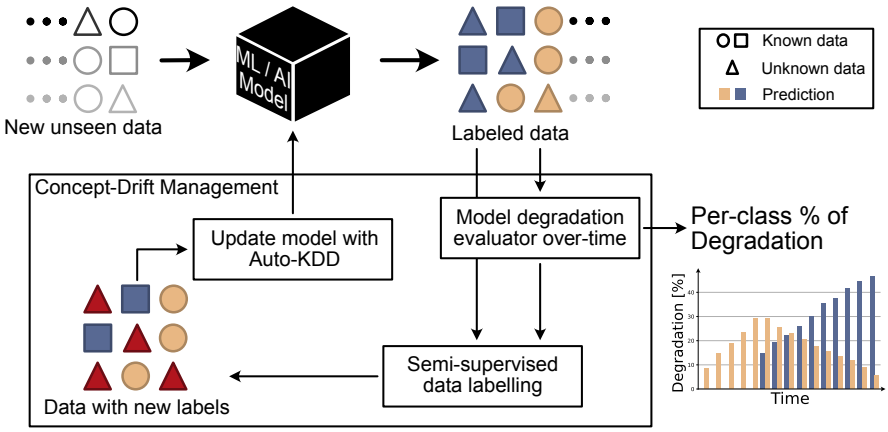


Figure 8.2: Concept-Drift management framework.

In [19] we proposed a new framework to manage concept drift that introduced a novel *Model Degradation Evaluation Over-Time* component. Figure 8.2 shows the main

components of the proposed framework. Consider a predictive model deployed in production that is analyzing a stream of data.

The Concept-Drift Management framework continuously monitors the presence of drifting distributions of data through the *Model Degradation Evaluation Over-Time*. The outcomes of this evaluation step are proposed to the user as well so that he can decide if to trust the model's prediction over-time.

Then, before updating the model, it is necessary to isolate the samples of data affected by concept-drift. To this aim, a *Semi-Supervised Data Labelling* step has been introduced. Due to the massive quantity of data collected and analyzed every day by modern systems, the labeling step is widely recognized as a huge bottle-neck in the machine learning pipeline. This step aims to support domain experts in identifying a meaningful pattern in the newly correlated data and allowing to assign semantically meaningful labels to the new automatically-discovered classes of data. A small subset of representative samples of the new classes will be manually inspected by domain experts. Their label assignments will be used for the remaining samples in each corresponding class.

Lastly, the *Data with New Labels* is exploited to *Update the Model with Auto-KDD* (Knowledge Discovery Process) fitting the new incoming data distributions and classification labels. This step can be automatically triggered based on the results of the previous ones, e.g., when the model degradation is higher than a given threshold. The latter step has already seen applications of state-of-the-art approaches tailored to specific data types, such as in [3] for predictive maintenance, in [24, 66] for addressing topic detection among textual document collections, and for network traffic characterization in [4].

The contribution of this chapter focuses on the *Model Degradation Evaluation Over-Time*, which we claim as a new core component of novel Explainable Artificial Intelligence pipelines. We are proposing a novel unsupervised process able to automatically detect per-class concept drifts and to manage predictive model rebuilding by evaluating the degradation of predictions of data streams. Specifically, we aim at identifying when additional or different class labels are required, without requiring ground-truth data labels for the new data, and with a scalable approach.

8.2 Evaluating model's performance over-time

Due to the absence of ground-truth labels when the model is deployed in real systems, it would not be possible to exploit standard validation techniques to assess the performance of the model. Thus, we propose an unsupervised approach to highlight whether the distribution of known classes is starting to drift and in turn when the prediction process is no more reliable. Figure 8.3 shows the main steps of the model's performance degradation evaluation over-time.

Let's consider a prediction process of a stream of data. A predictive model is first

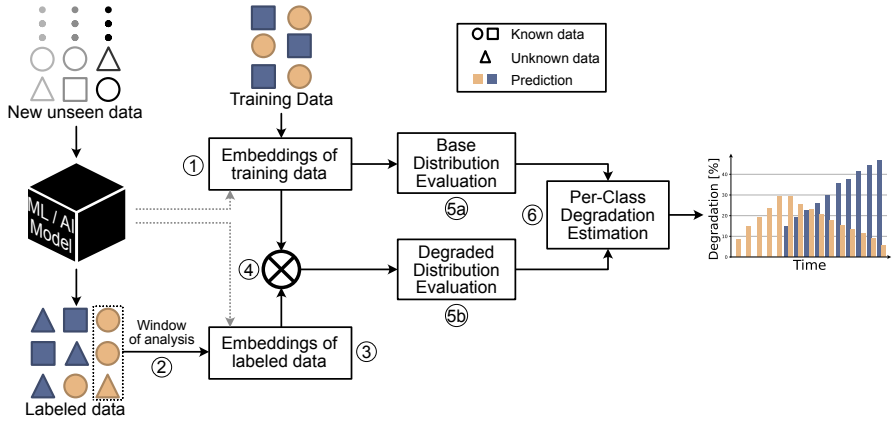


Figure 8.3: Model's Performance Degradation Evaluation Over-Time.

trained to exploit a labeled training dataset, thus, the model associates the distribution of the labeled samples to each class. The model's degradation evaluation starts by persisting the knowledge contained in the training data with an embedded form (Step ①). This operation is performed one-time. We will define in Section 8.2.1 what we mean by embedded form.

Then, the model is deployed and exploited to predict a stream of data. As soon as the predictions are produced, the labeled samples are collected and analyzed by windows (Step ②) and their information is persisted under the form of embedded data (Step ③) as well. The embedded data of the window of analysis is then merged with the embedded data representing the training data distribution (Step ④).

The distributions of the training data, i.e., the base distribution, and of the training plus the new incoming data, i.e., the eventually degraded data, are then evaluated (Steps ⑤a, and ⑤b). The cohesion and the separation for each class label separately are measured. Further details about the measurement of cohesion and separation will be reported later in this Chapter as well.

Finally, the outcomes of the evaluation phase of the base and the degraded distributions are used to estimate the degradation of the model's performance over-time (Step ⑥). At each time interval, as large as the window of analysis, the evaluation process provides the percentage of performance degradation for each known class label. The higher is the degradation, the higher is the probability of the presence of concept drift, reducing, in turn, the reliability of the model itself.

While new data are continuously classified by the model over time, the assessment of the predictive performance has to be performed periodically. Also, the evaluation of the base and degraded distributions depend on the cardinality of the new incoming data.

8.2.1 Embedding the model's knowledge

Exploiting an embedded representation of data has several advantages. The literature is prone of techniques and examples where the input data is transformed in its embedded form. Word embeddings in Natural Language Processing [40, 63, 25], image embeddings in image processing [58, 34], and more in general all the techniques used to reduce the dimensionality of a raw input sample are just a few of the examples that can be done. These techniques are able to reduce the complexity of the input data putting in evidence their semantical meaning in a geometrical fashion. Also, modern models like Word2Vec [40], BERT [23], and VGGs [80] incorporate the knowledge of thousands of concepts under the form of numerical tensors. We already saw in previous Chapters 4.2 and 6.2 how the knowledge of these models can be exploited to extract meaningful information about the prediction process. Similarly, also in this context, the inner knowledge of the model comes in help when dealing with monitoring the distribution of new incoming data.

In our predictive performance evaluation approach, the embedding representation of the input samples shows several advantages as well:

- The dimensionality of the input data is significantly reduced, lowering the requirements of memory and processing power required to process a large amount of data;
- The input sample is projected to a lower geometrical space, putting more in evidence where the shift in the distribution is occurring;
- The knowledge of most of the modern AI models can be accessed to extract the embedded representation of the input sample, as already explained in previous Chapters;
- The privacy of the real content of the data can be preserved;

It is not always possible to apply a data embedding process to the input data. In particular, three different scenarios are possible: (i) the analytical pipeline deployed along with the predictive model does not include an embedding step, (ii) the analytical pipeline deployed along with the predictive model already includes an embedding step (e.g. text classification exploiting Word2Vec embeddings), and (iii) the deployed model is a Deep Neural Network from which an embedded representation of the input can be extracted.

In the first scenario, an embedding process is not available and it should be avoided to preserve as much information as possible from the input data. Indeed, the predictive model has been probably trained on the input data after a custom preprocessing step. The data after the preprocessing step of the ML pipeline can be exploited in the predictive performance evaluation process. In this case, most of the advantages offered by the embedded representation will be lost and they have to be substituted by a robust

preprocessing process that produces a reliable vectorial representation of the input. An example of preprocessing that can be exploited in this step has been presented in [3, 19, 100]

In the second scenario, the ML pipeline already includes an embedding phase that can be then exploited also in the predictive performance evaluation process. Although the embeddings are extracted from outside the model itself, they can be considered reliable since the predictive model has been trained relying on this information. In this case, the predictive performance evaluation is agnostic about the predictive model, i.e., any typology of model as linear, tree-based, neural networks can be used since the analysis of the predictive performance will consider only the embedding of the input data and the prediction made by the model. In some cases, being model agnostic is a requirement. An example of embedding that can be exploited for example in NLP is Word2Vec [40].

The third and last scenario taken into account is when a Deep Learning Model is used. As in the previous chapters, the inner knowledge of the models can be exploited to get an accurate and valuable embedded representation of an input sample also expressing the knowledge learned by the model. Furthermore, deep learning models are often fine-tuned to address smaller more specific tasks, starting from a more general pre-trained checkpoint. The pre-trained model is able to recognize a wide range of concepts, e.g. the pre-trained BERT model incorporates the knowledge of millions of textual documents. Then, with the fine-tuning the models are specialized to solve a particular task, hiding all the latent knowledge used during the decision-making process. Instead, the embedded representation of the input extracted directly from the inner layers of the network will allow to include also latent knowledge in the unsupervised analysis of the predictive performance. To address this scenario, the techniques proposed in Chapters 4 and 6 can be exploited.

As previously cited, exploiting the embedded version of the input samples will allow preserving the privacy of their content. The embedding is a simplified vectorial representation of complex input data from which it is not possible to reconstruct the original input if the source of the model is not available for retraining [50, 13, 12].

8.2.2 Per-class predictive performance degradation

The knowledge of the predictive model is represented by the distribution of the training samples (historical data with labels). As long as new incoming data maintains a distribution similar to the one learned by the model at training time its performance can be considered unchanged. However, if the incoming data distribution starts to drift the model's predictions may become misleading or erroneous. Traditional evaluation techniques, e.g., f-measure, precision, recall, are obviously not applicable since they require ground-truth labels for the new incoming data, which is missing for newly-classified samples.

Hence, we exploit an unsupervised strategy that, given a set of data samples divided into classes (either ground-truth labels or assigned by the model), is able to quantify both their intra-class cohesion and inter-class separation. To efficiently capture the model performance degradation over time, the proposed approach performs (i) *Base Distribution Evaluation*, by computing unsupervised quality metrics on the training set, and (ii) *Degraded Distribution Evaluation*, by periodically recomputing the same metrics on the newly labeled data and comparing it to the values obtained in (i).

In this work, we exploit the Silhouette [70], a measure of the fit of each sample within its predicted class. It measures how similar a sample is to its own class (cohesion) compared to other classes (separation). The Silhouette value ranges from -1 to +1, with higher values indicating a better match to the assigned class and a poor match to other classes. In order to compute the Silhouette, for each sample, the pairwise distance between itself and each other sample of the dataset has to be calculated. For this reason, the traditional computation of the Silhouette coefficient is not suitable for Big Data scenarios. However, we recently developed a scalable alternative named Descriptor Silhouette (DS) that makes the silhouette suitable for large amounts of data [100]. We included this alternative index in the framework to address big data distributed platforms as well. In practice, if the size of the data requires distributing the computation the DS index is exploited approximating the real silhouette value. Otherwise, the original silhouette definition is exploited.

The degradation of the predictive model is then obtained by evaluating the quality metric before and after the arrival of new training data.

Finally, to fully automate the process, a degradation threshold should be defined to trigger predictive model rebuilding. This threshold depends on the expectations of domain experts and end-users, the risks and costs related to the specific application, and also the number of records and classes of the dataset. Its evaluation is out of the scope of this work.

Descriptor Silhouette index. The Silhouette [70] is a well-known index with the purpose of evaluating the quality of clusters of points in terms of cohesion and separation. However, the computational cost of this index is $O(N^2)$, where N is the cardinality of the dataset (a critical dimension in a Big Data context): to calculate the Silhouette index all the pairwise distances between the points of the dataset have to be computed. Thus, the Silhouette coefficient is not suited to be applied in a Big Data application.

To solve this problem, we propose a new approach, named DS (Descriptor Silhouette), that is tunable to have a computational cost linear in the number of records and with an error with respect to the standard Silhouette score definitely acceptable.

The DS describes the geometrical space defined by a group of points with a low number of descriptors, well distributed. For each group, i.e., classification label, a number of descriptors are extracted through clustering analysis.

Algorithm 2 shows the main steps necessary to compute the DS index. The algorithm takes into input the set of data samples X along with the class labels for each

Algorithm 2: Descriptor Silhouette computation.

Input : Input samples X; class labels y; number of descriptors κ;
Output: Descriptos Silhouette Samples DSS

1 $D \leftarrow \emptyset$;
2 $DSS \leftarrow \emptyset$;
/* Compute Descriptors for each class */
3 **for** c ***in*** *unique(y)* **do**
4 $X_c \leftarrow$ filterSamplesPerClass(X,c);
5 $KM \leftarrow$ fitClustering(X_c, κ);
6 $D_c \leftarrow$ getCentroids(KM);
7 $D \leftarrow D \cup D_c$;
8 **end**
/* Compute Descriptor Silhouette for each Sample */
9 **for** $x \in X$ **do**
10 $a(x) \leftarrow$ computeCoefficient_a(x, D) ; // Equation 8.1
11 $b(x) \leftarrow$ computeCoefficient_b(x, D) Σ ; // Equation 8.2
12 $dss(x) \leftarrow$ computeSilhouette($a(x)$, $b(x)$) ; // Equation 8.3
13 $DSS \leftarrow DSS \cup dss(x)$;
14 **end**
15 **return** DSS

point y and the number of descriptors κ.

The first part of the algorithm computes κ descriptors for each class separately (lines from 3 to 8). The descriptors, in our implementation, correspond to centroids extracted by the well-known K-Means algorithm [7]. Indeed, K-Means and generally clustering algorithms are able to describe the dataset with a limited number of centroids, that well represents the whole geometrical points distribution. Moreover, K-Means can be easily parallelized to scale horizontally within the available computational resources.

Then, the descriptor silhouette is computed for each sample contained in the input data $x \in X$ (lines from 9 to 14). The DS approach implements the Euclidean distance and it exploits a Silhouette definition similar to the original (Equation 8.3) to calculate the score for each point in the dataset.
First, the $a(x)$ coefficient is computed (line 10). It is the average distance of sample x from all the other descriptors $d \in D_x$ of class c (Figure 8.4-left), i.e.,

$$a(x) = \frac{1}{|D_x|} \sum_{d \in D_x, x \neq d} distance(x, d) \tag{8.1}$$

and then $b(x)$ (line 11) that represents the lowest average distance of x from all other classes' descriptors $d \in D_k, k \neq x$ (Figure 8.4-right), i.e.

$$b(x) = \min_{k \neq x} \left(\frac{1}{|D_k|} \sum_{d \in C_k} distance(x, d) \right) \tag{8.2}$$

Finally, the silhouette formulation is applied (line 12):

$$s(x) = \frac{b(x) - a(x)}{max\{a(x), b(x)\}} \tag{8.3}$$

As the Silhouette coefficient, the proposed DS index assumes values in $[-1, 1]$. The DS is computed for each sample in a dataset. It gives information about the quality of the assignment of that record to a specific classification class. Values lower than zero represent a bad assignment of the record to a group, while values higher than zero mean a good assignment. In the specific case of model assessment, the classes of the classifier can be considered as clusters to which the new incoming unlabeled data is assigned.

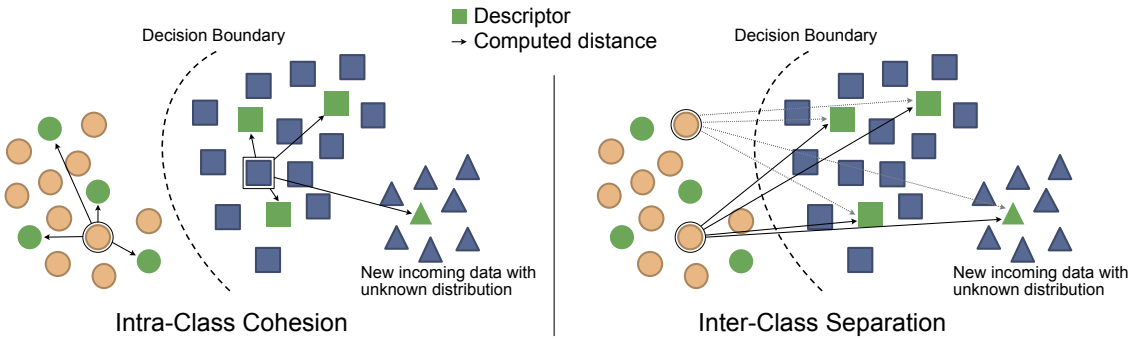


Figure 8.4: Visual example of distance computation among points and descriptors with $\kappa = 4$. intra-class cohesion on the left and inter-class separation on the right.

Increasing the number of descriptors the approximation of the DS will reduce. Theoretically, if $\kappa = |X|$ then the Descriptor Silhouette is equal to the Silhouette index. In practice, the current implementation will use the original definition of silhouette if $|X|$ is small enough to be computed locally or if $\kappa \ll |X|$ is not satisfied. In this way, the number of distances to compute is significantly reduced, computing the distances only between points and descriptors. The computational complexity of this new index is lowered to $O(|X| * C * \kappa)$, where $|X|$ is the cardinality of the dataset, C is the number of classes learned by the classifier, and κ is the number of descriptors computed for each class. The number of classes is usually very small w.r.t. the number of points and, commonly, it remains constant unless a model is retrained with a new set of classes. Also, the number of descriptors is usually several order of magnitude lower than the number of records in data collection under analysis.

Per-class degradation estimation. Figure 8.5 shows the main steps of the evaluation over-time of the predictive model performance degradation. The algorithm analyze both the data exploited at training time, i.e., time t_0, and windows of new incoming data labeled by the model until time t_n (Figure 8.5-(1)). The evaluation of the degradation is triggered as soon as s new incoming samples are collected. When a set of s new incoming samples has been labeled, the evaluate the window of size T of historical samples and

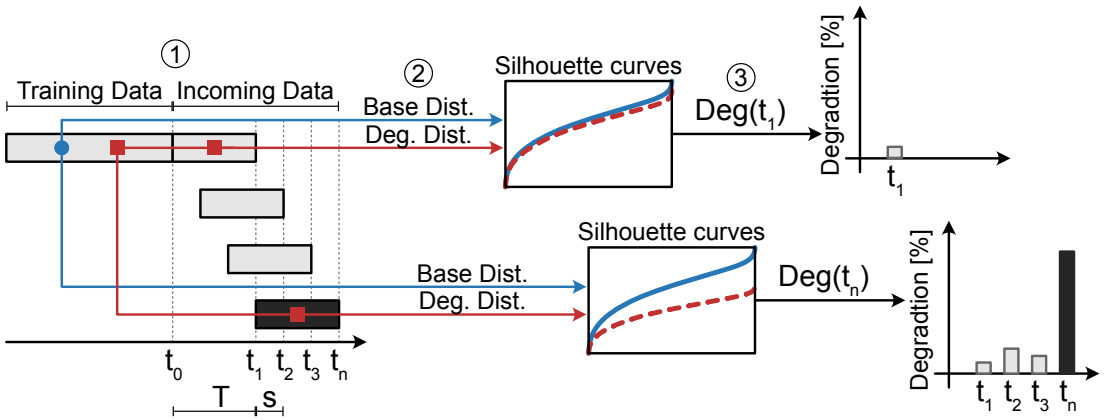


Figure 8.5: Degradation estimation over-time.

it recomputes an updated silhouette curve including both the training samples and the new labeled points. The window of analysis T should never be bigger than the number of samples in the training data. Otherwise, a possible drifting distribution may overwhelm the original one and then the silhouette may indicate this new distribution as cohesive and well separated producing misleading results.

Then, the algorithm compares the cohesion and separation, through the silhouette index, of two different distributions (Figure 8.5-②): (i) the base, computed on the training set only, and (ii) the potentially degraded one, computed on the training set plus the newly-labeled data. Hence, at each step, for each class, a curve is plotted, representing the Silhouette values of each point, sorted in increasing order. When comparing two Silhouette curves, an under-sampling of their points can be required to allow a point-to-point evaluation. An upward shift of the Silhouette curve represents an improvement in terms of intra-class cohesion and inter-class separation, while, on other hand, a downward shift denotes a degradation. This means that the current prediction model is not able to correctly assign the class labels to these new samples. Furthermore, the correct class labels might be new (additional) ones, unknown for the current model.

The degradation of the model's performance is thus quantified separately for each class c and aggregated to give the overall degradation of the model at time t (Figure 8.5-③). The degradation is translated in an evaluation of the errors between the two silhouette curves. The MAAPE (*Mean Arc-tangent Absolute Percentage Error*) [36] has been exploited in our implementation since it is more robust to the presence of values close or equal to zero. Although other error metrics can be considered. Given a predictive model trained on a set of classes C at time t_0 and characterized by a silhouette

curve Sil_{t_0}, the degradation of a class $c \in C$ at time t is described by the following:

$$DEG(c,t) = \alpha * \text{MAAPE}(Sil_{t_0}, Sil_t) * \frac{N_c}{N} \tag{8.4}$$

$$\alpha = \begin{cases} 1 & if : \overline{Sil_{t_0}} \geq \overline{Sil_t} \\ -1 & if : \overline{Sil_{t_0}} < \overline{Sil_t} \end{cases} \tag{8.5}$$

The coefficient α defines if the degradation is positive, meaning a possible reduction in performances of the prediction model, or negative, when the new data fits the training distribution, hence increasing the cohesion of class c. From (8.4), the degradation is modeled as the MAAPE error between the baseline Silhouette Sil_{t_0} on the training set and the possibly-degraded Silhouette Sil_t computed at time t on newly-labeled data. The degradation estimation is weighted by the ratio N_c/N, where N_c is the number of new records assigned to class c and N is the total number of new samples, with N capped at the number of samples in the training set N_{train}. The cap is introduced to allow a fair comparison when $N \gg N_{train}$. To this aim, Sil_t is computed on a trailing window containing the latest N samples up to the number of samples of the training set N_{train}. Finally, the overall degradation of the model at time t is then defined by the following:

$$DEG(t) = \sum_{c \in C} max(0, DEG(c,t)) \tag{8.6}$$

The overall model degradation considers only when the degradation of each class is positive, thus affecting its performance. In other words, as the new incoming data in increasing the cohesion of the class-of-interest, it is not considered as concept-drift.

The proposed degradation evaluation, other than defining when the predictions of a model are no more reliable, can be exploited to decide when to rebuild the model itself. This decision can be taken considering a threshold over which the newly collected data is considered as drifting. However, the retraining process of the model, given its degradation trend, is out of the scope of this work.

8.3 Experimental results

Datasets. We present experimental results on concept drift management on four datasets. Table 8.1 summarize the main characteristics of the dataset exploited in this experimental section. The concept shifts have been manually introduced following two different patterns and it is represented by the arrival of data belonging to previously unseen classes, unknown to the predictive model.

The first dataset, D1, is a synthetic dataset created with the *scikit-learn* Python library [62]. It has been generated with 4 normally distributed classes and 800,000 records, 200,000 for each class, and 10 features. D1 has been built to assess the proposed algorithm. Indeed, each class in the dataset is characterized by a high cohesion

Table 8.1: Summary of datasets exploited in concept drift management experiments.

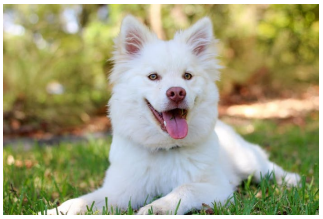
Id	Data Type	Model	# Classes (train + drift)	# Record	Drift Pattern
D1	Synthetic	Random Forest	4 (2+2)	800,000	P1
D2	Texts	Doc2Vec + Random Forest	3 (2+1)	3,000	P1
D3	Images	VGG16 (Fine Tuning)	3 (2+1)	1,000	P2
D4	Texts	BERT (Fine Tuning)	4 (3+1)	100,000	P2

and inter-class separation. This allows understanding if and how the predictive performance degradation is identified by our algorithm.

Dataset D2, on the contrary, is a real-world dataset containing Wikipedia articles, extracted from 3 classes: *mathematics*, *literature*, and *food-drink*. For each class, a selection of 1,000 articles appearing in the Wikipedia index for that class has been downloaded. Each article has been pre-processed to obtain an embedded representation of 100 features for it. The document-embedding process takes advantage of a Doc2Vec model [40], pre-trained on the English Wikipedia corpus [39]. For D1 and D2, a Random Forest classifier has been used as a predictive model. In both cases, the training set consisted of a stratified sample over classes 0 and 1, sampling 60% of the records in each class. The remaining part of the dataset (i.e., 40% class 0, 40% class 1, the whole unknown class 2 and/or 3 according to the dataset) is exploited as the test set to assess model degradation. Using a 3-fold cross-validation, the average f-measure of the predictive model is 0.964 for dataset D1, and 0.934 for dataset D2.

Dataset D3 is composed of images. A subsample of three classes, available in the Multi-domain Image Characteristics Dataset[1], has been exploited. The three classes are about animals and are divided into three categories: *Cat*, *Dogs*, and *Fox*. Figure 8.6 shows a sample of the images extracted from the dataset. Dataset D3 contains 1,000 images, of which about 280 equally distributed among *Cat* and *Dog* are exploited to fine-tune a VGG16 model (pre-trained on the Imagenet dataset). The fine-tuned model reached an accuracy of 0.99 on the binary task. The remaining samples are exploited to simulate the presence of concept drift, using class *Fox* as drifting data.

(a) Example for class *Cat* included in training.



(b) Example for class *Dog* included in training.

(c) Example for class *Fox* not included in training.

Figure 8.6: Data samples extracted from dataset D3.

[1] https://www.kaggle.com/realtimear/multidomain-image-characteristics-dataset

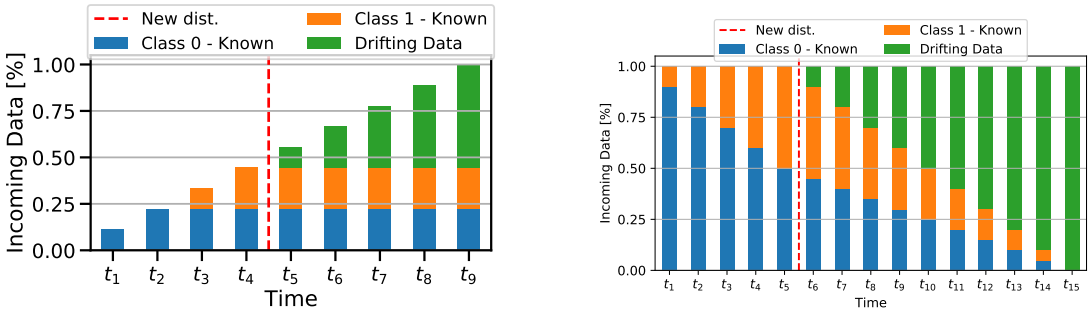
Dataset D4 is composed of about 100,000 documents divided into 4 classes. The classes available in this dataset are *world*, *sport*, *business*, and *Sci/Tech*. A subsample of 74612 rows, belonging to classes *world*, *sport*, and *business*, has been exploited to fine-tune a base uncased BERT model composed of 12 hidden layers. The model reached an accuracy of 0.97 in the validation process. All the remaining records have been used to test the performance of the concept-drift detection algorithm using class *Sci/Tech* as drifting data.

Patterns of analysis. Two different patterns of incoming concept drift have been evaluated. They allow showing the capability of the proposed concept drift evaluation strategy to correctly assess the predictive performance degradation over time when a new unknown data distribution is introduced. Figure 8.7 shows the two different patterns.

Pattern P1 simulates a window of analysis of increasing size (Figure 8.7a). The first four test sets, from t_1 to t_4, contain only data known to belong to the classes included in the training set. Then, from t_5 to t_n, a new drifting distribution is gradually introduced. The test sets have been designed to simulate the flow of time. This pattern has been applied to datasets D1 and D2. The window of analysis is incrementally added over time: t_1 contains half of the test set for class 0, t_4 contains the whole test set for classes 0 and 1, t_5 contains all the test samples for classes 0 and 1 and 20% of the unknown class (class 2 or 3), then at each step another 20% of the unknown class is added for the evaluation of degradation until t_n.

Pattern P2 simulates a sliding window of analysis with fixed size and a drifting distribution of the analyzed samples (Figure 8.7b). In this case, the window size remains fixed to T and the evaluation of the predictive performances is triggered as soon as new s samples are collected. The simulated flow of incoming data, even in this case, consists of samples belonging to known classes (but never exploited in the training of the model) from time t_1 to t_5. Then, from time t_6 to t_n a new data distribution is increasingly introduced, while previously collected data is consequently forgot. This pattern has been applied to datasets D3 and D4.

Experimental definition of degradation threshold. Defining when the degradation is correctly detected is a challenge in real scenarios. We defined an experimental way to decide whether the measured degradation is due to concept drift or if it is caused by the noisy nature of the data under analysis. In the firsts periods of time, after the deployment of the model, it is reasonable to assume that the distribution of the data is similar to the one collected and labeled at training time. Thanks to this assumption, the degradation of the first n periods of time t can be observed. These observations can be used to determine the $t - distribution$ of the degradations observed under non-drifting conditions. The $t - distribution$ with $n - 1$ degrees of freedom is used since n can be assumed to be lower than 30 (as defined in the literature). Then, each new observation



(a) Pattern P1

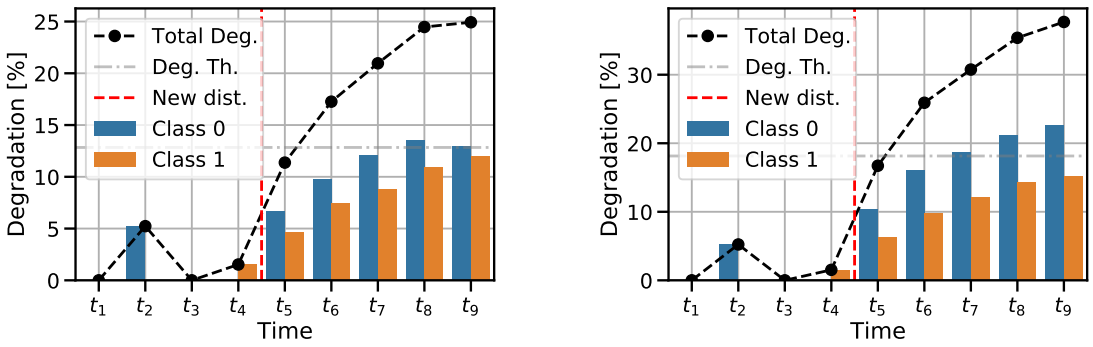
(b) Pattern P2

Figure 8.7: Concept Drift patterns.

after t_n is tested and if their p-value is lower than $\alpha = 5\%$ the null hypothesis is rejected. This means that the degradation measured after t_n is significantly higher w.r.t. the non-degrading observations.

Validation on degradation pattern 1. Figure 8.8 shows the percentage of degradation measured on pattern P1 for dataset D1. The synthetic distribution of the classes allows identifying exactly when the concept drift occurs and how much it is affecting the model performance. In this specific case, data embedding has not been required since the data is already numeric and it does not require any preprocessing being synthetically generated. Thus, the raw samples have been given as input to the algorithm. Also, the Descriptor Silhouette has been configured to extract 200 descriptors per-class. The presence of concept drift is correctly highlighted when both unknown classes 2 and 3 are introduced as drifting concepts (Figures 8.8a and 8.8b respectively). Up to time t_4 an average degradation of 1.7% is measured, while as soon as drifting records occur, i.e., time t_5, the degradation increases over 10% with Class 2 as drifting concept and reaches 16.7% when Class 3 is exploited as a drifting concept. Then, the degradation percentage increases almost linearly until time t_9, reflecting the trend of introduced drift showed in pattern P1 (each timestamp, from t_5, adds 40,000 records with concept drift). Moreover, Figure 8.9 shows in detail the DS curves before and after the arrival of degraded data. The Base and Degraded curves at time t_1 are almost coincident (Figure 8.9a). Instead, at time t_9, the degradation is evident, showing Degraded curves downshifted w.r.t. the Base ones. These results show that the approach is theoretically valid. However, real data is rarely well distributed in a geometrical space.

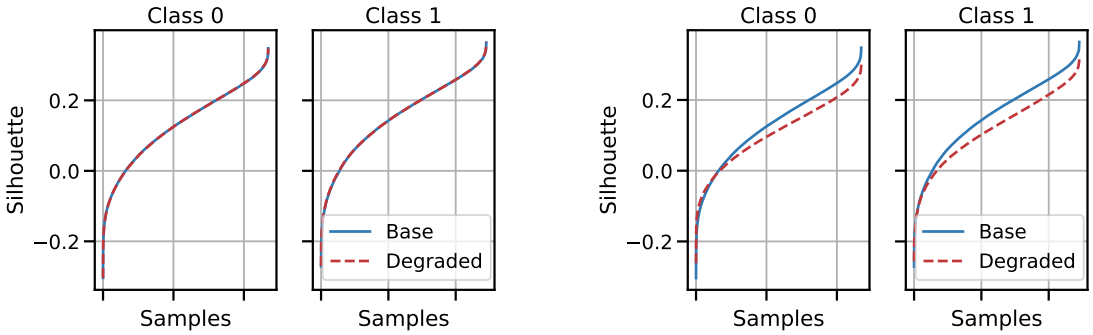
Dataset D2 is exploited to assess the proposed approach in the case of real and high dimensional data, exploiting pattern P1. Due to the sparse nature of textual data, dataset D2 has been preprocessed exploiting a Doc2Vec model. The output of this preprocessing is still high dimensional, i.e., each input document is represented by a vectorial



(a) Degradation with class 2.

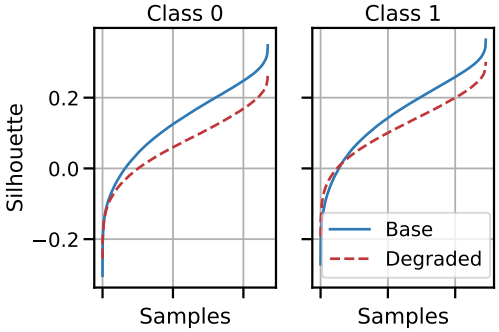
(b) Degradation with class 3.

Figure 8.8: Dataset D1. Model degradation over time, with training on classes 0 and 1.



(a) Base compared to time t_1 with no degradation

(b) Base compared to time t_9 with class 2 degradation



(c) Base compared to time t_9 with class 3 degradation

Figure 8.9: Comparison of Base DS curve at training time, and degraded DS curves at time t for dataset D1.

embedding of size 300 dimensions. However, thanks to the property of textual embedding, the documents have been projected on a much lower better separated geometrical space concerning the original one. Thus, the documents' embeddings have been

exploited as input to the algorithm. Figure 8.10 shows the degradation of the Wikipedia document classification model when the unknown Class *mathematics* is introduced. Between times t_1–t_4, the new unseen data classified by the model fit well the learned distribution, with an overall degradation below 15%. At time t_5, samples of an unknown class start arriving and the overall degradation raises above 28% (each timestamp, from t_5, adds 200 records with concept drift). Also, the class mostly affected by degradation over-time is Class 0 because most of the documents belonging to the unknown *mathematics* class have been wrongly assigned by the model to Class *food-drink*. Thanks to our unsupervised algorithm it is possible to highlight the presence of drifting data and to have evidence of which is the most affected class.

In both datasets, the analysis of the overall degradation of the known-classes has been correctly detected when the shift was introduced. Even if the proportion of new shifted data was low (i.e., 20% of the unknown class). Specifically, we note that the overall degradation at least doubled from the pre-drift (t_1 to t_4) to the post-drift (t_5 to t_n), hence proving to be a promising detector of these events.

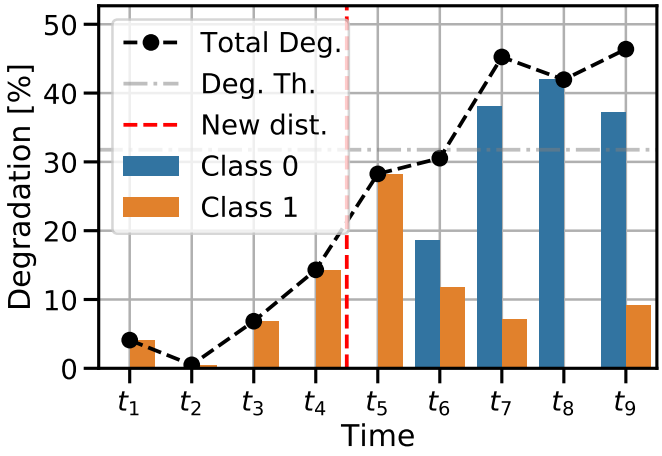
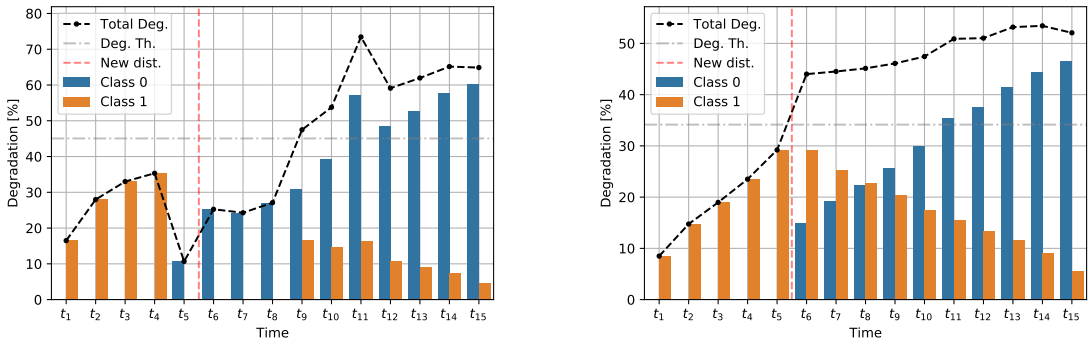


Figure 8.10: Dataset D2. Model degradation over-time, with training on classes 0 (*food-drink*) and 1 (*literature*). Degradation introduced with the new class 2 (*mathematics*).

Validation on degradation pattern 2. Dataset D3 shows an even harder domain of analysis. The dimensionality of image data is very high and deep learning models like VGG16 are exploited to deal with it. The tested predictive model is the fine-tuned VGG16 model. As already explained in Chapter 4, valuable embedding representation for images can be extracted directly from its hidden layers. For this analysis, only the last convolutional tensor of the VGG16 model has been exploited as an embedded representation of the input. The last convolutional layer of the VGG16 is composed of a tensor of shape $512x14x14$. Thus, each input image is converted into a vector of 100,352

dimensions. This vectorial representation is very sparse and with very high dimensionality but it offers a good test-bench to measure the robustness of our approach. Drifting concepts are introduced following pattern P2. Also, the Descriptor Silhouette has been configured to extract a number of descriptors equal to the 10% of the samples in the window of analysis. The window of analysis has max size $T = 100$ and step $s = 10$ samples. The small window size allows the computation of the standard silhouette index to show the differences between the two results. Figure 8.11a shows the measurement of degradation over-time exploiting the Descriptor Silhouette. The high dimensionality of the data affects the quantification of the predictive performance quality. The degradation is far from zero when known distributions occur, i.e., time t_1-t_5. This is mainly caused by three factors: (i) the intrinsic variability characterizing highly dimensional data, and (ii) the approximation introduced when computing the Descriptor Silhouette, and (iii) the known limits of the euclidean distance measure in the case of very sparse data. Despite these limitations, however, a significant increment in the degradation is measured just with a delay of three timestamps. The drifting data has been starting from t_6, while the increment is highlighted at time t_9. So, the drift is recognized when 40% of the window is composed of drifting samples. Also, the most affected class is *Cat*. The VGG16 is wrongly labeling most of the *Fox* images as *Cat*, thus the degradation of class 0 is significantly higher than the one of class 1. In the next paragraphs, the limitations of this approach will be described in detail.



(a) Degradation measured with Descriptor Silhouette.

(b) Degradation measured with Standard Shilouette.

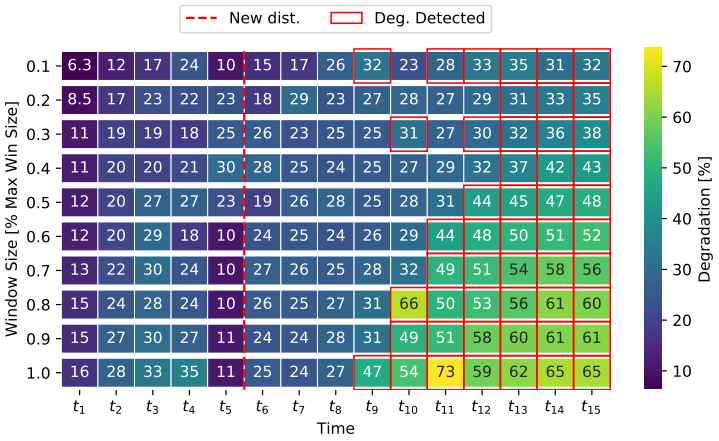
Figure 8.11: Dataset D3. Model degradation over-time with training on classes 0 (*Cat*) and 1 (*Dog*). Degradation introduced with the new class 2 (*Fox*).)

Limits of concept drift detection with Descriptor Silhouette. The small window size exploited for dataset D3 allows the computation of the standard silhouette index to show the differences among the two results. Figure 8.11 shows the measurement of degradation over-time exploiting Descriptor Silhouette (Figure 8.11a) and the standard silhouette index (Figure 8.11b). The already cited causes of error affecting the proposed

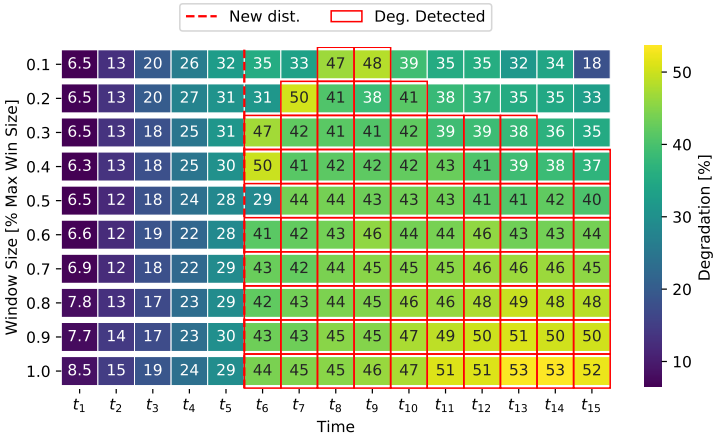
methodology are: (i) the intrinsic variability characterizing highly dimensional data, and (ii) the approximation introduced when computing the Descriptor Silhouette, and (iii) the known limits of the euclidean distance measure in case of very sparse data. In both cases, the overall degradation measurements before the introduction of drifting data, i.e., $t \leq t_5$, show values significantly higher than zero. The two indexes behave similarly, and both are affected by the dimensionality and the sparsity of the input data. However, differently than DS, the standard silhouette shows a significant increment of degradation as soon as the drifting data is introduced at time t_6. In other words, with just 10 drifting samples, over a window of size 100, the concept drift is detected. In practice, the approximations made with the DS can be one of the causes of delayed drift detection when very high dimensional data has to be analyzed. However, while the number of the collected samples increases the Standard Silhouette becomes even more costly in terms of computational resources, not being applicable anymore. While the Descriptor Silhouette remains scalable and its performances are acceptable if the dimensionality of the data remains in mid-low ranges.

Effects of the size of the window of analysis. Another parameter that is affecting the efficiency of our concept drift detection approach is the size of the window of analysis T. To measure its effects we perform the experiment in Figure 8.12. Each row of the matrix represents the over-time measurement of degradation with different window sizes. The window size of the first row is 10% of the Max Window Size and it increases by 10%. The step of analysis s for each window corresponds in turn to the 10% of the current window size. The Max Window Size is set to $T = 100$ for dataset D3. From the experiments, it is clear that the size of the window plays a key role in the algorithm performance. As the window size increases, the concept drift is detected with more precision. The solution with the Descriptor Silhouette (Figure 8.12a) is clearly showing the degradation trend only when the window size reaches 50% of its maximum. However, the drift is detected with a large delay of 6 timestamps. Then, as the size of the window increases, the detection is anticipated up to time t_9. So, a large window would be more effective in the detection of the drift. Of course, its size should not exceed the size of the data used to compute the base distribution. On the other side, the experiment performed with the Standard Silhouette (Figure 8.12b) is showing good performances even with very small window sizes. The drift however is detected on a non-continuous basis if the window is too small.

Effects of the number of descriptors. The number of descriptors to use in the proposed framework is a critical parameter that needs to be properly tuned. From a theoretical point of view, for dense data distributions, the number of descriptors can be very small. Instead, the more the distribution of data is sparse, the more it becomes hard to completely describe the problem with few descriptors. To overcome this problem it is necessary to increase the number of descriptors used by the Descriptor Silhouette.



(a) Degradation measured with Descriptor Silhouette.

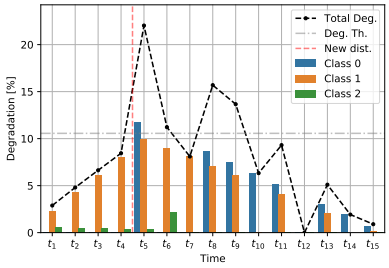


(b) Degradation measured with Standard Shilouette.

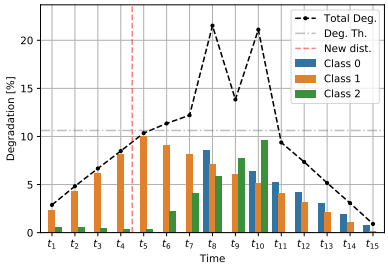
Figure 8.12: Model degradation over-time measured for dataset D3 varying the size of the window of analysis. Max window size $T = 100$ and step $s = 10$. Training on classes 0 (*Cat*) and 1 (*Dog*. Degradation introduced with the new class 2 (*Fox*).)

To show this effect we exploited dataset D4 which is composed of textual data analyzed with the BERT model. It is commonly known that textual data is usually characterized by very sparse data distribution. Then, we considered a complex task of topic detection for which BERT reached 0.97 of accuracy on the validation set. We tested our framework by introducing a new class as drift following pattern 2 in Figure 8.7b. The embeddings of the input are extracted from the last transformer layer, following the extraction process described in Chapter 6. We repeated the experiment by increasing the number of maximum descriptors extracted by DS, i.e. 100, 500, and 2000. Figure 8.13 shows the results of this experiment. As expected, with only 100 descriptors we are

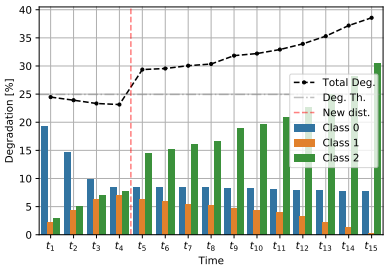
partially able to detect the drift introduced from time t_5, but the trend is very unstable. The situation slightly improves when increasing the maximum descriptors to 500, but still, the most drifting timestamps are wrongly identified as non-drifting. We have to further increase the number of descriptors to a max of 2000 to get the correct result. Basically, with this configuration, we are computing the minimum approximation possible renouncing to computational time. Still, we confirm that our approach can be configured to address very sparse data distributions. Deciding the correct configuration though is still an experimental task that has to be addressed before putting the system in production by analyzing the complexity of the user's data.



(a) Degradation measured with max 100 descriptors.



(b) Degradation measured with max 500 descriptors.



(c) Degradation measured with max 2000 descriptors.

Figure 8.13: Dataset D4. Model degradation over-time with training on classes 0 (World), 1 (Sports), and 2 (Business). Degradation introduced with the new class 3 (Science). The window of analysis is equal to 2000 samples.)

8.3.1 Evaluation of Descriptor Silhouette performance

Descriptor Silhouette scalability. The proposed approach for the estimation of predictive model degradation is based on the computation of the Silhouette index proposed in [100]. To take advantage of its scalability properties, the algorithm has been developed exploiting the *Map-Reduce* programming paradigm with the Apache Spark framework [106]. The Silhouette computation is the most onerous component in the concept

drift detection framework, hence extensive experiments are required to show its performance and scalability. We test its performance in a distributed configuration, showing its effectiveness in presence of very large amounts of data.

A synthetic dataset of 10,000,000 (10M) records with 10 features and a normal distribution over 3 classes has been created exploiting the *scikit-learn* Python machine learning library [62]. From this dataset, 6 sub-datasets with the same characteristics of the original one but with different cardinality have been extracted: d_1) 10K, d_2) 50K, d_3) 100K, d_4) 500K, d_5) 1M, and d_6) 10M records. The first 5 datasets have been used to test the single-node configuration, exploiting an Intel i7 8-core server with 32GB of memory, while the distributed configuration consists of 50 virtual nodes, each with 512MB of memory and 2 cores, running on top of the *BigData@Polito cluster*[2]. All experiments have been configured to compute 200 descriptors for each class, showing the tunable linearity in the single-node configuration and making the experiments comparable in the distributed configuration.

Figure 8.14 shows the results on single-node configuration. It reports the time the execution time for DS index (DS) compared to one required by the standard Silhouette available in *scikit-learn*[3] (SKS). DS presents linear scalability over the number of records in the dataset. Furthermore, the D curve shows the negligible constant time required to compute the descriptors.

The execution time of DS for dataset d_5 is much lower than the time required to compute the SKS in the single-node configuration, but some applications dealing with a stream of data might require even stricter timing constraints that can be reached with a distributed configuration.

Thus, we tested the approach in a distributed environment showing that the processing time of DS can be as low as few minutes (e.g., 3 minutes) for dataset d_5 (1M records) and the order of tens of minutes for dataset d_6 (10M records). In the Spark cluster, the data is split into partitions (slices of data) and redistributed in the executor nodes, enabling parallel computation and minimizing network traffic for sending data between executors. Figure 8.15a shows the time required to compute DS as a function of the number of partitions in which the dataset is redistributed in the cluster.

As already discussed, DS requires only 194 seconds for dataset d_5 (1 million rows) and 1519 seconds (approximately 25 minutes) for dataset d_6 (10 million rows) to be computed when data is redistributed in 500 partitions. Splitting the data into multiple partitions improves the parallelization of the algorithm even if the number of available workers is kept constant since the resource manager of the cluster assigns pending jobs to workers as soon as they have free CPU cores. This is why, in our cluster with 50 nodes, the computational cost speedup for dataset d_5 is almost linear up to 100 partitions. Figure 8.15b shows the speedup obtained running DS in the distributed configuration

[2]https://smartdata.polito.it/computing-facilities/

[3]https://scikit-learn.org/stable/

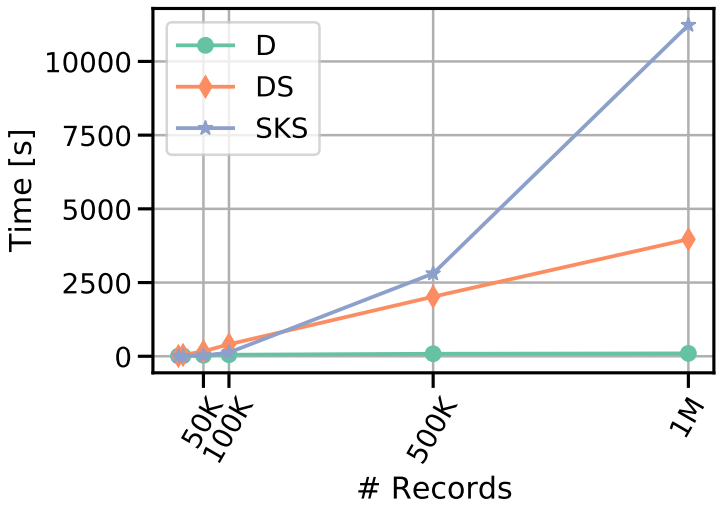
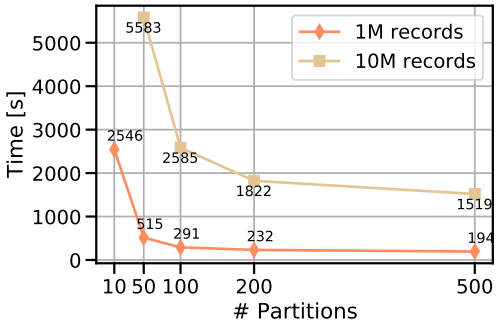


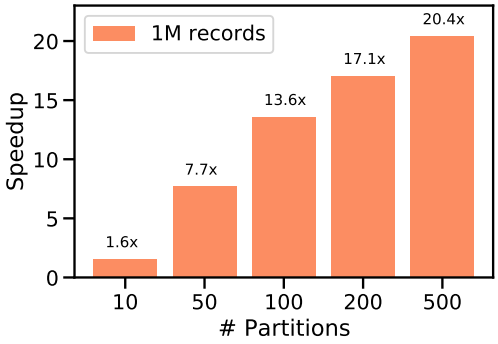
Figure 8.14: Scikit-learn Silhouette and DS cost comparison on a single-node. D is the time required to compute the descriptors, DS is the time required to obtain the DS values and SKS is the time required for the SciKit-provided Silhouette, for datasets from d_1 to d_5.

w.r.t. the single-node configuration for dataset d_5.

Exploiting a powerful single server, the execution time reaches approximately 3968 seconds while distributing the jobs on many small workers allows having an improvement of more than 20 times, thus enabling the DS to be calculated in near real-time even in presence of huge amounts of data.



(a) DS cost in distributed configuration for dataset d_5 with (1 million records) and d_6 (10 million records).

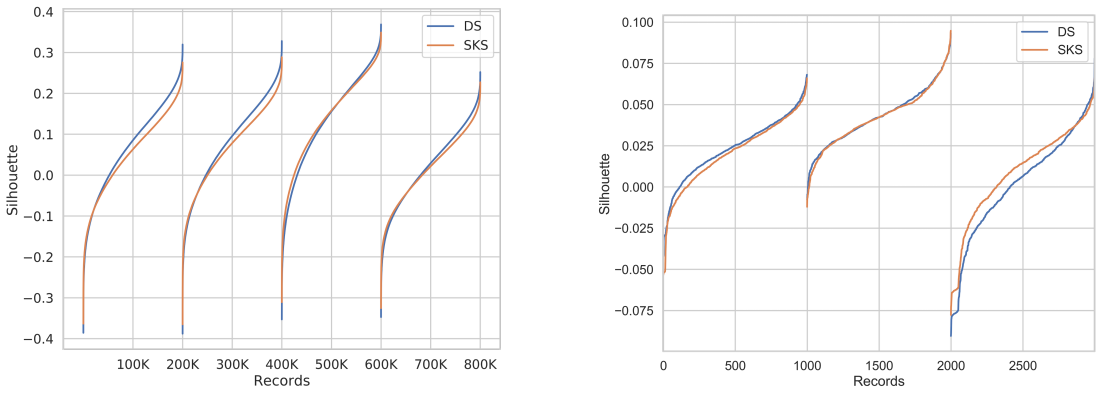


(b) DS speedup in a distributed configuration w.r.t. single-node execution for dataset d_5 with 1 million records.

Figure 8.15: DS scalability performance.

Descriptor Silhouette performance. As already shown in [100], the DS algorithm better approximates the standard Silhouette w.r.t. other scalable approaches (i.e. the Square Euclidean Silhouette implemented in Apache Spark MLLIB [54]).

To prove the accuracy of DS also for datasets $D1$ and $D2$ we show in Figure 8.16 a



(a) Synthetic dataset D1

(b) Wikipedia dataset D2

Figure 8.16: Silhouette curve comparison computed with DS configured with 200 descriptors and SKS.

Table 8.2: Silhouette scores comparison between DS, SKS and Squared Euclidean Silhouette (SES).

	DS_{avg}	SKS_{avg}	SES
D1	0.069	0.077	0.109
D2	0.024	0.021	0.044

qualitative comparison of the Silhouette curves calculated with the standard SKS and with DS configured with 200 descriptors. Each couple of curves represents the Silhouette of each class, calculated with DS and SKS. It is noticeable that the DS and SKS curves for each class are very close to each other, meaning that DS correctly models the more complex SKS ground-truth value. Even in the case of textual data (dataset $D2$ in Figure 8.16b), usually characterized by a high number of features and high heterogeneity, the DS curves are very similar to the ones obtained with the state-of-the-art Silhouette. Table 8.2 shows the average values of the scores obtained with DS, SKS, and the value of SES (Apache Spark Squared Euclidean Silhouette). DS and SKS maximum difference is 0.008 in D1 and only 0.003 in D2. The SES index, instead, is always much further from the other two indexes showing, in case of D2, a score that is more than double w.r.t. the state-of-the-art and almost twice w.r.t. our algorithm.

Chapter 9

Conclusions

In this work, we addressed the problem of managing the reliability of complex Deep Neural Networks at the level of local predictions, per-class model-global behavior, and measuring over-time prediction performance. To do so we propose different strategies that exploit the inner knowledge of the predictive model to ensure the reliability of the proposed explanations. The prediction local and model global explainability has been addressed by introducing a new explanation framework tailored to Deep Neural Networks. We prosed new unsupervised mining strategies to extract and analyze the inner knowledge of different kinds of models employed in different contexts, i.e. image-processing and natural language machine learning tasks. The proposed explanation framework has been developed to adapt to as many alternative models as possible going from Convolutional Networks to complex Natural Language Models like BERT. We demonstrated the high flexibility of the framework with several experimental results. Also, we validated our results both quantitatively and qualitatively through detailed analyses of the obtained explanations. Then, we compare with the currently available state-of-the-art, i.e., LIME [69], GRAD-CAM [76], and Shapley Values [85], showing the higher effectiveness and interpretability that we are able to achieve. Finally, we collected feedback from human users who turned out to prefer our explanations in more than 75% of the cases w.r.t. the other solutions. The main contribution of this first framework can be summarized in:

- The introduction of a flexible solution that can be adapted to a large number of state-of-the-art DNNs and that can exploit their inner knowledge.
- A higher interpretability of the visual explanations better highlighting the portion of the input influencing the prediction process and more detailed numerical quantification w.r.t. the state-of-the-art.
- The introduction of an unsupervised mining strategy of multiple inner layers of deep neural networks model.

- The effective and efficient identification of specific influential features in both visual and the textual domain.
- A qualitative human validation of the approach demonstrates that our solution, in visual tasks is the most interpretable and it is preferred by the vast majority of the interviewed.

The assessment of models' predictive performance over-time has been addressed instead by developing a new scalable framework. In our solution, we proposed a new unsupervised strategy that measures the drift of newly incoming data concerning the knowledge of the model at the training time. We introduced a new definition of per-class model degradation that is based on a new scalable version of the Silhouette called Descriptor Silhouette. The framework has been developed not to be tailored to any specific use-case and model. Indeed, exploiting the inner model knowledge extraction strategies proposed in the explanation framework we are able to apply this solution to complex models like DCNNs and BERT as well as Random Forests. We first demonstrate the effectiveness of our methodology by analyzing several use-cases and showing that is able to identify the presence of drift already when just 10% of data is drifting. Furthermore, we show the scalability property of the proposed Descriptor Silhouettecompared to the standard definition and the lower approximation error w.r.t. the Square Euclidean Silhouette implemented in Apache Spark. The main contribution of this last framework can be summarized in:

- The introduction of an unsupervised approach that measures the predictive performance of the predictive models deployed in production systems without having any knowledge regarding the ground-truth of the analyzed data.
- The introduction of the new interpretable definition of per-class model degradation over-time.
- The introduction of a highly scalable and well-approximated variant of the Silhouette index.
- The extensive validation on different use-cases and models with different architectural structures demonstrates the generality of the proposed solution.

Bibliography

[1] A. Adadi and M. Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052.

[2] David Alvarez-Melis and Tommi S Jaakkola. "A causal framework for explaining the predictions of black-box sequence-to-sequence models". In: *arXiv preprint arXiv:1707.01943* (2017).

[3] Daniele Apiletti et al. "iSTEP, an integrated Self-Tuning Engine for Predictive maintenance in Industry 4.0". In: *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE. 2018, pp. 924–931.

[4] Daniele Apiletti et al. "Selina: a self-learning insightful network analyzer". In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 696–710.

[5] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.

[6] Katie Atkinson, Trevor Bench-Capon, and Danushka Bollegala. "Explanation in AI and law: Past, present and future". In: *Artificial Intelligence* (2020), p. 103387.

[7] Bahman Bahmani et al. "Scalable K-means++". In: *Proc. VLDB Endow.* 5.7 (Mar. 2012). ISSN: 2150-8097. DOI: 10.14778/2180912.2180915.

[8] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.50.

[9] Albert Bifet and Ricard Gavalda. "Learning from time-changing data with adaptive windowing". In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pp. 443–448.

[10] Daniel Borkan et al. "Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification". In: *CoRR* abs/1903.04561 (2019). arXiv: 1903.04561.

[11] Arif Budiman, Mohamad Ivan Fanany, and Chan Basaruddin. "Adaptive Convolutional ELM For Concept Drift Handling in Online Stream Data". In: *CoRR* abs/1610.02348 (2016). arXiv: 1610.02348.

[12] Nicholas Carlini et al. *Extracting Training Data from Large Language Models*. 2020. arXiv: 2012.07805 [cs.CR].

[13] Nicholas Carlini et al. *The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks*. 2019. arXiv: 1802.08232 [cs.LG].

[14] T. Cerquitelli et al. "Clustering-Based Assessment of Residential Consumers from Hourly-Metered Data". In: *2018 International Conference on Smart Energy Systems and Technologies (SEST)*. Sept. 2018, pp. 1–6. DOI: 10.1109/SEST.2018.8495863.

[15] T. Cerquitelli et al. "Discovering electricity consumption over time for residential consumers through cluster analysis". In: *2018 International Conference on Development and Application Systems (DAS)*. May 2018, pp. 164–169. DOI: 10.1109/DAAS.2018.8396090.

[16] T. Cerquitelli et al. "Enabling predictive analytics for smart manufacturing through an IIoT platform". In: *IFAC-PapersOnLine* 53.3 (2020). 4th IFAC Workshop on Advanced Maintenance Engineering, Services and Technologies - AMEST 2020, pp. 179–184. ISSN: 2405-8963.

[17] Tania Cerquitelli et al. *Automating concept-drift detection by self-evaluating predictive model degradation*. 2019. arXiv: 1907.08120 [cs.LG].

[18] Tania Cerquitelli et al. "Enhancing manufacturing intelligence through an unsupervised data-driven methodology for cyclic industrial processes". In: *Expert Systems with Applications* ((under review)).

[19] Tania Cerquitelli et al. "Towards a Real-Time Unsupervised Estimation of Predictive Model Degradation". In: *Proceedings of Real-Time Business Intelligence and Analytics*. BIRTE 2019. Los Angeles, CA, USA: Association for Computing Machinery, 2019. ISBN: 9781450376600. DOI: 10.1145/3350489.3350494. URL: https://doi.org/10.1145/3350489.3350494.

[20] François Chollet et al. *Keras*. https://keras.io. 2015.

[21] Marcelo Corrales, Mark Fenwick, and Nikolaus Forgó. *Robotics, AI and the Future of Law*. Springer, 2018.

[22] A. Datta, S. Sen, and Y. Zick. "Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems". In: *2016 IEEE Symposium on Security and Privacy (SP)*. May 2016, pp. 598–617. DOI: 10.1109/SP.2016.42.

[23] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805.

[24] Evelina Di Corso, Tania Cerquitelli, and Francesco Ventura. "Self-tuning techniques for large scale cluster analysis on textual data collections". In: *Proceedings of the Symposium on Applied Computing*. ACM. 2017, pp. 771–776.

[25] Kawin Ethayarajh. "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings". In: *ArXiv* abs/1909.00512 (2019).

[26] "Explaining classifier decisions linguistically for stimulating and improving operators labeling behavior". In: *Information Sciences* 420 (2017), pp. 16–36. ISSN: 0020-0255.

[27] Ruth C. Fong and Andrea Vedaldi. "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). DOI: 10.1109/iccv.2017.371. URL: http://dx.doi.org/10.1109/ICCV.2017.371.

[28] João Gama et al. "A survey on concept drift adaptation". In: *ACM computing surveys (CSUR)* 46.4 (2014), p. 44.

[29] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[30] Philipp M Grulich et al. "Scalable Detection of Concept Drifts on Data Streams with Parallel Adaptive Windowing." In: *EDBT*. 2018, pp. 477–480.

[31] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018), 93:1–93:42. ISSN: 0360-0300. DOI: 10.1145/3236009. URL: http://doi.acm.org/10.1145/3236009.

[32] David Gunning and David Aha. "DARPA's explainable artificial intelligence (XAI) program". In: *AI Magazine* 40.2 (2019), pp. 44–58.

[33] Guodong Guo and Na Zhang. "A survey on deep learning based face recognition". In: *Computer vision and image understanding* 189 (2019), p. 102805.

[34] Bharath Hariharan et al. "Hypercolumns for object segmentation and fine-grained localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 447–456.

[35] MD Zakir Hossain et al. "A comprehensive survey of deep learning for image captioning". In: *ACM Computing Surveys (CsUR)* 51.6 (2019), pp. 1–36.

[36] Sungil Kim and Heeyoung Kim. "A new metric of absolute percentage error for intermittent demand forecasts". In: *International Journal of Forecasting* 32.3 (2016), pp. 669–679.

[37] Ralf Klinkenberg and Thorsten Joachims. "Detecting Concept Drift with Support Vector Machines." In: *ICML*. 2000, pp. 487–494.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[39] Jey Han Lau and Timothy Baldwin. "An empirical evaluation of doc2vec with practical insights into document embedding generation". In: *arXiv preprint arXiv:1607.05368* (2016).

[40] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. 2014, pp. 1188–1196.

[41] Tao Lei, Regina Barzilay, and Tommi Jaakkola. *Rationalizing Neural Predictions*. 2016. arXiv: 1606.04155 [cs.CL].

[42] Bruno Lepri et al. "The Tyranny of Data? The Bright and Dark Sides of Data-Driven Decision-Making for Social Good". In: *Transparent Data Mining for Big and Small Data*. Ed. by Tania Cerquitelli, Daniele Quercia, and Frank Pasquale. Cham: Springer International Publishing, 2017, pp. 3–24. DOI: 10.1007/978-3-319-54024-5_1.

[43] Li Fei-Fei, R. Fergus, and P. Perona. "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 2004, pp. 178–178.

[44] Jiwei Li, Will Monroe, and Dan Jurafsky. *Understanding Neural Networks through Representation Erasure*. 2016. arXiv: 1612.08220 [cs.CL].

[45] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[46] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 1557-9654. DOI: 10.1109/TIT.1982.1056489.

[47] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[48] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4765–4774.

[49] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[50] Aravindh Mahendran and Andrea Vedaldi. "Understanding Deep Image Representations by Inverting Them". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[51] Aravindh Mahendran and Andrea Vedaldi. "Visualizing Deep Convolutional Neural Networks Using Natural Pre-images". In: *International Journal of Computer Vision* 120.3 (May 2016), pp. 233–255. ISSN: 1573-1405. DOI: 10.1007/s11263-016-0911-8. URL: http://dx.doi.org/10.1007/s11263-016-0911-8.

[52] David Martens and Foster Provost. "Explaining Data-Driven Document Classifications". In: *MIS Q.* 38.1 (Mar. 2014), pp. 73–100. ISSN: 0276-7783. DOI: 10.25300/MISQ/2014/38.1.04. URL: https://doi.org/10.25300/MISQ/2014/38.1.04.

[53] Sherin Mary Mathews. "Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review". In: *Intelligent Computing*. Ed. by Kohei Arai, Rahul Bhatia, and Supriya Kapoor. Cham: Springer International Publishing, 2019, pp. 1269–1292. ISBN: 978-3-030-22868-2.

[54] Xiangrui Meng et al. "MLlib: Machine Learning in Apache Spark". In: *J. Mach. Learn. Res.* 17.1 (Jan. 2016). ISSN: 1532-4435.

[55] Jose G Moreno-Torres et al. "A unifying view on dataset shift in classification". In: *Pattern Recognition* 45.1 (2012), pp. 521–530.

[56] W. James Murdoch and Arthur Szlam. *Automatic Rule Extraction from Long Short Term Memory Networks*. 2017. arXiv: 1702.02540 [cs.CL].

[57] Mohammad Nuruzzaman and Omar Khadeer Hussain. "A survey on chatbot implementation in customer service industry through deep neural networks". In: *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*. IEEE. 2018, pp. 54–61.

[58] Yingwei Pan et al. "Jointly modeling embedding and translation to bridge video and language". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4594–4602.

[59] Simone Panicucci et al. "A Cloud-to-Edge Approach to Support Predictive Analytics in Robotics Industry". In: *Electronics* 9.3 (2020). ISSN: 2079-9292.

[60] European Parliament. Directorate General for Parliamentary Research Services. *The ethics of artificial intelligence: issues and initiatives*. EU Publications Office, 2020. DOI: 10.2861/6644. URL: https://data.europa.eu/doi/10.2861/6644.

[61] Eliana Pastor and Elena Baralis. "Explaining Black Box Models by Means of Local Rules". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. Limassol, Cyprus: ACM, 2019, pp. 510–517. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297328. URL: http://doi.acm.org/10.1145/3297280.3297328.

[62] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[63] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[64] Vitali Petsiuk, Abir Das, and Kate Saenko. "RISE: Randomized Input Sampling for Explanation of Black-box Models". In: *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*. 2018, p. 151. URL: http://bmvc2018.org/contents/papers/1064.pdf.

[65] Stefano Proto et al. "PREMISES, a scalable data-driven service to predict alarms in slowly-degrading multi-cycle industrial processes". In: *2019 IEEE International Congress on Big Data (BigData Congress)*. IEEE. 2019, pp. 139–143. DOI: 10.1109/BigDataCongress.2019.00032.

[66] Stefano Proto et al. "Useful ToPIC: Self-Tuning Strategies to Enhance Latent Dirichlet Allocation". In: *2018 IEEE International Congress on Big Data (BigData Congress)*. IEEE. 2018, pp. 33–40.

[67] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 1396–1408. URL: https://proceedings.neurips.cc/paper/2019/file/846c260d715e5b854ffad5f70a516c88-Paper.pdf.

[68] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. arXiv: 1606.05250 [cs.CL].

[69] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[70] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427.

[71] M. Roveri. "Learning Discrete-Time Markov Chains Under Concept Drift". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 1–13. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2018.2886956.

[72] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[73] Marcos Salganicoff. "Tolerating Concept and Sampling Shift in Lazy Learning UsingPrediction Error Context Switching". In: *Artif. Intell. Rev.* 11.1–5 (Feb. 1997), pp. 133–155. ISSN: 0269-2821.

[74] Sebastian Schelter, Tammo Rukat, and Felix Biessmann. "Learning to Validate the Predictions of Black Box Classifiers on Unseen Data". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020, pp. 1289–1299.

[75] Christin Seifert et al. "Visualizations of Deep Neural Networks in Computer Vision: A Survey". In: *Transparent Data Mining for Big and Small Data*. Ed. by Tania Cerquitelli, Daniele Quercia, and Frank Pasquale. Cham: Springer International Publishing, 2017, pp. 123–144. ISBN: 978-3-319-54024-5. DOI: 10.1007/978-3-319-54024-5_6.

[76] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* (Oct. 2019). ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: https://doi.org/10.1007/s11263-019-01228-7.

[77] Samad Sepasgozar et al. "A Systematic Content Review of Artificial Intelligence and the Internet of Things Applications in Smart Home". In: *Applied Sciences* 10.9 (2020). ISSN: 2076-3417. DOI: 10.3390/app10093074.

[78] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[79] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *CoRR* abs/1312.6034 (2013). arXiv: 1312.6034. URL: http://arxiv.org/abs/1312.6034.

[80] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[81] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[82] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[83] H. Strobelt et al. "LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 667–676. ISSN: 2160-9306. DOI: 10.1109/TVCG.2017.2744158.

[84] H. Strobelt et al. "Seq2seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 353–363. ISSN: 2160-9306. DOI: 10.1109/TVCG.2018.2865044.

[85] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41.3 (2014), pp. 647–665.

[86] Y. Sun et al. "Concept Drift Adaptation by Exploiting Historical Knowledge". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4822–4832. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2017.2775225.

[87] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[88] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[89] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[90] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[91] Mariarosaria Taddeo. “Trusting digital technologies correctly”. In: *Minds and Machines* 27.4 (2017), pp. 565–568.

[92] Valentin Trifonov et al. “Learning and Evaluating Sparse Interpretable Sentence Embeddings”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 200–210. DOI: 10.18653/v1/W18-5422.

[93] Alexey Tsymbal. “The problem of concept drift: definitions and related work”. In: *Computer Science Department, Trinity College Dublin* 106.2 (2004), p. 58.

[94] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[95] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[96] Francesco Ventura, Daniele Apiletti, and Tania Cerquitelli. *EBAnO - Explaining the convolutional decision-making process by mining its inner knowledge*. en. 2020. DOI: 10.24433/CO.6981109.V1. URL: https://codeocean.com/capsule/1347940/tree/v1.

[97] Francesco Ventura, Daniele Apiletti, and Tania Cerquitelli. “Explaining the convolutional decision-making process by mining its inner knowledge”. In: (To Be Defined).

[98] Francesco Ventura and Tania Cerquitelli. “What’s in the box? Explaining the black-box model through an evaluation of its interpretable features”. In: *arXiv e-prints*, arXiv:1908.04348 (July 2019), arXiv:1908.04348. arXiv: 1908.04348 [cs.CV].

[99] Francesco Ventura, Tania Cerquitelli, and Francesco Giacalone. “Black-Box Model Explained Through an Assessment of Its Interpretable Features”. In: *New Trends in Databases and Information Systems - ADBIS 2018 Short Papers and Workshops, AI*QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings*. 2018, pp. 138–149. DOI: 10.1007/978-3-030-00063-9_15. URL: https://doi.org/10.1007/978-3-030-00063-9%5C_15.

[100] Francesco Ventura et al. "A new unsupervised predictive-model self-assessment approach that SCALEs". In: *2019 IEEE International Congress on Big Data (BigData Congress)*. IEEE. 2019, pp. 144–148. DOI: 10.1109/BigDataCongress.2019.00033.

[101] Francesco Ventura et al. "Expand your Training Limits! Generating Training Data for ML-based Data Management". In: *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*. SIGMOD '21. Xi'an, Shaanxi, China, 2021. DOI: 10.1145/3448016.3457286. URL: https://doi.org/10.1145/3448016.3457286.

[102] Francesco Ventura et al. "Trusting natural-language black-box models via local and global explanations". In: *Knowledge and Information Systems (KAIS)* ((Under Review)).

[103] P. Vorburger and A. Bernstein. "Entropy-based Concept Shift Detection". In: *Sixth International Conference on Data Mining (ICDM'06)*. Dec. 2006, pp. 1113–1118. DOI: 10.1109/ICDM.2006.66.

[104] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. "Transparent, explainable, and accountable AI for robotics". In: (2017).

[105] S. Wang, L. L. Minku, and X. Yao. "A Systematic Study of Online Class Imbalance Learning With Concept Drift". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4802–4821. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2017.2771290.

[106] Matei Zaharia et al. "Apache Spark: A Unified Engine for Big Data Processing". In: *Commun. ACM* 59.11 (Oct. 2016), pp. 56–65. ISSN: 0001-0782. DOI: 10.1145/2934664.

[107] Q. Zhang, Y. N. Wu, and S. Zhu. "Interpretable Convolutional Neural Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. June 2018, pp. 8827–8836. DOI: 10.1109/CVPR.2018.00920.

[108] Xiaolin Zheng et al. "EXPLORE: EXPLainable item-tag CO-REcommendation". In: *Information Sciences* 474 (2019), pp. 170–186. ISSN: 0020-0255.

[109] B. Zhou et al. "Learning Deep Features for Discriminative Localization". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 2921–2929. DOI: 10.1109/CVPR.2016.319.

[110] Qifeng Zhou, Xiang Liu, and Qing Wang. "Interpretable Duplicate Question Detection Models Based on Attention Mechanism". In: *Information Sciences* (2020). ISSN: 0020-0255.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was LuaLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.