Semantics-aware image understanding

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

Semantics-aware image understanding

Andrea Pasini

* * * * * *

Supervisors

Prof. Elena Baralis, Supervisor
Prof. Benoit Huet, Co-supervisor

Doctoral Examination Committee:
Prof. Rosa Meo, Referee, Università degli studi di Torino
Prof. Elisa Quintarelli, Referee, Università degli studi di Verona

Politecnico di Torino
September 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.

. . . . . .

Andrea Pasini

Turin, September 2021

Summary

Deep learning models are characterized by high complexity and low interpretability, which are the payload for obtaining precise results in difficult tasks such as image understanding. Moreover, these models may suffer from an inadequate semantic understanding of the input data, as they are typically focused on a limited task (e.g., classifying images). Conversely, the human brain can learn from different real world activities and derive a more complete semantic knowledge. In this thesis, we design a methodology for inferring semantic knowledge directly from images, with the aim of enhancing image understanding tasks. Our work focuses on the study of object relationships, such as relative position and size, to infer a better semantic understanding of the analyzed pictures.

Our research provides a first application called SAD, a Semantic Anomaly Detection method to identify anomalies in the predictions of semantic segmentation neural networks. Semantic object relationships are exploited to derive an interpretable knowledge base, describing common configurations of normal objects. Our methodology highlights potential classification errors made by a neural network by identifying uncommon object relationships according to the learned knowledge base. The detected anomalies are presented to the user in an interpretable way, facilitating the analysis of the neural network accuracy.

Afterwards, we present SImS (Semantic Image Summarization), a framework designed to summarize big image collections. This task finds applications such as providing previews of personal albums (e.g., Google Photos) or suggesting thematic collections based on user interests (e.g., Pinterest). These objectives require a complete semantic understanding of the image collection, as simple visual features and textual tags would not be sufficiently informative. To achieve this goal, we propose a technique based on frequent subgraph mining, which analyzes scene graphs.

These data structures are automatically derived from the images by our algorithm and allow a complete representation of the image content. The output summary consists of a set of frequent scene graphs describing the underlying patterns of the collection. These results are more interpretable and provide more interesting descriptions with respect to previous techniques. Moreover, in the experimental results we show that our patterns achieve high summary quality in terms of coverage and diversity.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Elena Baralis, who made me grow during the Ph.D. period as researcher and teaching assistant. I also thank the reviewers of my thesis committee, Prof. Rosa Meo and Prof. Elisa Quintarelli, for their highly-valuable advice that allowed this thesis to grow in clarity and completeness. I also appreciated their positive comments that made me proud of this three-year work and encouraged me for my professional future.

I'd like to thank all the other professors - Paolo Garza, Daniele Apiletti, and Luca Cagliero - I collaborated with in this period. I would like to acknowledge all the colleagues at Lab5, for the time spent working together, hanging out, and during summer-school periods. In particular, a great thank to Eliana, who started this journey with me, Evelina, who integrated me in the group, and Elena, for her suggestions (and the nice walks between Politecnico and the train station). Thanks Flavio and Giuseppe, colleagues and roommates for a short but intense period, Francesco for his great technical advice, and Alessandro who is from my same town and shared with me a lot of pleasant train journeys. All of the Lab5 members were also fundamental to confront ideas and have support during 2020, which as we all know, has been full of difficulties. A special thanks to my parents and my brother, who supported (and bore) me in the most critical periods. I would also like to thank my cousin, Elisa, who was fundamental to encourage me to not give up with setbacks, my mentor Alex, and my friends (engineering colleagues) Luca and Ivan.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

During the last years, deep learning techniques are achieving increasingly higher performances in tasks such as image recognition, natural language processing, and sound analysis. The availability of big and heterogeneous data sets allows teaching these models how to recognize patterns characterized by high complexity and variability. Deep neural networks rely on many stacked layers to generate abstract representations of the input data and provide the final result, for example a class label or a segmentation mask, in the case of images. These complex structures allow understanding and classifying data with very high precision. However, the knowledge hidden in the inner layers of deep neural networks is not easily accessible since it is encoded in the form of numerical hyperparameters. For this reason, the inference process adopted by these kinds of models cannot be easily interpreted by human beings. Moreover, training a model for addressing a single task (e.g., classifying images), does not allow it to generalize the learned knowledge. Indeed, for example, gaining experience in multiple tasks and environments is fundamental for the human brain to understand more general patterns underlying the real world [30].

Following the previous considerations, deep learning models can suffer from an inadequate semantic understanding of the analyzed data. Fortunately, the application of semantics may help to tackle this issue. Semantic information can be provided by external sources, such as ontologies, or it can be derived directly from data, following a set of formal rules and procedures. Such representations can be used for obtaining interpretable descriptions of the analyzed data, with applications

in the fields of image understanding, speech recognition, and text processing.

Ontologies represent one of the most common approaches for encoding semantics. These formal languages, such as OWL (Web Ontology Language) [11], allow representing knowledge by means of concepts, properties, and relationships. Abstract concepts and categories (e.g., city, road), can be mapped to object instances (e.g., Turin, SP33) and form knowledge graphs or knowledge bases (e.g., ConceptNet [108], WordNet [82]). These data structures find common applications in data management tools for organizations and information retrieval. Furthermore, knowledge bases can be exploited by expert systems to address tasks such as medical diagnosis, fraud detection, and computer vision [7, 98]. In this thesis, we define semantics by means of high-level relationships between visual objects, with the aim of addressing different image understanding tasks.

Semantic reasoning finds analogies with the working principles of the human intellect, which is able to work with both low-level details and complex abstract representations [30]. In speech recognition and visual understanding, our brain considers local low-level proofs about sounds and objects, integrating them with higher-level contextual information [9]. Indeed, a high-quality interpretation of the external world requires an overall analysis of the available information. Unclear sounds or occluded objects can be reconstructed thanks to previous experiences and the abstract modeling of the real world priors. Different works in computer vision adopted this idea to improve standard image recognition techniques, either by creating new models [73] or by post-processing the results [39]. In our work, we will exploit contextual information as well, focusing on object relationships to describe the image content.

Although the research community aims at designing complex models with a full semantic understanding of the input patterns and with high-accuracy predictions, research studies have shown that another key property of machine learning models is interpretability. When deploying complex architectures to sensible fields, such as medicine, the final user may be interested in understanding the main reasons why the model produced a certain outcome. This requirement finds applications in trust, fairness, and debugging purposes [45, 94]. Deep learning models are known to be low-interpretable models (black boxes), as their internal weights hide the complexity of the reasoning underlying a certain prediction. Interpretability can be enhanced by analyzing the behavior of a model under slight variations of the input

or by analyzing its internal parameters [47]. Additionally, the research community is moving toward models that are interpretable by design. To this aim, our work relies on interpretable ways of representing semantics, such that the final results can be described with simple rules and visual representations.

Thesis statement: The aim of this thesis is to model the behavior of visual data by means of semantic and interpretable descriptions. The extracted knowledge can be exploited to enhance image understanding tasks by inspecting possible classification errors made by neural network models and to derive interpretable summaries from big image collections. The next section provides the complete dissertation plan.

1.1 Dissertation plan and research contribution

In this thesis, we aim at demonstrating that the introduction of semantic information in the computer vision field can enhance the understanding of the visual content and it is fundamental for generating interpretable and reliable results. As we will describe in Chapter 2, many image understanding techniques are purely based on deep learning models, which are hardly interpretable and require many labeled data samples.

Conversely, our research focuses on the extraction of semantic patterns from visual objects, trying to explicitly model contextual information and the overall description of the image. The proposed semantic approach is based on reasonings at object-level, which allow inspecting high-level patterns in a set of images. We managed to introduce this methodology in two computer vision fields, where the research community mainly considered lower level features (e.g., simple visual descriptors of the image) or black-box models such as deep neural networks. Specifically, we modeled the semantic image understanding task as a *knowledge extraction* process from a set of labeled images, with the following goals: (i) Learning common object relationship patterns to identify anomalies in labeled images (SAD [88]), and (ii) learning common object patterns to summarize the content of an image collection (SImS).

Both research lines share the same way of representing the visual information inside images. In particular, we designed a process to extend the semantic content

of *panoptic* and *semantic* image segmentation, which are techniques that mainly rely on deep learning models, as we will show in Section 2.1. Even if deep learning models can provide a detailed labeling of the input images, this information is limited to the association of pixels with a given object category. In our work we improve image understanding by introducing object relationships of different types, such as the *relative position*, *co-occurrence* and *size*. In Chapter 3 we describe our technique for inferring object relationships directly from pixelwise object labeling (i.e., semantic and panoptic segmentation).

We proposed a first application of the semantic reasoning about object relationships with our work on Semantic Anomaly Detection (SAD [88]). This study has the aim of identifying potentially misclassified objects in semantic segmentation, by comparing the input images with common high-level object patterns that are learned from a training dataset. In Chapter 4, after providing an overview of previous works in anomaly detection, we describe in detail the proposed semantic approach.

The second application, called Semantic Image Summarization (SImS [**pasini2020sims**]), exploits a similar learning pattern with the goal of summarizing an input image collection labeled with panoptic segmentation. We use Frequent Subgraph Mining techniques (FSM) to mine the important patterns that will be included in the final summary. Differently from our method, previous works (see Section 5.1.2) typically generate summaries based on visual features such as color histograms or Scale Invariant Feature Transform (SIFT). We show in Chapter 5 that our approach, based on the representation of the images with semantic scene graphs, is able to provide more interpretable and effective summaries.

1.1.1 Dissertation plan

This thesis is organized as follows. Chapter 2 provides background knowledge describing the fundamental image understanding tasks and their limitations. Chapter 3 describes a novel methodology for deriving object relationships, focusing on relative position. The aforementioned classifier is exploited in Chapter 4 to define SAD, our method for Semantic Anomaly Detection, and in Chapter 5 to derive image collection summaries. Finally, Chapter 6 draws conclusions and outlines future works.

Chapter 2

Background

Before describing the proposed research, we provide a complete overview of image understanding tasks, focusing on various types of image representation. Specifically, in Section 2.1 we will analyze the different image labeling approaches, in Section 2.2 we will describe previous works on visual context modeling, and in Section 2.3 we will review related works on scene graph representation.

2.1 Image annotation tasks

In the seventies, the earliest works in image classification were mainly based on template matching and part-based models [36]. The issues with these hand-crafted feature extraction methods were the high amount of effort for their design and the low scalability to more complex tasks. To contrast these problems, one of the first attempts of handwritten digits classification by means of trainable feature extractors was proposed in 1989 by LeCun et al. [70]. This work is popular for being the pioneer of a new type of neural networks based on trainable filters. The samples they used for training their model were then extended in 1998, giving rise to the famous MNIST dataset [71]. Later in 2012, with modern deep convolutional neural networks (CNNs), Ciregan et al. [25] succeeded in reaching approximately human performance in digit recognition.

The complexity of hand-written digit recognition tasks, in terms of image sizes and number of classes, was no more enough challenging and it was soon enhanced by more advanced classification problems. For example, in 2006-2007 the Pascal

VOC (Visual Object Classes) [34] project proposed new classification and detection benchmarks, focused on a wider variety of object classes. The dataset consisted of 9,963 images retrieved from Flickr and 20 object classes, such as animals, objects, and people. A subset of these images provided also pixel-level annotations to address the object segmentation task. In 2009, dataset size increased again with the Cifar-10 dataset [63], which provided 60,000 image samples belonging to 10 object classes (e.g., "airplane", "cat").

These new benchmarks brought researchers to design new architectures based on convolutional layers, achieving high-quality results. As a consequence, the growth of model complexity (i.e., number of parameters), which augments the likelihood of overfitting, required more and more data for the training process. Hence, a new attempt to scale up dataset size was the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [96], launched in 2010. The aforementioned benchmark contains 1.2 million training images with 1000 object categories, extracted from the ImageNet dataset [32]. Annotations include image classification and object detection labels. AlexNet [64], with 60 million parameters and 500,000 neurons, was one of the first models to successfully address the ILSVRC challenge, and for this reason it is considered one of the most influential papers published in computer vision. Together with prediction models, also dataset annotation techniques did significant improvements, relying on crowdsourcing frameworks where different workers are asked to either draw bounding boxes or validate the quality of existing annotations [109].

In the last ten years, newer datasets like Microsoft COCO [76], Cityscapes [28], and ADE20K [125] enhanced the complexity of image understanding, including many annotated objects in a single picture and proposing new tasks such as instance segmentation and panoptic segmentation [60]. To clarify the complex panorama of this research field, in the following we provide a categorization of the main image recognition tasks that have been developed so far.

2.1.1 Classification

One of the most straightforward image understanding tasks is represented by classification, which consists of describing a picture by assigning a single class label among a predefined set. Image classification algorithms typically output the

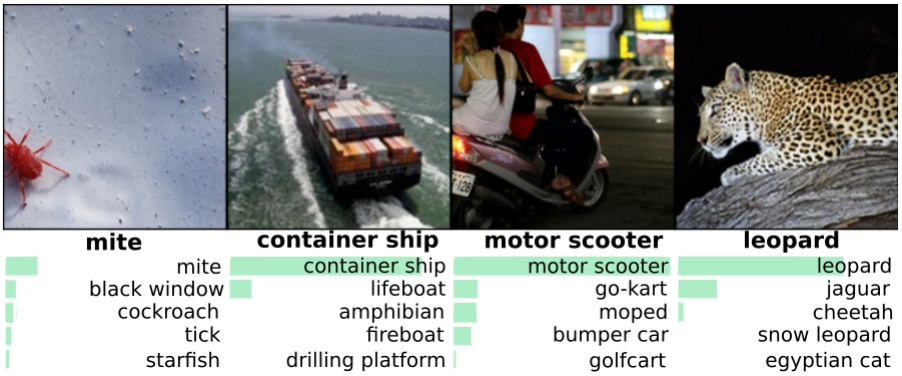


Figure 2.1: Image classification examples with AlexNet [64].

final class label or a discrete probability distribution specifying the likelihood of each class. Unfortunately, a single textual tag is not sufficient to describe complex visual scenes. For this reason, image classification datasets usually provide pictures containing one or few objects of the same type (e.g., a pet, a person, or a flower). Common applications of this approach are image retrieval, automatic personal album organization, user recognition, and medical data analysis.

Convolutional neural networks (CNNs) proved to be the best models for addressing image classification. These architectures are composed of different building blocks, such as *convolutional* layers and *pooling* filters. The former are responsible of a multi-stage feature extraction process, which consists of applying many sliding filters to transform the input features (or the input image for the first layer). Instead, pooling layers act as down-sampling filters to reduce the amount of processed information. Typically, CNNs terminate with a final *softmax* layer that is responsible for the generation of the output class vector with the different class likelihoods [44].

Various convolutional architectures have been proposed in literature. Most of these CNNs are used as backbones for detection and semantic segmentation, as we will see in the next sections. The turning point for the progress of convolutional networks is represented by AlexNet [64], which was successfully trained on the ImageNet dataset. This was the first method that managed to tackle the complexity of ImageNet, characterized by a very large number of classes and high in-class variability. Figure 2.1 shows some classification examples provided to AlexNet, where the confidence vectors of the predicted classes are shown in the form of

histograms below each image.

The structure of AlexNet consists of a deep convolutional neural network with five convolutional layers, three pooling layers and a 1000-neuron softmax layer. Its performances were overcome by VGGNet in 2014 [104], characterized by very small (i.e., 3x3) convolution filters and 16-19 weight layers. Even deeper models were proposed in 2015 by Google, such as Inception-v1 (GoogLeNet) [110]. This network is made of 22 convolutional layers with multiple size filters (i.e., 1x1, 3x3, 5x5). Unfortunately, such a high number of layers requires much data for the training phase and it more likely brings to the vanishing gradient issue. This problem occurs during back-propagation, where the derivative of the prediction error becomes smaller and smaller while flowing toward the input layer. A solution was proposed in 2016 with Residual networks [51], characterized by residual connections that allow the gradient to skip some convolutional layers when necessary. This architectural solution was fundamental to reach up to 152 stages, and it was precursor of other recent models like Inception-v4 [111].

Despite the wide range of applications, image classification is far from being considered a general methodology for image understanding. Indeed, it cannot identify multiple objects in the same picture and it is not designed to inspect object relationships. In the next section, we will focus on a task that partially solves this issue, namely object detection.

2.1.2 Object detection

Object detection requires to automatically inspect the position of each object in a picture and assign the correct class labels. It finds many applications, such as artificial vision for robotics or autonomous driving, security systems, and search-engines. The object positions are identified by means of enclosing rectangles, called bounding boxes. Specifically, the output of an object detection model is required to include a tuple of 4 values indicating the bounding box position and size (i.e., x, y, width, height) for each object. Every bounding box is then associated with a class probability vector.

The first object detection models worked by iteratively applying a classification CNN to rectangular subsets of the image at different scales. The rectangular regions with the highest prediction confidence were then included in the final result. Among

Figure 2.2: Object detection examples with YOLO [91].

these architectures, the R-CNN (2014) [43] was based on a region proposal method that used selective search to define up to two thousand bounding boxes where the CNN should be applied. This model was accurate but very slow since the convolutional neural network had to be executed for each proposed region.

To speed up the process, the Fast R-CNN [42] architecture derives a features map for the whole image, using VGG as backbone [104]. Afterwards, selective search is applied to the picture to propose a set of bounding boxes. These regions are exploited for cropping the features map at the specified positions. This operation, called ROI (Region Of Interest) pooling, allows deriving a set of features for each bounding box without running multiple times the convolutional layers. Each cropped features map can be finally processed with an MLP or an SVM classifier to derive the output labels.

This detection architecture was further refined in 2015 with Faster R-CNN [92], where selective search was replaced by a more efficient technique. Specifically, a region proposal network analyzes the features map to automatically derive the candidate bounding boxes. This operation reduces running time and allows obtaining better region proposals. A significant improvement was then brought by Redmon et al. with YOLO (You Only Look Once, Figure 2.2) [91], convolutional neural network (whose backbone is inspired by GoogLeNet [110]) that takes as input the whole image and directly predicts both the bounding box coordinates and the class vectors.

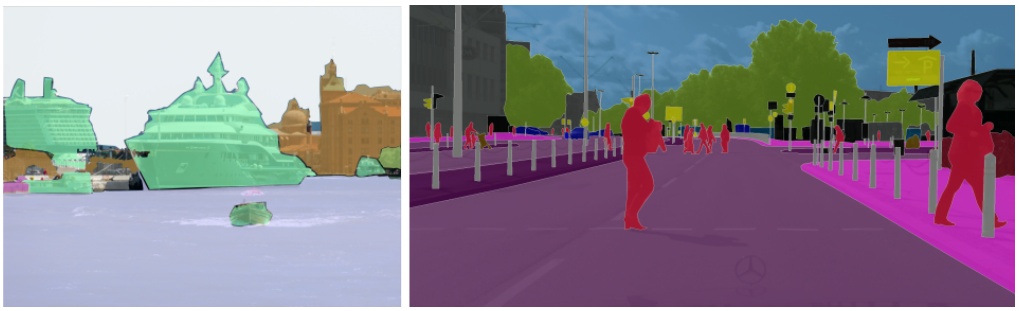


Figure 2.3: Examples of semantic segmentation for COCO [76] (left) and Cityscapes [28] (right) datasets.

2.1.3 Semantic segmentation

Semantic segmentation became popular with the need for more detailed labeling information. Indeed, differently from classification and detection, this task requires assigning an object class to each pixel of the image. This allows inspecting the presence of different objects, their position and shape, being suitable to address complex challenges such as autonomous driving, drone flight control, and robot-assisted surgery.

Datasets like Microsoft COCO [76], Cityscapes [28], and ADE20K [125] provide the correct annotation format to train semantic segmentation models in different domains. For example, Microsoft COCO presents 200K labelled images with 80 general purpose object categories (e.g.,"car", "cup") and 53 stuff classes (i.e., uncountable objects like "sky"). Other datasets are more specialized on a given domain, such as Cityscapes that focuses on street view images, with 30 different class labels. Figure 2.3 shows some examples of segmented images from COCO and Cityscapes.

Typically, the output of semantic segmentation is described with a three-dimensional matrix of shape $nClasses \times imageWidth \times imageHeight$. This entails a very complex task since, for each pixel of the image, the model is required to generate a vector with the class probabilities. The first methods for image segmentation were based on graphical models such as Conditional Random Fields (CRF) [65]. Later, with the development of convolutional neural networks, encoder-decoder architectures became more popular. Inside these models, the encoder is responsible for extracting a set of features whose size (i.e., width and height) is typically smaller than the one of the input image. The decoder has the task of upsampling these

(a) Semantic segmentation. (b) Instance segmentation.

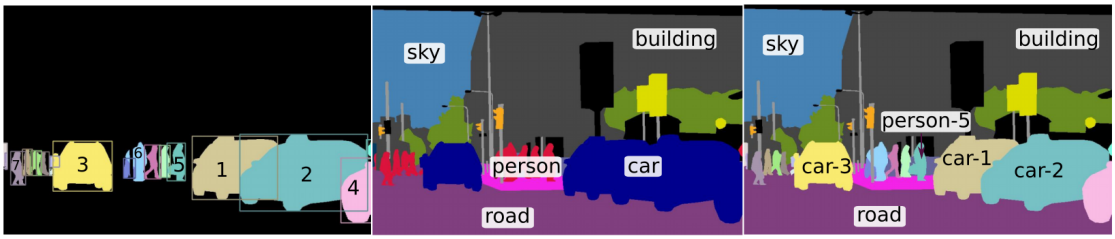
Figure 2.4: Differences between semantic and instance segmentation, on COCO [76] dataset.

features to match the dimension of the input picture and generate the final class labels. U-Net (2015) [95] and Seg-Net (2017) [8] are just some example neural networks following this architecture.

A lot of efforts have been made to improve the ability of these networks of considering contextual information, which is deemed to have high importance in this task. For example, some attempts exploited CRFs to understand the image globally and refine the final class labels [66]. However, these approaches were very slow and lost their popularity with newer neural networks. Indeed, recent methods successfully increased the receptive field (i.e., the amount of analyzed context) of convolutional neural networks by means of dilated convolutions (DeeplLab, 2017 [22]) and pyramid pooling layers (PSPNet, 2017 [123]).

2.1.4 Instance segmentation

Whereas semantic segmentation helps to describe visual objects in a very precise way, it still does not completely capture the semantic content of an image. Specifically, some objects with the same class (e.g., two or more cars, Figure 2.4a) may be touching together. With semantic segmentation, it would not be possible to distinguish the different object instances in this case. Moreover, occluded objects may be split into two or more segments in the same picture: In this situation, it would not be possible to understand whether the segments belong to the same instance. Differently from semantic segmentation, object detection can tackle this issue as each bounding box represents a single object instance. However, as a drawback, detection approximates object shapes with rectangular bounding boxes.



(a) Object instances ($\mathcal{Z}$). (b) Class labels ($\mathcal{L}$). (c) Panoptic segmentation.

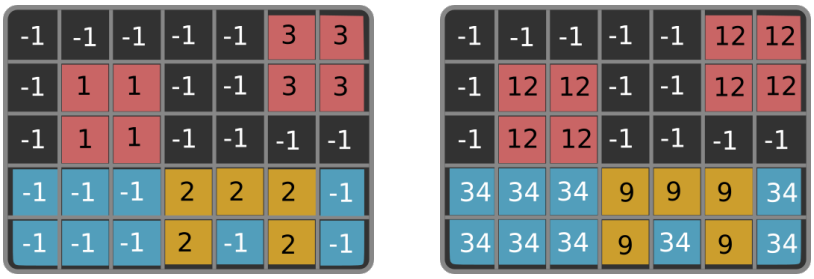
Figure 2.5: Examples of panoptic segmentation [60] on COCO dataset.

Instance segmentation brings the advantages of both semantic segmentation and object detection by identifying object instances with a bounding box and their shapes with a boolean mask (Figure 2.4b). The most popular architecture for instance segmentation is called Mask R-CNN [52], which is composed of a region proposal network followed by a model for classifying the objects and a decoder structure for generating the boolean masks.

2.1.5 Panoptic segmentation

Panoptic segmentation is a very recent computer vision task introduced in 2018 [60] with the Microsoft COCO dataset. The word *panoptic* derives from Greek and has the meaning of *overall view*: The objective is to merge instance segmentation and semantic segmentation within the same operation. COCO dataset distinguishes between countable *objects*, such as "car" or "table", and *stuff* classes, which refer to uncountable elements, like "sky" or "grass". In the case of countable objects, instance segmentation labels are provided, while for stuff classes the annotations are encoded with the semantic segmentation format. More specifically, panoptic segmentation assigns to each pixel of the analyzed image (i) a class label and (ii) an object identifier. Pixels that are labeled with stuff classes are associated with a placeholder identifier equal to -1, as their instances are not defined. The final output (Figure 2.5c) is encoded with two matrices $(\mathcal{Z}, \mathcal{L})$ with the same size of the input image. The former, $\mathcal{Z}$ (Figure 2.5a), specifies the object identifier of each pixel, while $\mathcal{L}$ (Figure 2.5b) associates the class label.

To clarify this description, Figure 2.6 shows panoptic segmentation applied to a very small-resolution image. This simplified picture reminds to a room with two red paintings on the background and a yellow table leaning on the blue floor.



(a) Object instances ($\mathcal{Z}$). (b) Class labels ($\mathcal{L}$).

Figure 2.6: Simplified example of panoptic segmentation labeling.

Figure 2.6a depicts the object identifier matrix $\mathcal{Z}$. It declares the presence of three object instances: 2 squares with identifier 1 and 3 (suppose they represent paintings on a wall), and a table with identifier 2. All the other pixels are identified with the placeholder -1, which means either they do not belong to a specific class or they belong to stuff annotations. Figure 2.6b completes the analysis by depicting the labeling matrix $\mathcal{L}$. Specifically, it shows that the two instances 1 and 3 belong to the same class 12 (i.e., painting), while the instance with identifier 2 belongs to class 9 (i.e., table). The blue pixels, which belong to a stuff annotation (marked with -1 in Figure 2.6a), are labeled with class 34 (i.e., floor). Finally, the background pixels are labeled in $\mathcal{L}$ with -1, which means that their class is not specified (either because a model is not able to recognize it, or because the ground-truth labels in the considered dataset do not include a background class).

The first approaches to panoptic segmentation worked by applying heuristics to merge the results generated from separate instance segmentation and semantic segmentation models [60]. Some of these techniques also defined ways of addressing occlusion issues between the predicted instances [68]. Other models, instead, take the advantage of a single convolutional neural network to directly produce the desired output [59, 116, 117].

In this thesis, we will exploit panoptic segmentation to derive object relationships, such as the relative position, and infer the necessary contextual information for anomaly detection (see Chapter 4) and image collection summarization (see Chapter 5). In the next section, we will review previous works on contextual information modeling, with the aim of better understanding the contribution of this thesis.

2.2 The role of contextual information

Interesting studies on the human brain processes that are responsible for object recognition show that contextual information plays a very important role [9]. Specifically, reasoning on context is fundamental when trying to recognize objects that are partially occluded or that are too far for being properly in focus [87]. For instance, many different object categories may be visually similar, such as a calculator and an old mobile phone. In these cases, the different context frames, such as the environment type (e.g., "kitchen", "garden") or the other surrounding objects in the image, have demonstrated to be helpful in the recognition task for the human brain [9].

In another study, Biederman et al. [13] demonstrated the importance of contextual information for human subjects by inspecting object relationships. The author identified five different types of relationships between objects in the same image: (i) interposition, describing objects occluding each other, (ii) support, for objects that are lying on surfaces, (iii) object co-occurrence in specific scenes, (iv) position in the image, and (v) relative size between objects. The study showed that object recognition suffers from a violation of such common relationships, causing a slower detection process or inaccurate interpretations.

2.2.1 Context types

Deep learning models for object recognition (i.e., detection or segmentation) are typically based on the appearance of visual features, such as colors, textures, edges, and shapes. These architectures implicitly include contextual information thanks to a wide receptive field of convolutional filters. Whereas the increase of the receptive field may be a solution for capturing more context [22, 123], a parallel research branch in computer vision is focused on the integration of deep learning models with explicit contextual information.

Literature works inspect the usage of context in two different ways. Some methods exploit post-processing techniques to refine the class labels of the identified objects, while others embed context modeling directly inside neural networks. Moreover, two types of contextual information have been identified: *local context* and *global context* [39]. Local context is defined as the information that can be retrieved from the immediate surroundings of the object to be recognized. For example, when

trying to distinguish a soccer player from a tennis player, the color of the playing field around the person may help in the detection. Visual features, such as colors and textures, are not the only type of information that should be considered for local context. Indeed, the relationships between the target object and those in its neighborhood can be exploited for a better scene understanding. For example, the relative position between objects could describe common patterns in real-world scenes, such as "chair is likely near table" or "water bottle is likely on table", which could be useful for refining the predicted object labels. Also scale relationships may be used for this purpose. For example, a bush will be typically smaller than a person, while a tree will be taller. Since scale context is more difficult to be considered due to perspective reasons, some methodologies solve this issue by mapping perceptual object sizes to their real ones by means of perspective reasoning [24].

Global context is instead more focused on the semantic meaning of the overall scene. To make an example, if the image under analysis represents a kitchen, this knowledge will help in recognizing objects like "oven" and "sink" that occur with a higher probability in this kind of scene. The relative position between objects may be also used for global context. For instance, the estimation of the object classes in a given scene could be refined based on the overall configuration of the objects in the image.

2.2.2 Contextual information for object recognition

To give some additional details about the usage of contextual information, we propose in this section a review of popular methods in literature. We divide our analysis by focusing first on the methodologies that exploit contextual information by integrating the results of standard object detectors. Afterwards, we will review other techniques based on deep learning models that embed context directly into the neural network.

Context as a separate building block

Instance level reasoning is certainly one of the most effective ways of modeling context. For example, Zitnick et al. [129] inspected the importance of contextual features, like co-occurrence, absolute and relative object location, for interpreting

generated image clip arts. They exploited a statistical measure, called *mutual information*, to understand the effect of each context type when trying to select scenes with semantically equivalent meanings. Similarly, Divvala et al. [33] conducted an empirical study on real images about context in object detection. They employed different types of contextual information, including global context (e.g., scene-level features), co-occurrence, absolute position, and size. The empirical results of these works demonstrated that high-level semantic information is fundamental for the tasks of image classification and understanding.

Simple detection algorithms focus on a limited number of context types. For instance, some of them only rely on the co-occurrence of the object classes, computed by counting the number of times an object pair appears in the training pictures. The results of baseline detection models are then typically refined by means of Conditional Random Fields (CRFs) [89]. The authors in [80] proposed a novel way of modeling object co-occurrence, without explicitly considering object classes. Their approach builds a *visual memex*, which is a graph containing as nodes all the dataset object instances, described with their visual features. A first type of graph edges connects visually similar object instances, which more likely belong to the same class. Additionally, another set of contextual edges connects the object instances that appear in the same image in the training set. When a new image is being labeled, its object instances are compared with the ones in the visual memex, then both contextual edges and visual similarities are used to refine the initial labels predicted by a baseline classifier.

A similar methodology is exploited in [72] for a different purpose. Given the application of a baseline detection method to a set of images, the output objects that present lower prediction confidence (i.e., they may be misclassified) are considered to discover possible new class labels in an unsupervised way. The contextual information of these objects is described by means of relative position relationships (e.g., above, below) with other instances in the image. At this point, the visual features of the low-confidence objects in the dataset, together with their context vector, are clustered to discover new categories. The obtained cluster centroids can be used to describe the discovered classes from a visual and contextual point of view.

Galleguillos et al. model the global contextual information by means of co-occurrence and relative position, which is described with a set of discrete values:

Above, below, inside, around. In their work, images are firstly segmented by color, then each region is classified by a *Bag of Features model (BoF)*. A post-processing step, based on a CRF, is employed to modify the class labels using the co-occurrence probabilities and the relative position of the objects. Similarly, in [24] the authors designed a context model based on Gaussian models. For the whole training set, co-occurrence and relative position are modeled with Gaussian distributions to define the common behavior of each class. Detections on new images are obtained by solving an optimization problem that takes as input the class scores of the baseline detector and the statistical context model.

In the field of semantic segmentation, Myeong et al. [85] integrated context into a *Markov Random Field (MRF)* model. The proposed system, given a query image to be segmented, extracts the k most similar samples from the training set, according to global features such as color histograms, spatial pyramid [69], and gist (i.e., exploiting PCA components to extrapolate a synthesis of the image content) [86]. Among the retrieved images, they consider object pairs as context exemplars to improve the labeling of the query picture.

Context embedded in deep learning models

Differently from the methodologies described so far, some works propose deep learning models that directly embed context reasoning in their framework. Some of these papers address local context modeling, while others are more focused on global context.

Region-based CNNs (R-CNNs, see Section 2.1.2) rely on the ROI pooling module, which provides to the final classifier only the information included in the bounding box under analysis. To obtain better results, useful local contextual information can be considered by augmenting the ROI size. Gidaris et al. [41] proposed an R-CNN-based model that embraces this approach. Specifically, for each proposed ROI, other bounding boxes including local context information are generated. These bounding boxes consist of the outer part of the original ROI (i.e., a bigger bounding box containing both the object under analysis and local context), and the left/right/up/down regions. The latter include only context, without information of the object under analysis. After their generation, the original ROI and the context bounding boxes are exploited for ROI pooling to obtain the final features and feed them to the final classifier. Another similar R-CNN is proposed

in [127]. This network, called CoupleNet, is capable of integrating standard ROI pooling with a double-sized region that also includes local context.

The authors in [121] suggested to use contextual bounding box features (such as [41] and [127]) with an additional gating module. They motivate their work with two example images. In the former, the detection of a rabbit head is facilitated by the identification of the rabbit ear. In the latter, the rabbit ear brings damage to the correct detection (e.g., a girl wearing a hat with rabbit ears). To solve this issue, they designed a methodology based on message passing to propagate local contextual information only when really necessary for the detection.

Global context, together with the local one, have been integrated into the same model by Jianan Li et al. [73]. For the local context, they consider multiple-scale bounding boxes around each Region Of Interest, while for the global context they include information from the full image features, filtered with an attention-based block to exclude misleading information. The neural network responsible for the classification of each ROI concatenates the contextual information with the object features and provides the final results. The global image information is also exploited in [12]. In this case, the whole picture is processed by a Recurrent Neural Network (RNN) responsible for summarizing the contextual information. In the following, we review some other interesting techniques that consider global context in a very different way. These few methods embed in the deep neural networks specific building blocks for context reasoning at instance level.

Chen et al. [23] introduced a spatial memory for context reasoning, demonstrating that their model is capable of recognizing very small and low-resolution objects. The proposed model outputs the different detections by iterating multiple times on the same image. At each step, the input image and the information of the previous detections (i.e., the spatial memory) are considered for generating a new bounding box. The spatial memory is updated at each iteration, by including the predicted object and its position in the image.

A different approach based on instance-level reasoning was proposed in [78]. The detection process is modeled as a graph structure inference task, where the objects are represented with nodes and their relationships with edges. The architecture is composed of a GRU (Gated Recurrent Unit, a type of RNN) for each node, which has the aim of identifying the correct class label for the corresponding bounding

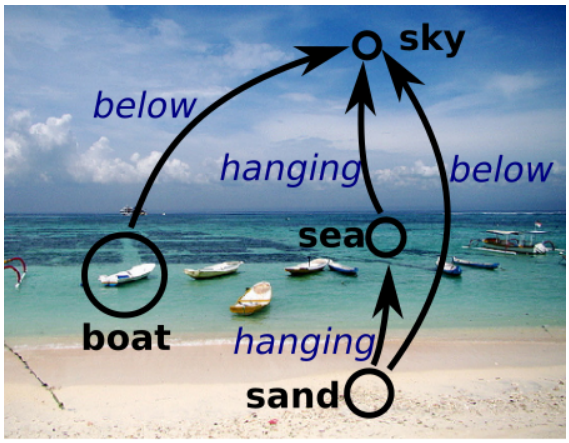


Figure 2.7: Example of scene graph overlapped to a picture.

box. A message-passing system among the different GRUs is adopted to model the contextual information. At each iteration the GRU hidden states are updated based on the exchanged messages and, at the end of the process, the final object labels are given. When designing the types of messages exchanged by the GRUs, the authors considered both scene-level contextual information and object relationships. Specifically, the feature vectors associated with object relationships include, for each object pair, the relative position and size (computed with bounding box margins), and the ROI visual features.

Finally, Hu et al. [55] designed a novel attention model for object detection. A relation module is integrated into standard R-CNN architectures to inspect object relationships. For each object pair, the attention system computes a feature vector including object relative positions and relative visual features. The description of each bounding box, necessary for the final classification, is built by adding the relationship contributions generated by the attention module for all the object pairs.

2.3 Scene graphs

In this section, we review some previous works that use the concept of scene graph, which we will use to model contextual information in the next chapters of this thesis. This specific graph structure includes nodes, typically representing object instances, and edges of different types, usually encoding different relationship categories. Figure 2.7 depicts an example scene graph with position relationships.

In computer vision, one of the most popular applications of scene graphs is image retrieval. For example, Fisher et al. [37] look for similar images by comparing scene graphs derived from 3D meshes. The scene graphs include relative position relationships among the detected objects and they are compared by means of graph kernels. Another option for image retrieval is to use textual descriptions of the image content. The sentences are first translated to scene graphs by means of recurrent neural networks, then they are used to retrieve meaningful pictures from the dataset [14, 100].

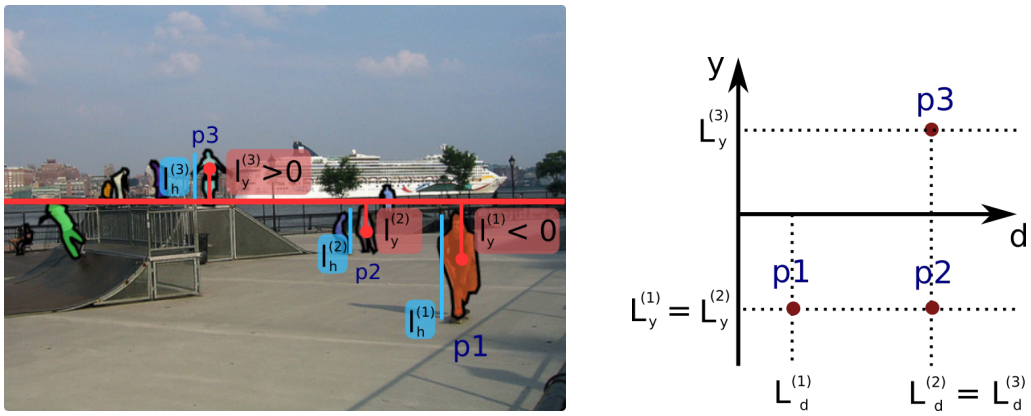
A second common application of scene graphs is image generation. Some literature works focused on the design of deep learning models, based on graph convolutional neural networks, that analyze scene graphs and generate realistic images [6, 57, 99, 114]. The scene graphs used in these works can be manually generated (e.g., from the Visual Genome Dataset [62]) or automatically derived from a query image by looking at the relative position of the different object bounding boxes.

As we will detail in Chapter 3 and 5, in our work we generate enhanced scene graphs by including fine-grained object relationships, automatically derived from panoptic segmentation labels. We will show that deriving the relative position directly from bounding boxes or object centroids [40, 57] is more limited than the proposed methodology.

Chapter 3

Diving into semantic image understanding: Position relationships

As anticipated in the previous chapter, an important part of our contribution lies in the computation of object relationships to enhance the understanding of image semantics. The learned representation will be exploited later in this thesis for anomaly detection (SAD) [88] and image collection summarization (SImS) [**pasini2020sims**]. Among the different types of relationships, this chapter focuses on the analysis of the relative position between object pairs (e.g., "above", "inside"). We design a technique to derive the relative position starting from either semantic or panoptic segmentation. In the case of semantic segmentation, an additional effort must be performed to identify the different object instances, as shown in Section 3.2. Conversely, by definition, panoptic segmentation already distinguishes the different objects. Section 3.3 presents the classification process that we specifically designed to compute the relative position between each pair of objects in the same image. The effectiveness of this methodology is instead verified in Section 3.4.



(a) 2D image.

(b) Horizontal section in the 3D representation.

Figure 3.1: Projection of 2D objects to a 3D space [24].

3.1 Related works on relative position

One of the key aspects of our SAD approach and SImS is the computation of the relative position between objects. We aim at inspecting semantic relationships, such as "on" or "inside", which can be accurately detected by considering the shape of the objects inside the image.

In the following, we provide a review of previous work that attempted to infer object positions. We organize the section as follows. First, we will describe the methodologies that compute object positions with a set of coordinates in the 2D or 3D space. Afterwards, we will focus on other techniques that consider bounding boxes and object centroids to derive discrete position descriptors (e.g., "above", "below"). Finally, we detail the previous works on *string-based* methodologies that inspired our technique.

3.1.1 Coordinate-based positions

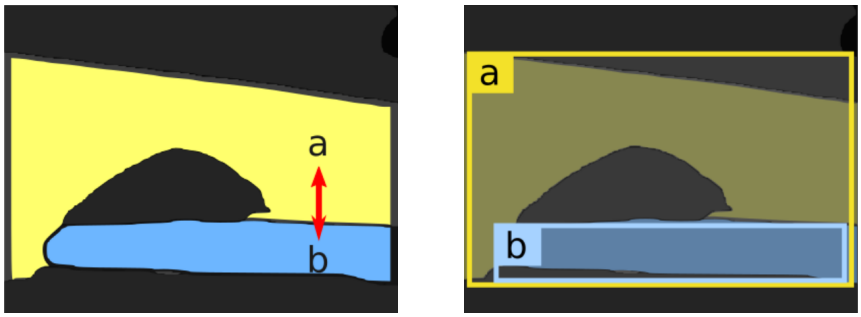
Myung Jin Choi et al. [24] base their approach on the location of bounding box centers. They show how to infer the object positions in the 3D world by only exploiting bi-dimensional coordinates and the average size of an object in the real world (e.g., 1.70 for a person). Let l_y, l_h be the vertical position of the bounding box center, and the bounding box height of a specific object. The perceptual vertical position (i.e., l_y) is 0 at the center of the image, lower than 0 below the center

and greater than 0 above. For example, Figure 3.1a depicts these variables for three skaters ($p1$, $p2$, $p3$) detected by a segmentation model. A horizontal red line highlights the horizon of the image. Let H be the average physical height of the object in the real world, inferred exclusively based on the object class. For this task the authors used a dataset that associates each object class with its average height. The technique for inferring the 3D coordinates firstly consists in computing the distance between the object and the observer (i.e., depth). This process is based on the assumption that the higher is the perceptual height of the object in the image, the closer is the observer. Hence, depth is given by $L_d = H/l_h$. For example, in Figure 3.1a, $p2$ and $p3$ have the same perceptual height and are depicted at the same depth (L_d) in Figure 3.1b. Conversely, $p1$ is closer as it has a greater perceptual height. The vertical position L_y of an object in the 3D world is computed with a similar heuristic reasoning. With a fixed L_y, the objects that are further from the observer (i.e., higher L_d) appear closer to the horizon (i.e., lower $|l_y|$) in the 2D image. Hence, $L_y = l_y \cdot L_d$. For example, in Figure 3.1a, $p2$ is further from the observer than $p1$. In the meantime $p2$ is also closer to the horizon than $p1$. Indeed, the two objects are depicted at the same height in Figure 3.1b. Even if these heuristic reasonings work in many cases, they typically fail with different perspectives (e.g., when the horizon is not in the middle of the image) and occluded objects (i.e., the height is erroneously inferred).

A second approach that exploits the coordinates of object centroids for computing positions is [72]. Differently from [24] this model infers a depth map using object occlusion and visual cues as specified in [54]. Specifically, depth is exploited as an additional feature together with the centroid coordinates.

3.1.2 Discrete positions with bounding boxes

These methods follow the approach of assigning a position label (such as “above” or “below”) for a given object pair. This allows to obtain a high level semantic description of the position, without referring to numerical values that are hardly interpretable. Galleguillos et al. [40] extract 3 features to model the relative position between two objects i, j. The first one analyzes the vertical position by measuring the difference between the y component of the object centroids. The other two features measure the percentage of overlap of bounding box of i with respect to the bounding box j and viceversa. The authors finally compute a feature vector for



(a) Actual shape. (b) Bounding boxes.

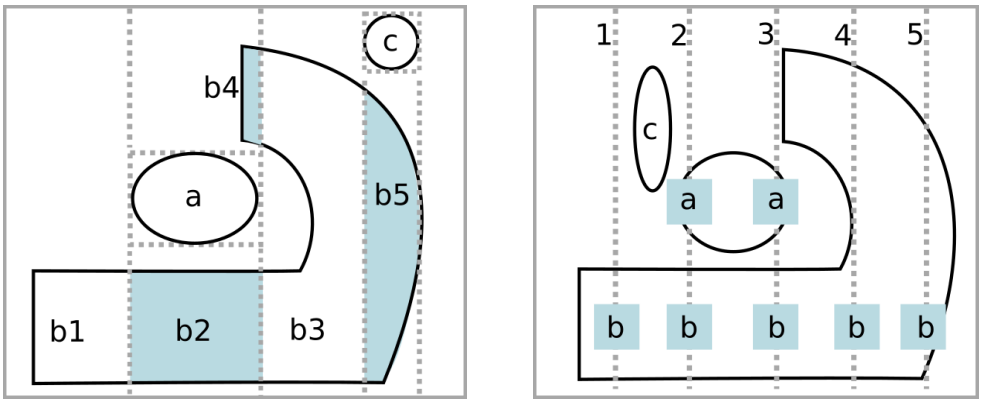
Figure 3.2: Issues with position computation by means of bounding boxes.

each object pair (i, j) in the dataset, then use clustering to learn, in an unsupervised manner, four groups of positions: *above, below, inside, around*. Differently from the technique we provide in our work, this methodology is unsupervised and does not inspect the object shapes (i.e., bounding boxes are often too coarse to describe the objects). They also do not distinguish between objects that are touching and those that are far in the image.

A similar work is represented by [93], where the authors identify 6 discrete positions (i.e., *left, right, up, down, touch*, and *front*) by inspecting bounding box centers and their overlap. Even if this technique tries to inspect touching objects, simple bounding box overlap may not be sufficient to achieve this task. Indeed, the bounding box shapes may be misleading since the objects may not occupy all the rectangular region. Bounding boxes are also too coarse for distinguishing between positions like "inside" and "above". Figure 3.2a provides an example to confirm this reasoning. The bridge depicted in the image (object a) is visually positioned on a river (object b). However, Figure 3.2b shows that, when considering only bounding boxes, the correct position relationship cannot be inferred (i.e., bounding box b is inside bounding box a). This issue is due to the irregular shape of the objects that cannot be accurately approximated with the enclosing rectangle.

3.1.3 String-based methodologies

We previously highlighted the need for better shape descriptors that provide the necessary information for computing object positions. When focusing on the vertical ordering of the objects in an image, the work proposed by Chang et al. [21] highlights an interesting solution. This method considers each object in the



(a) Bounding box projections [21]. (b) Five vertical scans [107].

Figure 3.3: Examples of string-based image representation.

image as a separate shape. These elements are inspected in pairs (i, j) to infer their relative positions. Specifically, the algorithm extends vertically the bounding box of i until it intersects the shape of j. The sub-regions of j that originate from the intersection with the extension of the bounding box of i are identified in the image. For example in Figure 3.3a the bounding box of object a is extended to intersect b. This generates the sub-regions *b2, b4*. Similarly, c is projected to b, generating *b5*. After this step, the image is described by means of vertical *strings* that specify the ordering of the obtained sub-regions. More specifically, each string contains, from left to right, the name of the sub-regions that appear in order from the top to the bottom of the image. For example, Figure 3.3a can be described with the strings $\{b1\}, \{b4\ a\ b2\}, \{b3\}, \{c\ b5\}$. The authors used these string representations to query image databases (e.g., identifying recurring patterns inside strings), but this kind of descriptions could be also used for inferring the relative vertical position for a specific object pair. However, we identified two limitations of this representation. The first drawback is that it cannot describe when two object are touching vertically. Indeed, if there is some void space between the two objects, this will not be encoded by the string. Moreover, the bounding box projections may be too coarse for a correct understanding of the object positions. For example, object a is between *b2* and *b4* in Figure 3.3a. However, *b4* represents a very thin region with respect to a. A more "fuzzy" approach should be able to consider the importance of each sub-region based on its wideness. For example, object a could be considered above object b, since the sub-region *b4* is not sufficiently wide to justify the relationship a *inside* b. We will show in Section 3.3 how our methodology overcomes the mentioned

issues.

A second work on string-based representations [107] derives the vertical object ordering by means of five equally spaced vertical scans of the image. The picture is firstly segmented by uniform color regions (i.e., objects). Afterwards, for each of the five scans, the algorithm specifies a string with the order of the detected objects from top to bottom. An example is depicted in Figure 3.3b, where the five scans output the following strings: $\{b\}, \{a\ b\}, \{a\ b\}, \{b\}, \{b\}$. For a better compression of the generated representation, the authors propose to derive a matrix M where each cell $m_{i,j}$ counts the number of times that object i is found before j in the image strings. In this way M allows understanding the vertical ordering of two objects (e.g., if the sequence ij is found many times, then object i can be considered above j). The approach suffers from similar problems of [21], as the vertical scans are very few representative of the image and may miss important regions (e.g., region c in Figure 3.3b).

From the previous works mentioned so far, we obtain the following takeaways. The simplification of object shapes to centroids or bounding boxes yields imprecise computations of the relative position. For this reason we inspired from the works on string representation [21, 107] to obtain more detailed results. Specifically, the idea of string-representations is extended in our work to identify finer details in the images and generate features for training a relative position classifier. In our work we selected 9 discrete relative position labels to inspect the vertical position of the objects. Differently from the other methods, we enhance the semantic relationships by distinguishing between "above" and "on" to identify when the objects are touching. Furthermore, when the objects are not vertically aligned, differently from [93] we do not distinguish between "left" and "right", since images can be typically mirrored horizontally without changes in their meaning. Conversely, we distinguish between "side", "side-up" and "side-down", to better inspect the vertical position even when the objects are not aligned along the vertical axis.

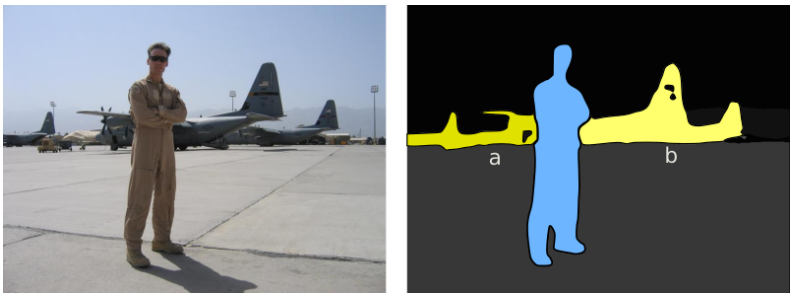


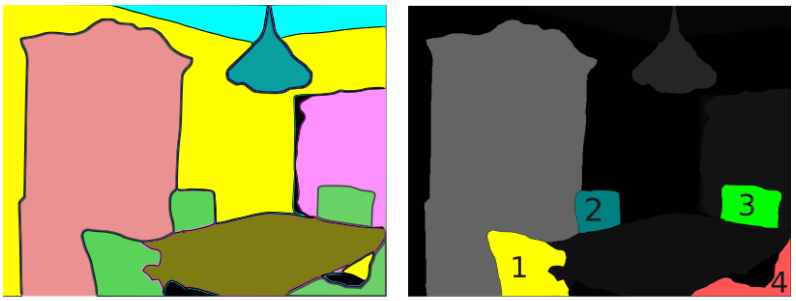
Figure 3.4: Object occlusion. The airplane is split by the man into two regions.

3.2 Semantic segmentation to panoptic segmentation

The algorithm we designed for computing relative object positions, which will be described in Section 3.3, requires the identification of the different object instances in the image. For a more general applicability, we aim at executing our classifier on the output of either semantic segmentation or panoptic segmentation. Remind from Section 2.1.3 and Section 2.1.5 that semantic segmentation is designed to assign a class label for each pixel in the image, without identifying object instances. Hence, this representation is not suitable to identify which pixels belong to the same real-world object. For example, if the shapes of two cars are touching, with the output of semantic segmentation we are not able to distinguish them as two separate objects. Conversely, with panoptic segmentation we can categorize the image content into stuff (uncountable objects, such as sky or grass) and instances (countable objects, such as car or person). In the case of instances, we are also able to distinguish between different objects even if their shapes are touching.

In this section we define a preprocessing step that allows obtaining object instances also from semantic segmentation, with the limitation that distinct objects should not be touching together to be correctly identified. Specifically, we provide a heuristic technique to convert semantic segmentation results to the standard format used for panoptic segmentation (Section 2.1.5).

The proposed method assumes that each object is characterized by a fully connected region in the image, which maintains the same class for all its pixels. This simplifying assumption may be violated in specific cases, as depicted in Figure 3.4.



a) segmented image b) components for "chair"

Figure 3.5: Extraction of connected components.

In this example, a person is dividing the airplane on the background into two distinct parts (i.e., *a, b*). This occlusion issue cannot be solved with the information provided by semantic segmentation. To avoid this drawback, panoptic segmentation should be directly computed on the image, without applying the conversion to semantic segmentation as described in this section.

Let $\mathcal{L}$ be a matrix assigning a class label to each pixel of the image (i.e., semantic segmentation). Our technique performs a breadth first search (BFS) to discover patches of adjacent pixels with the same class. We call each distinct patch of adjacent pixels with the name *connected component*. Afterwards, the algorithm assigns a unique identifier to each connected component, assuming that each of them represents a distinct object instance. This allows us to obtain a new matrix $\mathcal{Z}$ that assigns each pixel to the correct object identifier (i.e., connected component). The obtained matrices $(\mathcal{L}, \mathcal{Z})$ are then coherent with the definition of panoptic segmentation, as previously described in Section 2.1.5. Figure 3.5 depicts an example of connected component detection. In Figure 3.5a, the image colors provide a visualization of the matrix $\mathcal{L}$, which is the result of a standard semantic segmentation classifier. Specifically, the green pixels belong to class *chair*. The instance matrix $\mathcal{Z}$, computed by our BFS methodology, is shown in Figure 3.5b. We highlighted the different connected components of class *chair* with four distinct colors and numerical identifiers in range $[1, 4]$. These components represent the four instances of class chair identified in the matrix $\mathcal{Z}$.

Label	Description
above	s is above r without contact
below	s is below r without contact
on	s is on top of r with contact
hanging	s is below r with contact
side	s and r are not vertically aligned
side-up	s and r are not vertically aligned, s is in a higher position
side-down	s and r are not vertically aligned, s is in a lower position
inside	s pixels are inside r shape
around	s pixels are around r shape

Table 3.1: Relative position labels for a subject-reference pair.

3.3 Relative position computation

In this section we describe our algorithm for computing the relative position between two objects. Based on the considerations made in Section 3.1, we accurately designed a technique relying on the string representation. This way of modeling an image allows inspecting the relative vertical position, which is relevant for scene understanding since pictures are not symmetric along this direction. For example, regions like "ceiling" and "floor" cannot be exchanged along the vertical axis. On the contrary, the horizontal position is less significant because the scenes can often be flipped horizontally without producing any change of meaning. Moreover, understanding whether two objects are touching along the vertical axis is important to understand structural relationships in the represented scene. For example, it could be fundamental to understand whether an object is laying on a specific surface, or it is just above without contact. Our modified version of the string representation is capable of recognizing these details, giving the possibility of defining 9 different relative position labels. The list of labels that can be assigned to a pair of objects in the same image is provided in Table 3.1. Each position describes how a *subject* (s) is positioned with respect to a *reference* (r).

The *above/below* positions describe a subject that is at the same horizontal position of the reference, but it is vertically separated by an interleaving region. This allows inspecting pairwise relationships where there is no contact between the objects. Figure 3.6a depicts two examples of the *above* property. The first one represents a *building* above the *street*, while in the second one you find a room where a *slide viewer* is above the *floor*, separated by a thin region belonging to

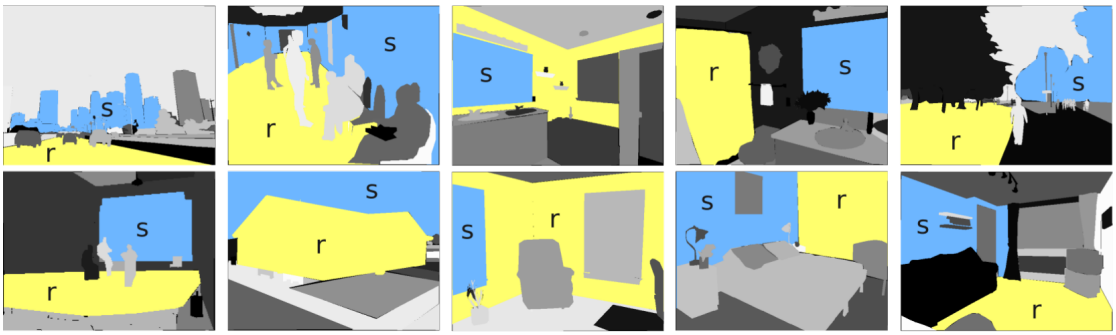


Figure 3.6: Object positions. This image provides some examples of position relationships between a subject (light blue region marked with s) and a reference (yellow region marked with r).

the background wall. When the subject and the reference are instead touching and vertically aligned, they apply to the *on/hanging* positions. If the subject is laying on the reference, we use *on* (Figure 3.6b), while *hanging* when the subject's top touches the bottom edge of the reference. In the case where the two objects are one inside the other, we assign them to the *inside* (s included in r, as shown in Figure 3.6c) and *around* (s surrounding r) positions. Finally, when the two objects are not vertically aligned we define the *side, side-up, side-down* positions. As previously anticipated, we do not distinguish between "left" and "right" positions, but we identify the different heights (i.e., up, down) where the subject is located. The *side* position (Figure 3.6d) is selected when the subject is at the same height of the reference. The 9 properties described so far are mutually exclusive (i.e., only one can apply to a specific object pair) and symmetric. This means that the relative position classifier can be applied only once for each pair, after selecting which of the two objects is the subject. When inverting the role of subject and reference in the same pair, the associated relative position can be derived by selecting the opposite class label (e.g., above-below, on-hanging, inside-around, ...).

As anticipated in the beginning of Chapter 3, our position classifier derives the string representation from images labeled with panoptic segmentation (or with semantic segmentation that has been post-processed as described in Section 3.2). Let s, r be the object identifiers (extracted from the panoptic segmentation matrix $\mathcal{Z}$) of a subject and reference, respectively. Our algorithms assigns to the object pair one among the 9 relative position labels presented in Table 3.1. This classifier

Rule	Label	Applies if	Example strings ($\mathcal{Z}_x$)
r_1	*s on r*	r is directly after s	$\{..., s, r, ...\}$
r_2	*s hanging r*	s is directly after r	$\{..., r, s, ...\}$
r_3	*s above r*	r is after s with interleaving region	$\{s, ..., r\}$
r_4	*s below r*	s is after r with interleaving region	$\{r, ..., s\}$
r_5	*s around r*	r is between s	$\{s, r, s\}$ or $\{s, ..., r, ..., s\}$
r_6	*s inside r*	s is between r	$\{r, s, r\}$ or $\{r, ..., s, ..., r\}$
r_7	*other*	none of other statements applies	$\{r, s, r, s\}$

Table 3.2: Rules to extract relative position features from strings.

infers the relative position based on a set of features that are retrieved from $\mathcal{Z}$. The training process was conducted on a dataset of manually labeled samples that we built specifically for this task, due to the unavailability of existing online resources. More details on the choice of the classifier and on the position dataset are given in Section 3.4. In the following we provide a complete description of the feature extraction process.

3.3.1 Relative position features

We distinguish between two groups of features that are exploited by our algorithm: (i) *string-based*, and (ii) *bounding-box-based*. The first group ($f_1 - f_7$) relies on an extension of the standard string based representation we described in Section 3.1. Since it inspects the ordering of the objects along the vertical axis, this description does not provide information when the subject and the reference are not vertically aligned. To this aim, bounding-box-based features ($f_8 - f_{11}$) exploit the position of the enclosing rectangles of each object to detect the *side, side-up, side-down* labels.

In the following paragraphs we describe the way string-based features are computed. The algorithm extracts column by column the information from the matrix $\mathcal{Z}$, which associates the object identifiers to each pixel. Each column of pixels in $\mathcal{Z}$ is encoded as a vector, then compressed to reduce the computational complexity of the next parts of the algorithm. Specifically, we merge all consecutive pixels with the same object identifier. The resulting compressed column is called *string*,

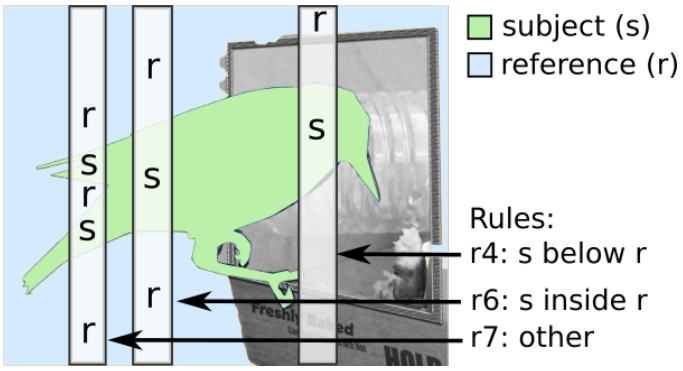


Figure 3.7: String-based feature extraction.

denoted as:

$$\mathcal{Z}_x = \{z_1, \quad \cdots, \quad z_i, \quad \cdots, \quad z_n\}, x \in [1, width(\mathcal{Z})]$$

where x is the horizontal position of the column in $\mathcal{Z}$, z_i are the object identifiers, and $width(\mathcal{Z})$ is the number of columns in the matrix $\mathcal{Z}$. The string representation of the image can be exploited to compute the *string-based* features separately for each object pair s, r. Our technique considers only the strings $\mathcal{Z}_x \in \mathcal{Z}$ that contain both objects s and r. All the other strings are not analyzed, since they do not provide information about the relative vertical ordering of the two objects. At this point, the set of rules described in Table 3.2 are applied to the selected strings for a specific object pair (s, r), with the aim of detecting the object positions column by column. For example, rule r_1 (s *on* r) applies to a string when it contains r that is immediately after s, which entails that the subject is touching the top edge of the reference. Indeed, the elements z_i in $\mathcal{Z}_x$ with lower values of i are located next to the top of the image, while the ones with higher index are close to the bottom. The rules are mutually exclusive (only one rule can be assigned to a string), but different rules may apply to the various strings of the same object pair. An example of the string-based approach is provided in Figure 3.7, where we inspect the relative position between a bird (subject) and a house wall (reference). The image confirms how a specific object pair may have strings applying to different rules.

The final string-based features are then designed to bring the information of the rule-matching process to the position classifier. Specifically, for each rule (*r1-r7*), our technique generates a corresponding feature that counts the number of strings

$\mathcal{Z}_x$ for which the rule applies. We formalize this concept with:

$$f_i = \frac{count(r_i, \mathcal{Z}, s, r)}{min(w_s, w_r)}, i \in [1,7]$$

where f_i is a feature, $count(r_i, \mathcal{Z}, s, r)$ is the number of strings in $\mathcal{Z}$ that contain s, r and satisfy rule r_i, w_s and w_r are the width of the subject and reference respectively. The width of subject and reference objects corresponds to the horizontal pixel span that covers their shape. The count value is divided by the minimum width between the two objects for a normalization purpose. In this way, the output feature represents a percentage of columns (i.e., strings) with respect to the total of the object with the smallest width. Note that other normalization options are not suitable for this computation. For example, dividing by $max(w_s, w_r)$ excessively penalizes these features, giving them very low values when one of the two objects is much larger than the other. Conversely, dividing by the number of strings that contain both s and r would not consider the width of the objects. In this last case, objects that have a high horizontal overlap (i.e., strings where both object appear) relative to their size would be considered in the same way as those with a lower overlap. This behavior could negatively affect a fuzzy decision between *side* (i.e., with low horizontal overlap relative to the object size) and the other classes where the objects are vertically aligned.

String-based features cannot provide information about the vertical position of s and r when these objects are not vertically aligned. This is due to the fact that in that case there are no strings where the two objects appear together. This reasoning justifies the need for integrating the previously mentioned features with bounding-box-based attributes. These features are computed by inspecting the position and size of the object bounding boxes.

Specifically, we first define a set of measures that are exploited to construct the final features:

$$\begin{aligned} dx_1 &= right(r) - left(s), \quad & dx_2 &= left(r) - right(s) \\ dy_1 &= bottom(r) - top(s), \quad & dy_2 &= top(r) - bottom(s) \end{aligned}$$

where the functions $top(), bottom()$ describe the position of the vertical margins of the bounding boxes and $left(), right()$ describe the horizontal ones. Figure 3.8 provides an example of object pair for which we highlighted these quantities. Note

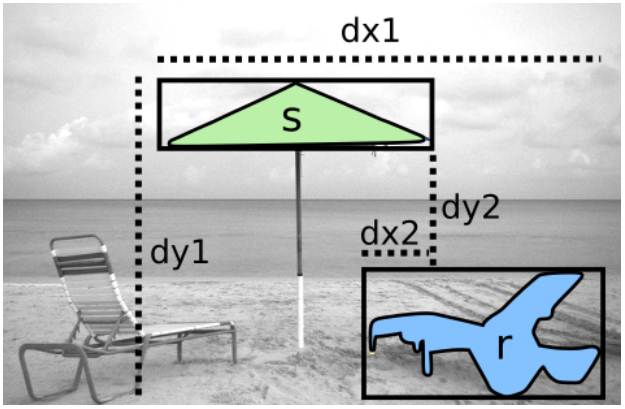


Figure 3.8: Bounding-box-based feature extraction.

how they can be either positive or negative and they consider both the object sizes and the distance between the bounding boxes.

Finally, the four bounding-box-based features are computed by normalizing these values:

$$f_8 = dx_1/max(|dx_1|, |dx_2|), \quad f_9 = dx_2/max(|dx_1|, |dx_2|)$$
$$f_{10} = dy_1/max(|dy_1|, |dy_2|), \quad f_{11} = dy_2/max(|dy_1|, |dy_2|)$$

This normalization is necessary to consider the distances between the two objects in percentage with respect to their size. For example, bigger objects should be separated by a wider space to be considered perceptually distant.

The set of string-based and bounding-box-based features are finally computed for each object pair and used as input by a random forest classifier (see Section 3.4) to decide one among the 9 position labels defined in the beginning of this section.

3.4 Experimental evaluation

In this section we first provide the details on the dataset we designed for training the position classifier. Afterwards, we describe the tuning phase of the final model, showing quantitative results. The labeled dataset, together with the classifier implementation, are openly available in our code repository:

https://github.com/AndreaPasini/SImS

Our dataset with relative position samples was created by manually labeling 700 images from the training set of Microsoft COCO [76]. We designed a labeling tool that selects a random object pair for each image to be annotated. The task of the annotator was to select one among the 9 position labels for the highlighted subject-reference pair. Some examples of annotated images are depicted in Figure 3.9. After

Figure 3.9: Labeled samples from our position dataset.

Classifier	above	around	below	hanging	inside	on	side	side-down	side-up	macro-avg
KNN	0.89	0.93	0.95	0.90	0.92	0.78	0.82	0.89	**0.92**	0.89
RBF-SVC	0.89	0.94	0.92	0.84	0.94	0.82	0.75	0.88	0.85	0.87
Decision tree	0.92	0.95	0.94	0.91	0.93	0.86	0.78	0.92	0.82	0.89
Random forest	**0.93**	**0.96**	**0.98**	**0.92**	**0.95**	**0.93**	**0.85**	**0.94**	0.88	**0.93**
Average	0.91	0.94	0.95	0.89	0.93	0.85	0.80	0.91	0.87	0.89

Table 3.3: F1 score for the pairwise relative position computation.

this process we got 1000 labeled object pairs, from which we extracted a balanced dataset. Specifically, in the end we obtained 540 images including 60 examples for each position label.

The goal of the position classifier is to assign a position label for each object pair. To this aim, it takes as input the features defined in Section 3.3. We inspected the performances of the following list of classifiers available in the scikit-learn library [17]: decision tree, SVM, random forest, naive bayes, KNN. Each model has been evaluated with leave-one-out cross-validation and a grid search methodology. Specifically, for decision trees and random forests we inspected the maximum depth hyperparameter in range 5-35 with step 5. In the case of random forests we tuned the number of estimators in range 10-100 with step 5. Finally, for KNN we set the value of k in 5-15 with step 5. The best configuration for each classifier has been selected by considering the macro-average F1 score among the 9 classes. This grid-search process, including the feature extraction step and the actual classification took 4 hours on the following hardware configuration: Intel Xeon Gold 6140, CPU @ 2.30GHz, RAM 40 GB. The results of the best configurations are provided in Table 3.3. They show that the best macro-average score (0.93) is achieved by a random forest with maxdepth=20 and 35 estimators, while RBF-SVM performs the worst. Moreover, when looking at the average of the classifier scores separately for each class (i.e., last line of Table 3.3), it may be noticed that the *side* label is the one with the lowest value. This is due to the frequent ambiguities between the three positions *side, side-up, side-down*, where sometimes it is difficult to assign a sharp category.

These experiments conclude the analysis of relative object positions. In the next chapters we will exploit the obtained classifier to derive interesting insights from a set of images and enhance their semantic understanding.

Chapter 4

SAD: detecting anomalies in image classification by means of semantic relationships

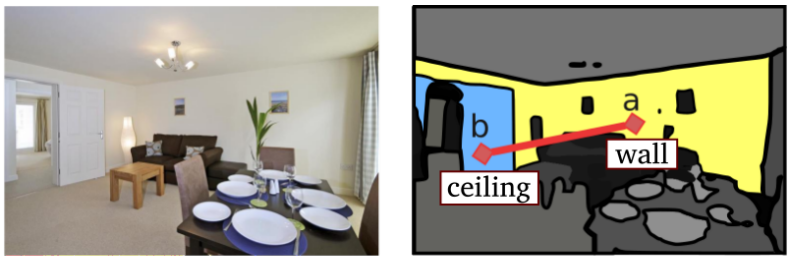
As we have seen in Section 2.2, contextual information is fundamental for the object recognition task. The studies made by Biederman et al. [13], showed that different types of relationships such as object co-occurrence, relative position and relative size, play an important role also for the human brain. In their experiments the authors presented different images to the patients, some of them with objects that violated their usual relationships. For example they proposed objects in unlikely positions (e.g., a couch floating in the air), elements outside their usual context, or objects with a not plausible size when compared to the other scene elements. The experiments showed that the object detection time for the human brain was higher when the images presented these kinds of anomalies. Motivated by these evidences, in our work we exploited different contextual information types to inspect anomalies in the semantic segmentation results generated by artificial neural networks.

Anomaly detection is a well-known data mining task that aims at detecting data that deviates from its expected behavior. This technique finds applications in many safety critical environments, such as intrusion detection in network systems or fraud detection [4, 5]. The identification of outlier documents in textual data [128] or the inspection of anomalies in video sequences are other common applications [77, 79].

In our work we focused on the inspection of anomalies in semantic segmentation (see Section 2.1.5) results by analyzing the *global context* of each object. The anomalies that we detect are related to the semantic meaning of the objects in the image. Hence, we do not consider the visual appearance of the single objects, but we focus their relationships (e.g., co-occurrence, size and position) and their class label.

The experiments of this work aim at detecting anomalies inside semantic segmentation results. Since this technique does not provide the information of the object instances, but only pixel-wise labeling of the image, our methodology distinguishes between objects by looking at connected pixel regions, as we detailed in Section 3.2. Alternatively, instance (or panoptic) segmentation could be used to directly show the different object instances in the image. Nevertheless, we are interested in inspecting the classification errors that typically occur when a model does not take into consideration the semantic analysis of object relationships. For this reason we focused on semantic segmentation, which is instance-agnostic, with the aim of showing why it is more error-prone with respect to instance or panoptic segmentation. Moreover, even if panoptic segmentation is more accurate, it is also much slower when performing the computation and this could sometimes be a motivation for choosing simpler (but more error-prone) semantic segmentation algorithms. Our technique could then be used to inspect anomalies in the segmented images, hence highlighting possible errors and providing an alert to the final user.

Inspired by [13], in this work we model three types of contextual information: (i) *object co-occurrence* to inspect object classes that frequently appear together in the same image, (ii) *relative position* to understand the configuration of the objects in the image, and (iii) *relative size* between objects. The proposed methodology, called *SAD* (i.e., Semantic Anomaly Detection) [88], learns common patterns that characterize the different object classes by analyzing a labeled training set, then builds an interpretable *knowledge base* to store this information. The knowledge base is made of a set of interpretable rules modeling normal data and can be exploited to understand whether an object is anomalous with respect to the context. Objects that deviate from the set of rules in the knowledge base are deemed to be anomalous. Specifically, SAD exploits the collected information to automatically detect anomalies among the labeled objects, highlighting possible classification errors made by the segmentation model. Finally, our methodology can show, in an interpretable way, the pieces of information of the knowledge base that have been



a) query image b) segmented image

Anomaly: wall (a) on ceiling (b) has likelihood <0.01

c) model output

Figure 4.1: Example of anomaly detected by SAD on the ADE20K dataset.

used to detect the anomaly. We exploit this anomaly detection method to pursue the objective of showing that segmentation models that do not make explicit usage of the contextual information (e.g., PSPNet [123]) lack of a complete understanding of the image and may make important errors. Moreover, when ground truth is not available, our methodology can be used to assess the quality of the segmented labeled images.

Figure 4.1 provides an example of anomaly that has been detected by SAD. The query image has been taken from the ADE20K [125] dataset, and automatically labeled by the PSPNet neural network. The segmented image is shown in Figure 4.1b, where we highlighted two objects with yellow and blue. The two objects a and b are assigned to the classes *wall* and *ceiling*, respectively. At this point, if we consider the relative position between a and b, we can see that the yellow patch (a) of pixels is partially on the blue patch (b). This would entail that a *wall* object has been found on a *ceiling* object. From a semantic point of view, this relationship is meaningless, since we are used to see walls supporting ceiling and not viceversa. SAD tries to emulate human reasoning by drawing a set of rules from the knowledge base and inspecting the likelihood of the different object relationships. In this example, it finds that *wall* on *ceiling* is an unlikely situation (probability <0.01), hence it presents an anomaly as output (Figure 4.1c). From this observation it can reasonably be pointed out that the semantic segmentation model has made an error in the classification of either of the two objects. Specifically, the object b should be classified as *door*, instead of *ceiling*. This example shows how the detection of anomalous object configurations allows identifying classification errors and

providing a human readable description of the detected issue.

In general, SAD is interpretable, as it is able to point out the potentially misclassified objects and show the rules that helped in detecting the anomaly. Moreover, even if no anomalies are detected, the likelihood of the object relationships in the image may provide a semantic enrichment of the image description.

Based on these preliminary considerations, the main research contribution within this work is threefold: (i) we exploit semantic object relationships as contextual information to inspect classification anomalies (ii) the generated knowledge base and the detected anomalies are interpretable since they are provided in the form of semantic rules that describe object relationships (iii) the approach is semi-supervised and does not require a training set with ground truth anomalies.

The next sections are organized as follows. Section 4.1 presents related works on anomaly detection. Afterwards, Section 4.2 describes the overview of the SAD approach, Section 4.3 goes in depth with the knowledge base definitions, while Section 4.4 details the anomaly detection algorithm. Finally, Section 4.5 describes the experimental results.

4.1 Related works in the anomaly detection field

In this section we provide an overview of the different anomaly detection algorithms that have been proposed in literature, with focus on highlighting the differences with the proposed Semantic Anomaly Detection.

The anomaly detection tasks can be categorized based on three anomaly types: *point*, *group*, and *contextual* anomalies [4]. *Point anomalies* are defined as those single data points that deviate from the characteristics of the common samples in a specific dataset. For example Zhuang et al. [128] detected outlier documents by means of statistical measures defined on the document topics, while in [113] the authors proposed a clustering-based network anomaly detection method to inspect NetFlow data. Instead, *Group anomalies* identify a set of instances whose global behavior deviate from the expected patterns in a dataset. An example of this task was provided in [79], where the authors inspected anomalous pedestrian motion patterns by means of mixture models defined on dynamic textures. Finally, *Contextual anomalies* highlight instances that are anomalous only within a specific

context. In other words, an instance may be considered as abnormal depending on the contextual information and not only on its own features. SAD leverages on this type of anomalies, as it relates the objects with the contextual information of the whole image.

A second way of categorizing anomaly detection tasks consists in distinguishing them based on the recognition algorithm. Specifically, anomalies can be detected by means of *supervised*, *unsupervised*, and *semi-supervised* methods.

4.1.1 Unsupervised methods

Unsupervised techniques do not require training data specifying which types of samples should be considered anomalous. Among these techniques, clustering is one of the most widely adopted solutions. Specifically, clustering algorithms can be exploited to highlight contextual anomalies by identifying the data points that deviate too much from the modeled clusters.

For example, Hayes et al. [50] identify contextual anomalies in streaming sensor networks by leveraging on clustering. Specifically, they apply the k-Means algorithm to group the different sensors in the network by means of their recorded values and their associated meta-data. After this process, each cluster identifies a sensor profile, whose characteristics are modeled by training a Gaussian predictor. Each Gaussian model is trained on the past history of the values recorded by the sensors belonging to a specific profile. During the anomaly detection phase, the new values streamed by each sensor are analyzed by the Gaussian predictor of the corresponding sensor profile to identify their likelihood. Finally, the values that obtain a low score are deemed to be anomalous within the context defined by the sensor profile.

A second example of contextual anomaly detection based on clustering was provided by Liu et al. in [77]. They analyze time series of satellite images, with the aim of detecting unusual warming and cooling events. The outliers are detected when a pixel or a region is highly different from its spatial-temporal neighbors (i.e., regions that are close both in space and time). To this aim, each image region is described with pixel-level and object-level features. Afterwards, the regions are clustered with an extended Expectation-Maximization algorithm that iteratively aggregates Gaussian clusters. Small groups and isolated elements are deemed to be

outliers, hence possible events that require an action by domain experts.

The main issue with the described techniques is that they are not interpretable, as they only rely on numerical features and provide the final outliers without describing the reasons why they differ from normal data. Differently, our technique makes use of additional semantic information that contributes to obtain a human readable report of the final anomalies.

4.1.2 Supervised methods

Supervised methods rely on training examples to learn modeling both anomalies and normal data. To this aim, they typically exploit classification and regression algorithms. For example, in [10] the authors developed an intrusion detection system that includes both supervised and unsupervised algorithms. Their objective consists in detecting anomalous network packets, by inspecting a set of connection-based features, such as source and destination IPs/ports. Network packets are first analyzed in an unsupervised manner to identify suspicious data. Specifically, each packet is considered as a transaction [3] whose items are represented by connection attributes. The proposed algorithm applies frequent itemset mining on attack-free packets to describe normal instances. Afterwards, it repeats the analysis on other data including attacks in the same time interval. Therefore, the system recognizes suspicious frequent itemsets that are not present in the description of normal data. The packets including the suspicious itemsets can sometimes represent false positives, hence their are manually analyzed by specialized annotators. A supervised classifier is finally trained on the ground truth provided by the annotators to distinguish between real attacks and false positives.

Related to a similar application, the authors in [1] propose to train a decision-tree-like rule based classifier for distinguishing between different types of anomalies, such as DoS attacks, scans and botnets. Also in this case, the classifier is trained by means of hand-labeled data. Similarly, Zhang et al. [122] propose a neural network model to address the same task.

One of the most common issues of supervised methods is the fact that labeled data is not always available. Moreover, they are typically not able to detect new types of anomalies that have not been described in the training data. These kinds of issues can be solved with semi-supervised techniques, as we describe in the next

paragraphs.

4.1.3 Semi-supervised methods

Semi-supervised techniques learn from training data describing only normal instances. Afterwards, they detect anomalies as the samples that deviate from the normal behavior of training set points. These approaches are typically based on statistical methods (e.g., mixture models) that model the probabilistic distribution of the features of normal data. An example of semi-supervised technique is presented in [38], where they identify anomalies in telementry data, which consists of time series recorded by satellites and orbital transfer vehicles. This methodology learns the distribution of normal time series by means of Principal Component Analysis (PCA). Specifically, each time series is first windowed, then described by means of high-dimensional features. Afterwards, the algorithm extracts the direction of the Principal Components related with these training data and compares it with the one of new incoming samples. Anomalies are detected when the difference in direction is above a predefined threshold.

Laxhammar et al. [67] inspect anomalous vessel traffic by modeling normal data with a Probability Density Function (PDF), such as Gaussian Mixture Models or adaptive Kernel Density Estimators. Since there is no information about anomalous data, their PDF cannot be estimated. Hence, the likelihood of new data, computed with the PDF of normal instances, can be used as an indication of the degree to which the observation is normal. To this aim, a threshold for detecting anomalous data must be defined.

The method proposed in [15] exploits a semi-supervised technique in the computer vision field, with a similar goal to the one proposed in our work. Specifically, they detect irregularities of visual data by means of spatial ensembles. Given a segmented query image, they extract a set of representative regions described with visual features. The spatial configuration of the extracted patches is then compared with the examples in a training database. Finally, if a similar configuration cannot be found, the query image is deemed to be anomalous. An important weakness of this work is that the authors do not consider the semantic information of the analyzed images. Indeed, relative positions are computed by means of object centroids, without analyzing discrete values as we propose in SAD. Moreover, they only rely

on visual features and do not interpret objects classes.

Our method relies on semi-supervised algorithms, as well. Similarly to other techniques, we learn the characteristics of normal samples and define anomalies as the instances with a likelihood below a predefined threshold. Interestingly, SAD differentiates from previous works since it describes normal data by means of interpretable semantic rules. The semantic relationships we define among objects are able to derive more abstract information with respect to what could be obtained by simple visual features.

4.2 The Semantic Anomaly Detection approach

We begin to describe SAD with an overview of its main building blocks, depicted in Figure 4.2.

Knowledge base definition. This is the core process of SAD, which is responsible for the semantic information extraction and summarization from a labeled image collection. Images labeled with ground-truth semantic segmentation are first analyzed to identify object instances, then parsed to derive the knowledge base. This is defined as a collection of semantic rules modeling the behavior of *normal data*. As described in Section 4.1, this methodology can be categorized with the semi-supervised anomaly detection approaches.

The knowledge base stores information on common object relationships of the

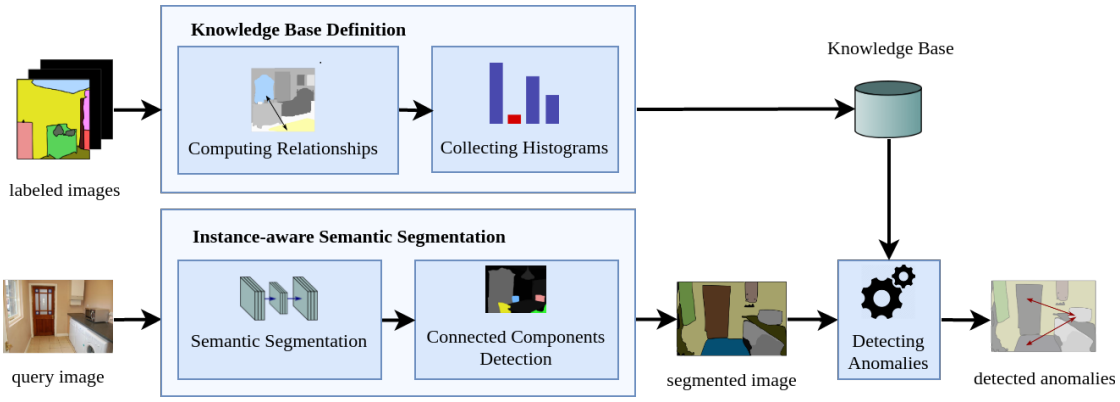


Figure 4.2: The Semantic Anomaly Detection process.

following three types: (i) object co-occurrence, (ii) relative position, and (iii) relative size. The collected rules are stored in the form of histograms, representing the probability distributions of the different object configurations. Section 4.3 provides the details on the knowledge base definition and its extraction phase.

Instance-aware Semantic Segmentation. This step of the process is responsible for the generation of automatically labeled images that will be evaluated by the anomaly detection process. It is composed of two blocks: (i) Semantic image segmentation and (ii) Connected components detection. The semantic image segmentation model can be chosen among those available in literature, and it represents the model being evaluated by SAD. In the experimental section of this work we chose *PSPNet* [123], which was winner of the ImageNet segmentation challenge in 2016. However, other deep learning models for semantic segmentation [8, 22, 120] can be used interchangeably.

The connected components detection phase is instead designed to highlight the different object instances starting from the semantic segmentation result. As we have described in Section 3.2, this step looks for the different object instances by applying a BFS algorithm that detects patches of adjacent pixels with the same class. Each patch of pixels (i.e., connected component) identifies a distinct object instance. The final result is encoded in the *panoptic segmentation format*, to store both object classes and instance identifiers.

Anomaly Detection The anomaly detection phase takes as input a query image, whose objects have been classified, and the previously extracted knowledge base. The objective is the inspection of possibly misclassified objects by identifying anomalies in their configuration, according to the likelihoods stored in the knowledge base. The details of this process are provided in Section 4.4, while the effectiveness of the SAD methodology is shown in Section 4.5.

Once an anomaly is detected, the anomalous configuration could be used to improve the knowledge base with an active training process. For example, when the same type of anomaly is found many times during inference, the system could start to consider that object configuration as normal and add it into the knowledge base. However, this process may not work properly when the neural network under evaluation tends to make repeating errors, which could be erroneously considered as normal after some iterations. To avoid this drawback, the anomalies that occur

Table 4.1: Properties and associated categories.

Category	Properties
position	*above, below, on, hanging, inside, around, side-up, side, side-down*
width	*bigger, same, smaller*
height	*bigger, same, smaller*
area	*bigger, same, smaller*
co-occurrence	*co-occurs*, $\neg$*co-occurs*

many times and could be used to improve the knowledge base should be manually evaluated. We reserve these considerations and analyses for future developments of our technique.

4.3 Knowledge Base Definition

As anticipated in Chapter 4, our knowledge base includes different types of global contextual information: (i) relative object position, (ii) relative object size, and (iii) object co-occurrence. We modeled these three types of information with the same notation, to make the knowledge base more uniform and easily interpretable. In the following we provide the definitions (i.e., *property, triplet,* $\mathcal{T}$*-histogram*) useful to describe the patterns collected in the knowledge base.

Definition 1 (Property)**.** *A property p describes the relationship between two objects inside the same image. Each property belongs to a specific category c.*

Categories are defined to represent the different types of relationships that SAD is able to model in the knowledge base. Some examples are the *relative position* or the *co-occurrence*. The full list of properties, associated with their categories, is provided in Table 4.1.

The properties and categories defined so far are exploited to describe the semantic relationships between objects of different classes within the same image. Each relationship is represented with a data structure we call triplet.

Definition 2 (Triplet)**.** *Let* s_l *(subject label) and* r_l *(reference label) be two object classes. Let c be the category of a property. A triplet* $\mathcal{T} = \langle s_l, c, r_l \rangle$ *describes the*

relationships for each property of category c between two objects of class s_l and r_l.

This permits inspecting different types of relationships between a specific pair of object classes (s_l, r_l). The two variables (s_l, r_l) represent object classes and must not be confused with object instances. Indeed, the objective of our knowledge base is to describe the normal behavior of objects with specific classes, without referring to a specific image. To make an example, if we want to describe the relative position between *bottle* and *table* we will use $\mathcal{T} = \langle$*bottle, position, table*$\rangle$.

The last concept useful to define the knowledge base is represented by histograms. Each triplet is uniquely associated with a histogram ($\mathcal{T}$-histogram) that quantitatively describes the likelihood of the different properties (e.g., "above", "below") of the triplet category (e.g., "position") for the specified class pair (s_l, r_l).

Definition 3 ($\mathcal{T}$-histogram). *Let $\mathcal{T} = \langle s_l, c, r_l \rangle$ be a triplet. A $\mathcal{T}$-histogram $h(\mathcal{T})$ is given by*

$$h(\mathcal{T}) = [l(p_1), \ldots, l(p_i), \ldots, l(p_N)]$$

where each value $l(p_i)$ specifies the likelihood that the subject s_l and the reference r_l of $\mathcal{T}$ satisfy a particular property p_i of category c, and N is the number of properties belonging to category c.

The different likelihoods in a $\mathcal{T}$-histogram represent a discrete probability distribution, as they are constrained in range [0,1] and add up to one. They are computed from the training data during the knowledge base definition step by inspecting the images that contain the two objects with class s_l, r_l. For example, the triplet $\mathcal{T}_1 = \langle$*bottle, position, table*$\rangle$ and the histogram $h(\mathcal{T}_1) =$ [l(*below*)=0.90, l(*side-down*)=0.1, l(*above*)=0.0, ...] indicate that bottles are 90% of the time on the table. Properties with 0 likelihood in the histogram are omitted for the sake of brevity.

The knowledge base KB is finally modeled as the set containing the extracted histograms:

$$KB = \{h(\mathcal{T}) \mid \mathcal{T} \in L \times ||C|| \times L\}$$

where $L \times ||C|| \times L$ is the set of possible triplets that can be generated, being $||C||$ the number of property categories and L the number of object classes.

Algorithm 1 SAD: knowledge base definition

Input: Labeled images $\mathcal{D}_I$
Output: Knowledge base histograms KB

1: $KB = \{\}$
2: **for all** I in $\mathcal{D}_I$ **do**
3: $(\mathcal{L}, \mathcal{Z}) = connectedComponents(I)$
4: **for all** (s, s_l, r, r_l) **in** $objectPairs(\mathcal{L}, \mathcal{Z})$ **do**
5: **for all** $c \in C \setminus \{co\text{-}occurrence\}$ **do**
6: $p = computeProperty(s, c, r, \mathcal{Z})$
7: $\mathcal{T} = \langle s_l, c, r_l \rangle$
8: $h(\mathcal{T}) = getOrCreateHistogram(KB, \mathcal{T})$
9: $increment(h(\mathcal{T}), p)$
10: **end for**
11: **end for**
12: $updateCo\text{-}occurrence(KB, \mathcal{Z})$
13: **end for**
14:
15: **for all** $h(\mathcal{T})$ in KB **do**
16: $h(\mathcal{T}) = h(\mathcal{T})/sum(h(\mathcal{T}))$
17: **end for**
18: **return** KB

The pseudo-code for the computation of the knowledge base is provided in Algorithm 1. The algorithm takes as input the ground-truth dataset of segmented images $\mathcal{D}_I$ and generates as output the set of histograms in KB. From the computational point of view, the algorithm has complexity $O(||D_{\mathcal{I}}|| \times ||P|| \times ||C||)$, where $||D_{\mathcal{I}}||$ is the number of dataset images, $||P||$ is the average number of object pairs in each image, and $||C||$ is the number of property categories. Line 1 initializes an empty set that is going to be filled in the next iterations over the different images (line 2). From each image the connected components are extracted (line 3, see Section 3.2 for details), generating the object instances in the panoptic format $(\mathcal{L}, \mathcal{Z})$, where $\mathcal{L}$ is the matrix containing class labels and $\mathcal{Z}$ is the matrix containing object identifiers (see Section 2.1.5). The algorithm continues by iterating on all the possible object pairs in the image (line 4), where s, r are the object identifiers and s_l, r_l are their respective class labels. Given the specified object pair, SAD operates separately for all the different categories c (line 5). Specifically, it computes the property value for s, r by inspecting the object pixels inside the matrix $\mathcal{Z}$ (line 6). Note that the *co-occurrence* category is not computed at this stage

since it requires auxiliary information about the number of times that s and r do not appear together. This specific property is updated separately at the end of the for loop (line 12). Details on property computation, separately for each type, are provided in the next sections.

Afterwards, the algorithm prepares the triplet $\mathcal{T}$ (line 7) and retrieves the associated histogram, if present, from the knowledge base (line 8). If the histogram is not available in KB, then it is created and initialized as a void histogram. Subsequently, SAD increments the count of the property p in the histogram $h(\mathcal{T})$ (line 9). This operation has the objective of counting the number of object pairs with class l_s, l_r that satisfy the property p. Since the resulting histograms should model discrete probability distributions, the final step (line 15) consists in normalizing the histogram counts. Specifically, for each histogram $h(\mathcal{T}) = [l(p_1), \ldots, l(p_i), \ldots, l(p_N)]$, the algorithm divides its values $l(p_i)$ by their sum. In this way, each $l(p_i)$ is the conditional probability of satisfying property p_i, given the two object classes s_l, r_l.

After this process, the knowledge base will approximately contain $L \times L \times C$ histograms, where L is the number of object classes and C is the number of categories defined in Table 4.1. However, not all the histograms represent relevant information due to (i) an insufficient number of training samples for a given object pair, or (ii) the absence of preferred properties that are satisfied by the two object classes. Hence, we reduce the knowledge base size by filtering the most relevant information. The histograms that are kept in the knowledge base are those that satisfy the following two criteria: (i) they are supported by a minimum number of training pairs, and (ii) they show likelihood distributions modeling the concepts *always* and *never*. According to the former, the algorithm keeps in the knowledge base only the histograms that satisfy the following condition:

$$support(h(\mathcal{T})) \geq minsup$$

where $support(h(\mathcal{T}))$ indicates the number of training pairs used to derive the histogram and $minsup$ is a threshold, whose value is discussed in Section 4.5.

Following the second criterion, a histograms is deemed to be relevant when it presents an unbalanced probability distribution, which means that the object pair tends to satisfy some particular properties, but not the others. For example, the histogram *[l(above)=0.99, l(below)=0.0, l(inside)=0.01, ...]* models an object pair

where the subject is always above reference, while the histogram *[l(above)=0.4, l(below)=0.0, ...]* specifies that the subject is never below the reference. Hence, the histograms satisfying the presented criterion should describe meaningful relationships such as "*tree is never below river*" or "*carpet is always on the floor*". To identify these patterns, we only select the histograms that satisfy at least one of the following constraints:

(i) $h(\mathcal{T})$ contains only one likelihood $l(p_i) > thr_h$

(ii) $h(\mathcal{T})$ contains at least one likelihood $l(p_i) < 1 - thr_h$

where thr_h specifies a tolerance threshold, whose value is discussed in Section 4.5. The first constraint identifies the histograms that present a property with a very high likelihood (i.e., modeling the *always* concept). The second one identifies those that present one or more low likelihoods modeling the *never* concept.

In the following sections we describe in details the different types of properties and the way they are computed by SAD from the segmented images. As specified in Algorithm 1, these properties are computed by means of the *computeProperty()* and *updateCo-occurrence()* methods, that take as input the segmented image in the panoptic segmentation format $(\mathcal{L}, \mathcal{Z})$, generated from semantic segmentation with the method *connectedComponents(I)*.

4.3.1 Object Positions

The first type of contextual information modeled in this work is relative position. The different properties that can be assigned to this category are listed in Table 4.1. To derive the relative position between objects, we exploit the classifier that we designed in Section 3.3. The model takes as input the image encoded with the panoptic segmentation format $(\mathcal{L}, \mathcal{Z})$ and assigns a position label to the specified object pair (s, r). More formally, we can use the notation:

$$computeProperty\ (s, position, r, \mathcal{Z}) = applyPositionClassifier\ (s, r, \mathcal{Z})$$

4.3.2 Object Sizes

The second relationship category we consider is the relative size between objects in the same picture. We define three different categories for describing the

relative size (i.e., *width, height* and *area*), which can assume the following property values: *bigger, same, smaller* (see Table 4.1). In the following we detail how these properties are assigned to an object pair in the picture under analysis.

Size is evaluated separately for width and height, since these two categories could highlight interesting object relationships. For example the algorithm may learn of objects that are taller than others (e.g., a tree with respect to a bush) and others that are wider (e.g., a bench with respect to a chair). Width and height are computed by comparing the size of the object bounding boxes in percentage with respect to the image size.

Let s, r be two object identifiers in the panoptic segmentation matrix $\mathcal{Z}$. SAD assigns (*computeProperty()* function, used in Algorithm 1) the property values for the category *width* according to the following criteria:

$$computeProperty\ (s, width, r, \mathcal{Z}) = \begin{cases} bigger, & if\ w(s)/w(r) > 1 + thr \\ smaller, & if\ w(r)/w(s) > 1 + thr \\ same, & otherwise \end{cases}$$

where $w(s)$ and $w(r)$ are the width (in percentage) of the bounding boxes of s and r, and thr represents a tolerance value for distinguishing between the three cases.

For example, if the subject s is 80% wider than the reference r and $thr = 0.75$, then the computed property will be *bigger*. In the experimental evaluation we inspected that variations of thr in the range between 0.7 and 0.9 do not significantly affect the results of the anomaly detection phase.

The relative *height* is computed in a very similar way to width:

$$computeProperty\ (s, height, r, \mathcal{Z}) = \begin{cases} bigger, & if\ h(s)/h(r) > 1 + thr \\ smaller, & if\ h(r)/h(s) > 1 + thr \\ same, & otherwise \end{cases}$$

where $h(s)$ and $h(r)$ are the height (in percentage) of the bounding boxes of s and r.

The last relative size category, the *area*, has been defined to inspect the space occupancy of the objects within their bounding box. Specifically, some objects may

have a very large bounding box, but only cover a small percentage of area inside it. For example objects such as a "ladder", or a "fence", which are characterized by the presence of holes in their body, present a lower value of area even if the bounding box covers a large part of the image.

Based on this definition, the area relationship is computed with:

$$computeProperty\ (s, area, r, \mathcal{Z}) = \begin{cases} bigger, & if\ area(s)/area(r) > 1 + thr_a \\ smaller, & if\ area(r)/area(s) > 1 + thr_a \\ same, & otherwise \end{cases}$$

where:

$$area(\star) = \frac{\#pixels\ of\ \star}{imageWidth \cdot imageHeight}, \quad \star \in \{s, r\}$$

The denominator of $area(\star)$ has the purpose of making the measure independent of the image size. Similarly to the relative size, the threshold thr_a ranges between 0.7 and 0.9.

4.3.3 Co-Occurrence

The last way of modeling global context for a given image is object co-occurrence. When analyzing the class label for a specific object, this should be coherent with those in the surroundings. For example inside an indoor scene with objects such as "table" and "sink", we may expect other indoor objects, while we would be surprised in discovering things like "zebra" or "boat". To adapt this reasoning to our methodology, we consider again object pairs and analyze the likelihood they should appear in the same picture. Hence, the co-occurrence probability can be exploited to detect anomalous objects in the segmentation results.

Before analyzing our approach for inspecting co-occurrence, we first review the most common measures of association between two variables. In statistics, *correlation* allows inspecting the relationships between two numerical variables.

One of the most famous correlation coefficients is the Pearson's one [16]. Let X, Y be two random variables with variance σ_X, σ_Y, respectively. Pearson's correlation is defined as:

$$corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

where $cov(X, Y)$ is the covariance between the two variables. Pearson's score takes values in range $[-1, 1]$, where 1 indicates a perfect direct linear relationship, while -1 indicates a perfect inverse linear relationship. Values close to 0 entail low correlation. This coefficient is suitable for continuous variables, hence it does not represent a proper choice for our approach that aims at inspecting correlations between object classes in the images.

For the discrete case, the *chi-squared* test evaluates how likely two variables present different values by chance [16]. Let X, Y be two categorical random variables X, Y. The chi-squared test computes the p-value for the null hypothesis $H0$ that there is no relationship between the two variables. The first step to obtain chi-squared statistics is creating a matrix $O_{i,j}$ that specifies in each cell the number of samples with a specific pair of values $X = x_i, Y = y_j$. Afterwards, a second matrix $E_{i,j}$ is created, where each cell contains the expected number of samples we would obtain if X and Y were statistically independent. Chi-squared statistics are then computed by comparing the actual values and the expected ones: $\chi^2 = \frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$. The output chi-squared score is given by the sum of all the elements in χ^2. Lower values entail that X and Y are not correlated. Finally, the analysis can be completed by computing the p-value [16] for χ^2 to understand whether $H0$ can be refused (i.e., X and Y present a correlation). In our case study, modeling class co-occurrence with chi-squared could be implemented by assigning to X and Y the labels of object class pairs that co-occur in the same image. However, the output of the test would not be sufficiently informative, as it could state whether there is a correlation between the class pairs, but it would not be able to detail which of them are the most correlated (e.g., deriving that car is correlated with road since they co-occur many times).

Mutual information is another correlation metric that can be computed for two categorical variables. It measures the amount of information shared by the two variables X, Y. This metric is again not suitable for our analysis, since it presents the same limitations cited for chi-squared.

Frequent itemset mining approaches [3] implement interesting ways of analyzing correlations between discrete items. These techniques work with the concept of *transaction*, which is defined as a collection of discrete items (e.g., 'a, b, c'). Multiple transactions can be collected together to obtain a transactional database,

which is suitable to be analyzed by frequent itemset mining algorithms. This research field introduced metrics such as *confidence* and *lift*, designed to inspect the correlation between items inside the transactions (e.g., detect which items are most frequently bought together in a supermarket).

Let a, b be two items occurring in a transactional database. Confidence [112] is defined as the conditional probability $P(a|b) = \#(a,b)/\#(b)$, where $\#(a,b)$ is the number of transactions including both a and b, while $\#(b)$ is the number of transactions containing b. This probability value allows inspecting how it is likely to have a in a transaction, if b is present. Hence, it can be interpreted as a measure of correlation between two different item types.

To apply this reasoning to our class co-occurrence problem, we model each picture as a transaction and the set of contained object labels as the items. When a picture contains more instances with the same class, the object is represented with just one item in the transaction. By computing confidence between two class labels s_l, r_l (subject and reference), we can inspect the likelihood of finding one object given the presence of the other in the same image.

Lift [112] is a second well-known metric in the itemset mining field. It returns a measure of correlation between two items. It is defined as:

$$lift(a,b) = \frac{P(a \wedge b)}{P(a)P(b)} = \frac{P(a|b)}{P(a)}$$

It takes values greater than 1 when the two items are positively correlated (the presence of one of them encourages the presence of the other). When lift is 1, a and b are independent, while for values < 1 the presence of one item discourages the presence of the other. Differently from confidence, which inspects the conditional probability of one item with respect to the other, lift allows understanding negative correlations (when it is < 1). For example we may expect that object classes like “table” and “chair” would increase the probability of having “bottle”, but would decrease instead the likelihood of “boat” or “grass”. However, as a drawback, lift is symmetric. Indeed, in the case of positive or negative correlation, this metric cannot specify the causality between the two items (e.g., whether it is a that encourages/discourages the presence of b or vice-versa).

Due to the limitations of the presented metrics, in our work we exploit the

Certainty Factor (CF) measure to model co-occurrence probability. This metric was presented for the first time in the expert system MYCin [53, 102]. The Certainty Factor is asymmetric (it allows understanding the causality of the relationships between the two classes a, b) and it allows inspecting both positive and negative correlations. These two properties are not satisfied together by the previously mentioned metrics.

The certainty factor is defined as the difference between the Measure of Belief (MB), which specifies the increase of probability of having s_l in a transaction containing r_l, and the Measure of Disbelief (MD), which specifies the decrease of probability of s_l given r_l.

Let s_l, r_l be a class pair. Its associated CF is defined as [102]:

$$CF(s_l, r_l) = \begin{cases} \dfrac{P(s_l|r_l) - P(s_l)}{1 - P(s_l)}, & \text{if } P(s_l|r_l) > P(s_l) \\ \\ \dfrac{P(s_l|r_l) - P(s_l)}{P(s_l)}, & \text{otherwise} \end{cases}$$

The value of CF ranges between -1 and $+1$. When $P(s_l|r_l)$ is greater than $P(s_l)$, the presence of r_l increases the likelihood of s_l. On the contrary, if $P(s_l|r_l) < P(s_l)$ then r_l discourages the presence of s_l. In the first case, the positive likelihood increment (i.e., $P(s_l|r_l) - P(s_l)$), is normalized with the probability of not having s_l in the transaction (i.e., $1 - P(s_l)$). This factor helps to increase the module of the (positive) certainty factor when s_l already has a high probability of being found in the transactions, hence even a small numerator should be considered as meaningful. In the second case, the negative likelihood increment is normalized with the probability of having s_l in the transaction. This time, the normalization increases the module of the (negative) certainty factor when s_l has a low probability of being found in the transactions. Hence, if $P(s_l)$ is small, even a small decrease of probability caused by r_l should be treated as meaningful.

The certainty factor is computed from the training images by evaluating the support count of each item pair (i.e., the number of transactions in which it is contained), denoted as $\#(s_l, r_l)$. We also collect the support counts of the items separately, i.e., $\#(s_l), \#(r_l)$, then compute the conditional probabilities required for

the CF. The different counts are updated with the function *updateCo-occurrence()* in Algorithm 1 (line 12).

The choice of using the CF for modeling co-occurrence is due to its asymmetric behavior for the two classes, differently from lift (or correlation) defined for association rules [112]. Another commonly used metric is confidence [112] (i.e., $P(s_l|r_l) = \#(s_l, r_l)/\#(r_l)$) that, similarly to CF, is not symmetric, but does not relate the presence of one class to the absence of the other.

After computing the CF for all the possible class pairs, class co-occurrences are modeled in the knowledge base with the triplets $\mathcal{T} = \langle s_l, \textit{co-occurrence}, r_l \rangle$ and the following histogram:

$$\begin{aligned} h(\mathcal{T}) = & [l(\textit{co-occurs}) = CF_{norm}, \\ & l(\neg \textit{co-occurs}) = 1 - CF_{norm}] \end{aligned}$$

where CF_{norm} is the certainty factor normalized between 0 and 1. To normalize CF, which ranges in [-1, 1], we exploited the min-max normalization equation:

$$CF_{norm} = (CF(s_l, r_l) - min(CF))/(max(CF) - min(CF)) = (CF(s_l, r_l) + 1)/2$$

The generated histograms are finally stored in the knowledge base, as specified in Algorithm 1.

4.4 Anomaly Detection

The main goal of the SAD approach consists in detecting potentially misclassified objects from segmented images. Anomalous objects are highlighted when they are involved in at least a relationship that does not satisfy the configuration rules stored in the knowledge base. To avoid false positives in the anomaly detection phase, we also introduce the concept of *supporter*. Supporters represent object pairs which are strongly consistent with at least one configuration rule in the knowledge base.

The method for extracting anomalies and supporters is shown in Algorithm 2. The input of the procedure consists of a segmented image $\mathcal{I}$ to be analyzed and the knowledge base (KB) that associates histograms with triplets, while the output is the set of anomalies and supporters detected in the analyzed image. The complexity

Algorithm 2 SAD: anomaly detection

Input: Segmented image $\mathcal{I}$, knowledge base histograms KB
Output: Set of anomalies and supporters $\mathcal{A}n, \mathcal{S}up$

1: $\mathcal{A}n = \{\}$
2: $(\mathcal{L}, \mathcal{Z}) = connectedComponents(I)$
3: **for all** (s, s_l, r, r_l) **in** $objectPairs(\mathcal{L}, \mathcal{Z})$ **do**
4: **for all** c **in** C **do**
5: $p = computeProperty(s, c, r, \mathcal{Z})$
6: $\mathcal{T} = \langle s_l, c, r_l \rangle$
7: $h(\mathcal{T}) = getHistogram(KB, \mathcal{T})$
8: $l(p|\mathcal{T})$ = getLikelihood($h(\mathcal{T})$, p)
9: $l(\neg p|\mathcal{T})$ = 1 - $l(p|\mathcal{T})$
10: **if** $l(\neg p|\mathcal{T}) > thr_h$ **then**
11: $\mathcal{A}n = \mathcal{A}n \cup$ anomaly($s, r,\ conf = l(\neg p|\mathcal{T})$)
12: **end if**
13: **if** $l(p|\mathcal{T}) > thr_h$ **then**
14: $\mathcal{S}up = \mathcal{S}up \cup$ supporter($s, r,\ conf = l(p|\mathcal{T})$)
15: **end if**
16: **end for**
17: **end for**
18: **return** $\mathcal{A}n, \mathcal{S}up$

of the anomaly detection algorithm is $O(||P|| \times ||C||)$, where $||P||$ is the number of object pairs in the image and $||C||$ is the number of property categories.

The first operation consists in preparing the object instances in the panoptic segmentation format (line 2). Afterwards, SAD iterates on all the object pairs and all the property categories (lines 3, 4). For the selected pair and category, it computes the associated property and triplet (lines 5, 6), as discussed in Section 4.3. At this point, the knowledge base is exploited to inspect the likelihood of the computed property for the selected triplet $\mathcal{T}$, with classes s_l, r_l and category c (lines 7, 8). Next, it complements the likelihood to generate $l(\neg p|\mathcal{T})$, which represents the probability of having an object pair with classes s_l, r_l that does not apply to property p. When this likelihood is greater than the threshold thr_h (defined in Section 4.3), an anomaly is detected for the object pair s, r with confidence $conf = l(\neg p|\mathcal{T})$ (lines 10, 11). Indeed, anomalies have higher confidence when the likelihood of p is lower. As opposite concept to anomalies, a supporter between two objects has confidence $conf = l(p|\mathcal{T})$ and it is selected only if this value is greater than thr_h (line 14).

After the detection of anomalies and supporters, SAD exploits them to label each object as *normal* or *exception* with respect to the contextual information. This binary classification task aims at demonstrating that the detected object relationships can help in the detection of misclassified objects in the segmented image. We inspect three approaches: (i) the *Anomaly-Only Method*, (ii) the *Delta Method*, and (iii) the *WTA method*. All of them are experimentally evaluated in Section 4.5. The general principle of the proposed methods relies on the hypothesis that objects with many anomalies and few supporters will more likely present a classification error performed by the neural network.

4.4.1 Anomaly-Only method

This method assigns the exception class to all the objects which are either the subject or the reference of at least one contextual anomaly in $\mathcal{A}n$. More formally, let z be an object identifier in $\mathcal{Z}$ and $\mathcal{A}n_z$ the set of anomalies for which z is either the subject or the reference. Its anomaly detection label is defined with:

$$label(z) = \begin{cases} exception, & \text{if } ||\mathcal{A}n_z|| > 0 \\ \\ normal, & \text{otherwise} \end{cases}$$

As discussed in Section 4.5, this method shows a high recall for the exception class, but a lower precision.

4.4.2 Delta method

We design a second methodology to detect misclassified objects, which considers both anomalies and supporters. Let z be the object under analysis, $\mathcal{A}n_z, \mathcal{S}up_z$ the set of anomalies and supporters for which z is either the subject or the reference. The *Delta* method computes a score for the object z in the following way:

$$score(z) = \sum_{sup \in \mathcal{S}up_z} conf(sup) - \sum_{an \in \mathcal{A}n_z} conf(an)$$

where $conf(sup)$ and $conf(an)$ are the confidence scores of supporters and anomalies, respectively. The higher the score (i.e., the supporters are stronger than the anomalies), the most likely is the fact that z has been given the correct label during

Algorithm 3 WTA Method

Input: Anomalies and supporters $\mathcal{A}n, \mathcal{S}up$, object identifier matrix $\mathcal{Z}$
Output: Labeled objects *normal*, *exception*

1: $scores = \{\}$, $normal = objects(\mathcal{Z})$, $exception = \{\}$
2: **for all** z in *normal* **do**
3: $\quad scores(z) = \sum_{sup \in \mathcal{S}up_z} conf(sup) - \sum_{an \in \mathcal{A}n_z} conf(an)$
4: **end for**
5: **while** $||\mathcal{A}n|| > 0$ **do**
6: $\quad z' = \arg\min_{z \in \mathcal{Z}} scores(z)$
7: $\quad \mathcal{A}n = removeAnomalies(\mathcal{A}n, z')$
8: $\quad exception = exception + z'$
9: $\quad normal = normal \setminus z'$
10: **end while**
11: **return** *normal*, *exception*

semantic segmentation. Hence, the final anomaly label is given based on whether the obtained score is positive:

$$label(z) = \begin{cases} exception, & \text{if } score(z) < 0 \\ \\ normal, & \text{otherwise} \end{cases}$$

4.4.3 WTA method

The *WTA* (Worst Take All) method is designed to prevent accurate objects from being labeled as exception. Indeed, since anomalies are defined between object pairs, it is possible that only one of the two objects has been misclassified, while the other one is correct. Assigning the exception label to both objects could generate in the result many false positives. Differently from the Delta method, WTA tends to penalize only the worst object which is involved in each pairwise anomaly, reducing the number of false exceptions.

The pseudocode of this method is presented in Algorithm 3. In the beginning, the set *normal* is initialized with all the object identifiers in $\mathcal{Z}$ (line 1) and *exception* is prepared as an empty set. During the process, some of the objects in the *normal* set will be moved to *exception*. Firstly, the system computes the scores of the objects in *normal* as specified for the *Delta method* (lines 2, 3). After this step, a

while loop iterates until $\mathcal{A}n$ is empty. At each iteration the object z' with the lowest score (i.e., the most anomalous) is selected and all the anomalies where z' is either the reference or the subject are removed from the set $\mathcal{A}n$ (lines 6, 7). Afterwards, z' is moved from *normal* to the *exception* set. The loop continues until all the anomalies in $\mathcal{A}n$ have been removed. In the end, the sets *normal* and *exception* are returned by the algorithm as the final labeled objects.

4.5 Experimental evaluation

This section describes in details the experimental results obtained with the application of the SAD methodology. We divide the analysis into four subsections, providing (i) the description of the selected dataset, (ii) the insights that can be derived from the extracted knowledge base, (iii) the anomaly detection results, and (iv) the lessons learned. The implementation of SAD is openly available in our code repository: https://github.com/AndreaPasini/SAD2019

4.5.1 Dataset

We extracted the knowledge base and performed the anomaly detection experiments on the MIT Scene Parsing Benchmark [84]. Its data samples have been generated starting from the *ADE20K dataset* [125]. The selected benchmark is a rich collection of *20,000* training samples and *2,000* test images, depicting both indoor and outdoor sceneries. These pictures are pixel-wise annotated, highlighting objects from 150 different classes. Some pixels of the labeled images are not assigned to a class, due to unrecognizable objects (e.g., occlusion issues) and to the presence of some objects whose class is not included in the predefined set of labels for this dataset. These missing regions are ignored by SAD, as specified in the challenges built on the Scene Parsing Benchmark. In our experiments, we extract the knowledge base from the training samples and we apply SAD anomaly detection to assess the behavior of a deep convolutional neural network applied to the test images. Specifically, the convolutional neural network studied in this work is the pre-trained PSPNet model [123].

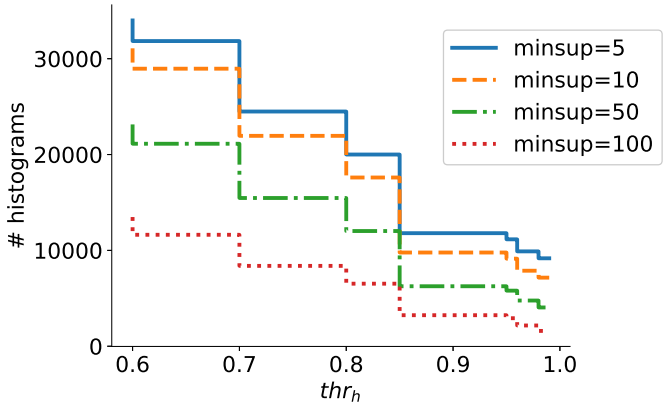


Figure 4.3: Histograms for different *minsup* and thr_h values.

4.5.2 Knowledge base analysis

In Section 4.3 we defined the knowledge base and its main components. We also specified that not all the histograms in the knowledge base are kept, but the *minsup* filter (i.e., minimum support) and thr_h (i.e., threshold to identify *always* and *never* histograms) are applied to select the relevant information. Figure 4.3 shows the number of resulting histograms while varying these two parameters. On the training set of the MIT Scene Parsing Benchmark, we obtained up to 34,000 histograms for $minsup = 5, thr_h = 0.6$. Their number diminishes to about 13,000 for $minsup = 100, thr_h = 0.6$. Besides, the amount of histograms decreases with increments of thr_h. For example, with $minsup = 5$ and $thr_h = 0.99$ we get about 10,000 histograms in the result.

We further analyze the number of histograms separately for the different property categories defined in Section 4.3. Figure 4.4 provides this information for thr_h in range $[0.6, 0.99]$. With the setting $thr_h = 0.6, 0.7$, the histograms of the different categories are balanced in number. This means that all of the defined property categories are useful to generate meaningful histograms describing the training data. By increasing the threshold value to 0.99, the final number of histograms becomes unbalanced instead. Specifically, the *position* category presents a very high amount of histograms (approximately 5,500), which means it is more reliable and informative than the other categories (presenting less than 500 histograms each).

We continue our analysis of the knowledge base by showing some qualitative examples of its content. Table 4.2 reports the values of the Certainty Factor (CF)

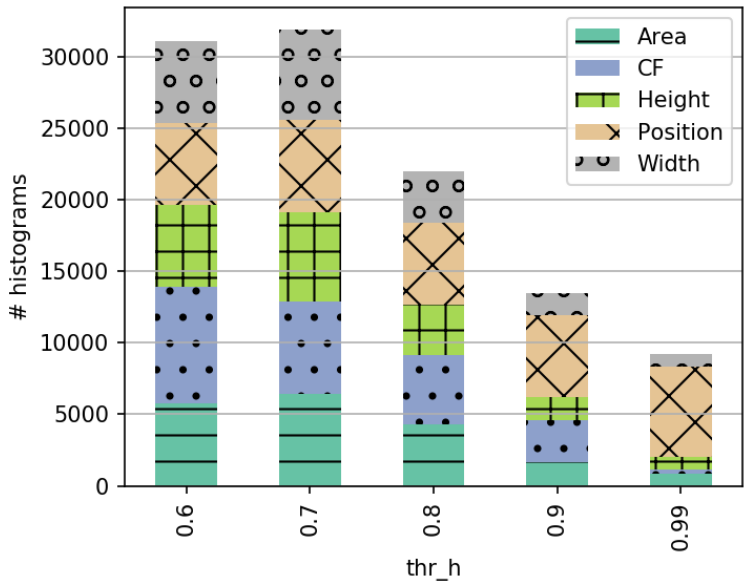


Figure 4.4: Histograms for each category. *minsup*=10.

Table 4.2: Certainty Factor examples.

Class Pair	CF	Class Pair	CF
wall, oven	1.00	sky, microwave	-0.99
wall, sink	0.99	cabinet, road	-0.99
floor, sofa	0.96	sofa, car	-0.99
bed, pillow	0.94	sky, countertop	-0.99
building, sidewalk	0.93	floor, hill	-0.98
sky, mountain	0.91	lamp, river	-0.98

associated to some of the object pairs that can be found in the training set. On the left part of the table, some examples of positive CF are provided. We remind that a positive CF implies class pairs that are positively correlated. More specifically, the presence of the reference in an image entails a higher probability of the subject in the same picture. Consider for example the first class pair (i.e., *wall-oven*). It is associated with a very high CF, which indicates that every time the "oven" object is found in an image, also "wall" will occur. Indeed, this specific object pair identifies frequent indoor scenes included in the dataset. Since the Certainty Factor metric is not symmetric, $CF(oven, wall)$ takes a different value: 0.008. To justify this result, we can reason on the fact that the presence of a "wall" does not always entail

Table 4.3: Area relationship examples.

Class Pair	Sup	Histogram
plate, swivel chair	25	*bigger*=0.00 *same*=0.00 *smaller*=1.00
light, microwave	378	*bigger*=0.02 *same*=0.06 *smaller*=0.92
runway, van	20	*bigger*=0.95 *same*=0.05 *smaller*=0.00
painting, pool table	271	*bigger*=0.03 *same*=0.04 *smaller*=0.94

Table 4.4: Position relationship examples.

Class Pair	Sup	Histogram
runway, sky	151	*below*=0.87 *side-down*=0.1
ball, pool table	33	*inside*=0.91 *above*=0.03
light, sink	1321	*side-up*=0.83 *above*=0.17
armchair, cradle	35	*side*=0.8 *side-up*=0.06
painting, pillow	1709	*side-up*=0.6 *above*=0.3 *side*=0.1
bus, path	31	*side-up*=0.6 *above*=0.16 *on*=0.1
curtain, window	8077	*side*=0.6 *on*=0.14

the presence of "oven" (i.e., indoor scenes are not always representing kitchens). Nevertheless, even if the value of $CF(oven, wall)$ is small, it is still positive, which indicates that the presence of "wall" does not discourage the one of "oven". The right part of Table 4.2 reports instead some examples of negative Certainty Factor. For example, the presence of "sky" in an image strongly discourages the presence of "microwave", which is an indoor object. This pattern also occurs for the other pairs in the right part of the table, which contain objects belonging to different environments.

Some examples of relative area histograms are provided in Table 4.3. The first column of the table reports the class pairs (i.e., subject and reference), while the second one shows the support of the associated histogram. The likelihood values of the histograms are finally shown in the third column. Specifically, each histogram describes the discrete probability distribution of the relative object sizes, including the following three properties: *bigger*, *same*, *smaller*. From the examples, we can learn that 95% of the "runways" are bigger than "vans" and 94% of the times a "painting" is smaller than a "pool table". Even if, for perspective reasons, the apparent size of the objects may be distorted, we can see from the examples that

the proposed methodology is typically able to identify interesting relationships that describe real-world objects.

We conclude the analysis of the knowledge base by providing some examples of relative position histograms in Table 4.4. We reduce cluttering by showing only the most relevant likelihood values of each histogram. The first line of Table 4.4 states for example that "runways" are typically below "sky" (87%), and the second line shows how "balls" can be typically found inside "pool tables".

4.5.3 Anomaly detection

This subsection analyzes the results obtained with the anomaly detection techniques described in Section 4.4. In these experiments, the knowledge base extracted from the training set of the MIT Scene Parsing Benchmark is exploited to detect anomalies in the test set. Each test image is first segmented with the PSPNet model, then processed by SAD.

As described in Section 4.3, our technique is based on two parameters for the selection of the histograms in the knowledge base. We perform the following experiments by defining two parameter configurations:

(i) *Configuration a*, with $minsup = 10$, $thr_h = 0.98$ for co-occurrence and $thr_h = 0.99$ for position and size.

(ii) *Configuration b*, with $minsup = 10$, $thr_h = 0.98$ for co-occurrence and $thr_h = 0.97$ for position and size.

The first configuration requires, for the position/size categories, highly-confident histograms to detect anomalies, while the latter relaxes this constraint to exploit a higher number of histograms, even if they are less reliable. Variations of these thresholds in a small range (e.g., [0.95, 0.99]) do not significantly affect the results in the anomaly detection phase. Other sensitivity experiments on thr_h are provided later in the text.

Table 4.5 presents the number of detected anomalies for the two configurations defined so far. The total number of anomalies seems to be very high (i.e., 4581 for Config. a, 6731 for Config. b), compared to the number of test images (i.e., 2000). Actually, since each image presents many class pairs, the presence of a single misclassified object may yield many anomalies in the same picture. To verify this

Table 4.5: Number of detected anomalies in 2000 test images.

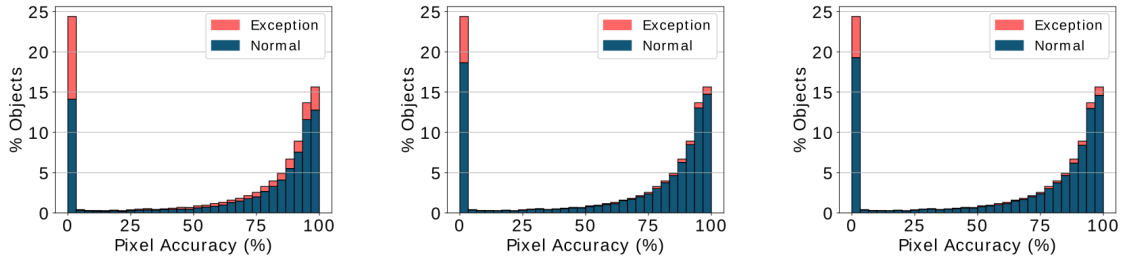
Config.	Total	Total (%)	Position	Area	Width	Height	Co-occurrence
Config. a	4581	6.4	1242	35	14	20	3270
Config. b	6731	9.3	3347	69	19	26	3270

concept and have a better estimate, the second column of Table 4.5 presents the percentage of object pairs (with respect to the total in the dataset) that are involved in anomalies. These numbers show that only 6.4 and 9.3 percent of the total object pairs in the test images are affected by anomalies.

The other columns of Table 4.5 provide the number of anomalies separately for each relationship category. It is worth noting that *co-occurrence* determines most of the anomalies for the two configurations. Also the *position* category generates more anomalies than the others: 1242 for Configuration a and 3347 for Configuration b. The highest number of *position* anomalies for Configuration b is consistent with the results we previously showed in Figure 4.4. Indeed, with higher thr_h values, the number of position histograms overcomes the amount of co-occurrence ones, increasing the probability of having anomalies of type *position*.

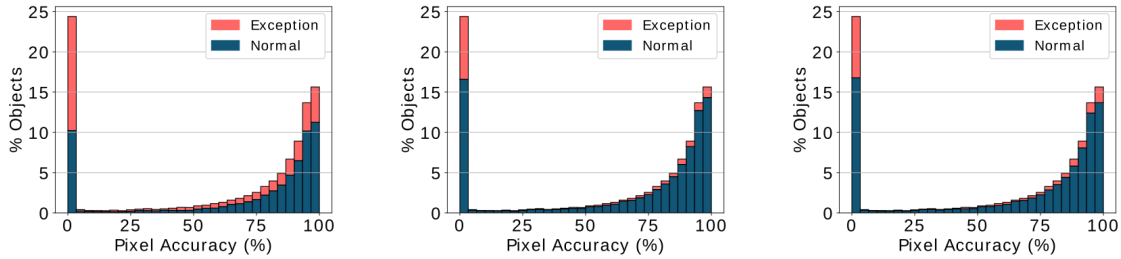
We can now analyze the quality of the detected anomalies by identifying misclassified objects with the three methods presented in Section 4.4. The analysis can be performed by inspecting the distribution of the discovered anomalies with respect to a quality metric typically used for evaluating semantic segmentation [29], called pixel accuracy. This metric is defined as the percentage of correct pixels predicted by the neural network model. It can be computed either on the overall pixels of the images to be recognized, or separately for each object. In the latter case, an object with pixel accuracy equal to zero is totally misclassified by the segmentation model. Conversely, when accuracy is one, the object has been correctly recognized. Finally, when the pixel accuracy takes intermediate values in the range [0, 1], it is likely that some parts of the object are recognized, but some others are confused with the background or the neighbor objects.

We expect that the detected anomalies will more likely involve objects that are partially or totally misclassified, while avoiding as much as possible those with high accuracy. Figure 4.5 and 4.6 show the results for Configuration a and b, respectively.



(a) Anomaly only method. (b) Delta method. (c) WTA method.

Figure 4.5: Anomaly detection results, Configuration a.



(a) Anomaly only method. (b) Delta method. (c) WTA method.

Figure 4.6: Anomaly detection results, Configuration b.

Each figure provides the results obtained with the corresponding configuration, separately for the Anomaly only, the Delta, and the WTA methods. The height of the histogram bins represents the number of objects in the test images whose pixel accuracy takes the values specified by the labels on the horizontal axis. Since the semantic segmentation model used in all the experiments is always PSPNet, the histogram heights are the same for all the charts in Figure 4.5 and 4.6. Finally, the number of objects is expressed in percentage with respect to the total predicted by the neural network in the test set. Consider that the segmentation model typically predicts more objects that the correct ones in the ground truth. Many of the additional objects labeled by PSPNet are just noisy patches of pixels, hence they will have pixel accuracy close to zero.

From the histograms, it can be immediately noticed how the semantic segmentation model tends to (i) predict object classes totally wrong, or (ii) almost-totally correct. The intermediate cases (e.g., accuracy values in range [0.1, 0.75]) present fewer objects. Despite the high number of objects classified with a very low accuracy (almost the 25% with accuracy < 0.1), the total pixel accuracy obtained

Table 4.6: Precision and recall for the *exception* and *normal* classes.

	Config.	Method	Precision (Ex)	Recall (Ex)	Precision (Norm)	Recall (Norm)	Macro avg. precision	Macro avg recall
1	a	Anomaly only	0.55	**0.39**	**0.71**	0.82	0.63	**0.61**
2	a	Delta	**0.66**	0.21	0.68	**0.94**	**0.67**	0.58
3	a	WTA	0.62	0.19	0.68	0.93	0.65	0.56
4	b	Anomaly only	0.50	**0.56**	**0.74**	0.70	0.62	**0.63**
5	b	Delta	**0.63**	0.29	0.70	**0.91**	**0.67**	0.60
6	b	WTA	0.59	0.29	0.69	0.89	0.64	0.59

by PSPNet on the test images is 0.8, which can be considered a good value. The reason is due to the fact that many of the almost-zero accuracy objects are just very small noisy patches of pixels, which weakly contribute to lower the final accuracy.

Consider now the two parts inside the bins, colored differently. The upper one, highlighted in red, represents the number of objects that are classified as exception by the specified anomaly detection method. The lower part of the histograms, depicted in blue, represents the objects that are classified as normal. On average, Configuration b (Figure 4.6) shows a higher number of exception objects than the one in Configuration a. This is due to the relaxed constraint on thr_h for position and size histograms in Configuration b. Indeed, more histograms in the knowledge base imply a higher number of detected anomalies. The Anomaly-only method is capable of assigning the exception class to many of the objects presenting a very low accuracy. Unfortunately, it also gives the exception class to very accurately classified objects. The Delta and the WTA methods predict instead a lower number of exception items, reducing the recall of exceptions among objects with low pixel accuracy. Conversely, they improve the prediction quality on high-accuracy objects, which are almost totally not involved by anomalies.

To better inspect the results from a quantitative point of view, we can assess the quality of the anomaly detection methods in the following way. We consider an object labeled with the *exception* label as a true positive if its pixel accuracy is lower than 75% (i.e., it is not totally recognized by the neural network). Conversely, the objects with higher pixel accuracy labeled as *exception* are considered false positives. With these definitions, it is possible to evaluate SAD by computing standard classification measures, such as precision and recall. Table 4.6 shows these results for the two experimental configurations and the three different anomaly detection methods.

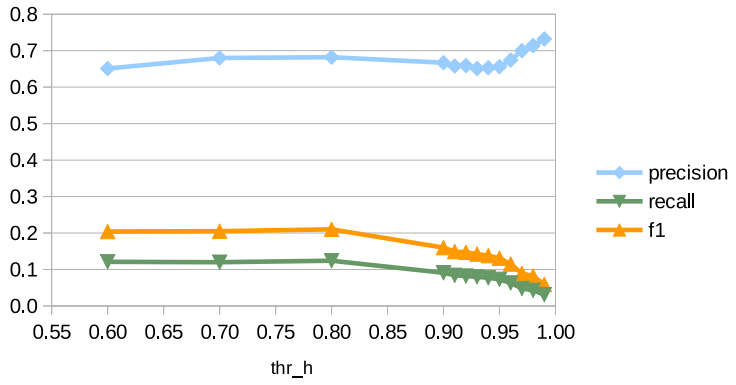


Figure 4.7: Delta method results, using only *co-occurrence* histograms

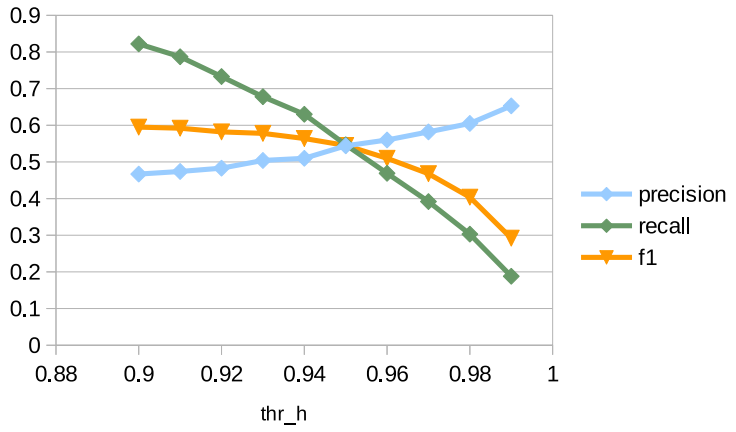


Figure 4.8: Delta method results, using only *position* and *size* histograms

As expected, for the exception class, the Anomaly only method presents the highest recall and the lowest precision, since it considers both the objects involved by an anomaly as if they have been misclassified by the neural network. The Delta method shows instead the best precision for the exception class and the best recall for normal data. These results highlight how this method is able to recognize exception objects in a precise way, whilst not affecting normal data. Similar considerations can be done for the WTA method, which presents on average slightly lower results with respect to the Delta method. From this analysis, we can see how the Delta method is the best trade-off between precision and recall on the exception class. Unfortunately, the WTA method did not produce improvements despite the attempt of reducing false positives with a more complex algorithm.

Given the promising results obtained with the Delta method, we further analyze its sensitivity with respect to the thr_h threshold. Figure 4.7 and 4.8 depict precision, recall, and f1 when varying this threshold. To analyze separately the influence of different categories, in Figure 4.7 only co-occurrence histograms are exploited for the prediction, while in Figure 4.8 only position and size histograms are considered. In Figure 4.7, precision increases significantly with $thr_h > 0.95$, but recall becomes lower. The highest f1 score is established at $thr_h = 0.8$, with precision 0.68. In Figure 4.8, precision is lower than 0.6 until $thr_h = 0.97$. At $thr_h = 0.99$ recall decreases to 0.2, but precision becomes comparable to the one obtained with co-occurrence histograms at $thr_h = 0.8$.

4.5.4 Lessons learned

We have seen how the SAD methodology can automatically derive a knowledge base describing, in an interpretable way, a set of training images. The results on anomaly detection show that SAD can correctly detect many low accuracy objects. In particular, the Delta method yields the most promising results, highlighting exception objects with a good trade-off between precision and recall. Unfortunately, the choice made while designing the WTA method did not provide any other improvements, reason why this methodology is not presented in our paper [88]. Finally, the sensitivity analysis of the Delta approach demonstrates that the threshold values to be set for our algorithm present smooth trends, which entails the possibility of choosing a good value without an excessive fine-tuning.

Chapter 5

SImS: Semantic Image collection Summarization with frequent subgraph mining

We have shown in Chapter 4 how contextual information and object relationships can be used for a better understanding of the image content and the inspection of possible errors in semantic segmentation. This chapter presents a second case study that leverages on object relationships and data mining techniques for summarizing image collections.

A first application of image collection summarization could be the generation of highlights and previews of personal albums (e.g., Google Photos) [58, 103]. Additionally, services such as Pinterest and Flickr could exploit the generated summaries to identify and suggest thematic collections based on user interests. Video-sharing platforms (e.g., Youtube) may also derive semantic summaries from video frames to automatically generate tags and categories [46]. Finally, another important application of image summarization can be found in the deep learning field. Specifically, image understanding tasks require huge amounts of labeled data for the training process. To improve this learning phase, data samples should be characterized by a high coverage of all the visual scenes a model has to recognize and a high diversity to avoid overfitting issues [18, 126]. Image collection summaries could be useful to assess these characteristics while designing training datasets.

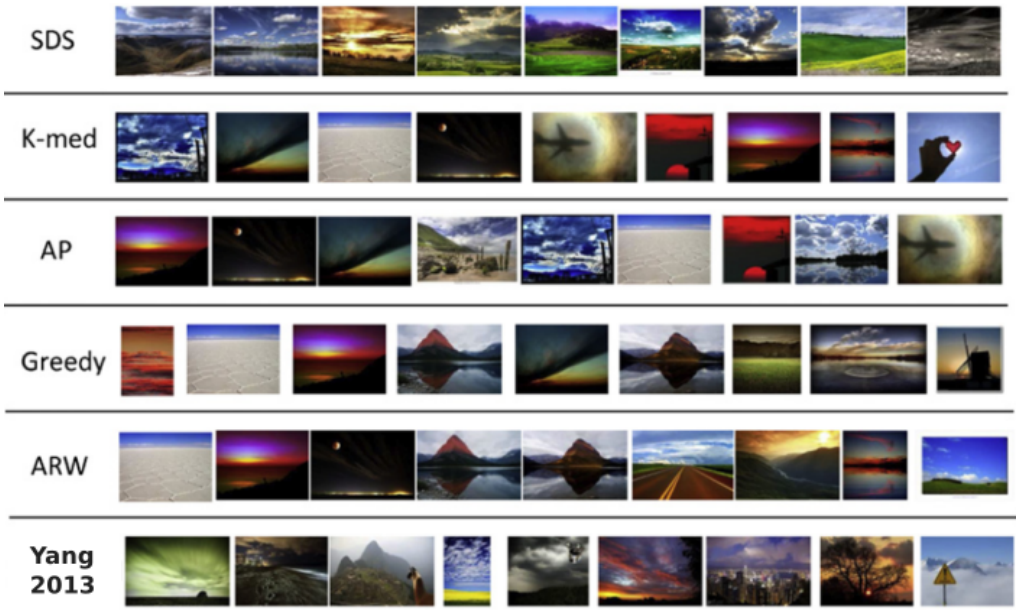


Figure 5.1: Comparison between traditional summaries [119].

Previous methods for image collection summarization typically rely on low-level visual features or on simple textual tags associated to each picture (e.g., Flickr image tags). Commonly, the output generated by these techniques is a subset of the input images or a set of image patches extracted from the original dataset. There are two main issues that characterize these summarization approaches. The former is that the usage of visual features and image tags does not allow capturing the semantic information of the pictures. Indeed, object classes and their relationships should be considered for a more effective analysis. The second issue is related to the output format of the generated summary. In fact, using a subset of the input collection does not allow to easily evaluate the representativity of the result. For example, the output could contain hidden patterns that are not immediately visible to the final user, reducing interpretability.

Figure 5.1 depicts the output of 6 traditional summarization techniques, each showing exactly 9 summary pictures. By inspecting these images, a user may easily infer that the input collection contains natural landscapes. However, there are no explicit cues about which semantic content is really important to summarize the collection. Some questions that a user could be interested in are: “Does sky appear in all images?”, “Is the airplane in the k-Medoids result important, or just the sky around it?”.

SImS, acronym for *Semantic Image Summarization*, is our technique to extract semantic summaries from large image collections. Differently from previous methods, it is designed to analyze the semantic content of the input dataset, in terms of object classes and relationships. The resulting summary consists of a set of abstract patterns that recurrently occur in the collection, paired with example images. Specifically, these output patterns are encoded in the form of scene graphs (see Section 2.3), which describe frequent object configurations discovered in the images under analysis. This characteristic makes our methodology more interpretable and semantics-aware. Additionally, frequent scene graphs generated by SImS can represent a source of information to answer queries such as "Find the Pinterest boards that show a person on a bike" or, more complex, "two people on the same bike". Other applications could involve the output patterns to create complex textual descriptions considering object relationships, such as "In the collection you can typically find scenes with a castle surrounded by vegetation". The novelty of SImS consists in applying frequent subgraph mining algorithms to analyze the scene graphs associated to the input images. In this work, we also extend the semantic information of standard scene graphs with fine-grained relationship types describing the relative position between objects. Specifically, SImS automatically derives them by applying the position classifier defined in Section 3.3 to images labeled with panoptic segmentation methods [59, 117].

Our technique considers two types of patterns: (i) *frequent pairwise object relationships*, and (ii) *frequent scene graphs*. Similarly to the SAD approach (see Chapter 4), frequent pairwise object relationships describe the distribution of the object configurations with a set of histograms. This first type of patterns is stored by SImS in the *Pairwise Relationship Summary* (PRS), which has a similar structure to the knowledge base defined for the SAD method. In this work, we only focus on relative position relationships, which is the most relevant type for describing the image content, as we learned from the anomaly detection experiments in Section 4.5. Figure 5.2 depicts some examples of PRS histograms. Specifically, it shows that the analyzed collection presents images where the pairs "mountain-sky" and "car-sky" frequently occur. Moreover, the two histograms report that the object "mountain" is typically found *hanging* (i.e., below with contact) from "sky", while "car" is typically located *below* "sky".

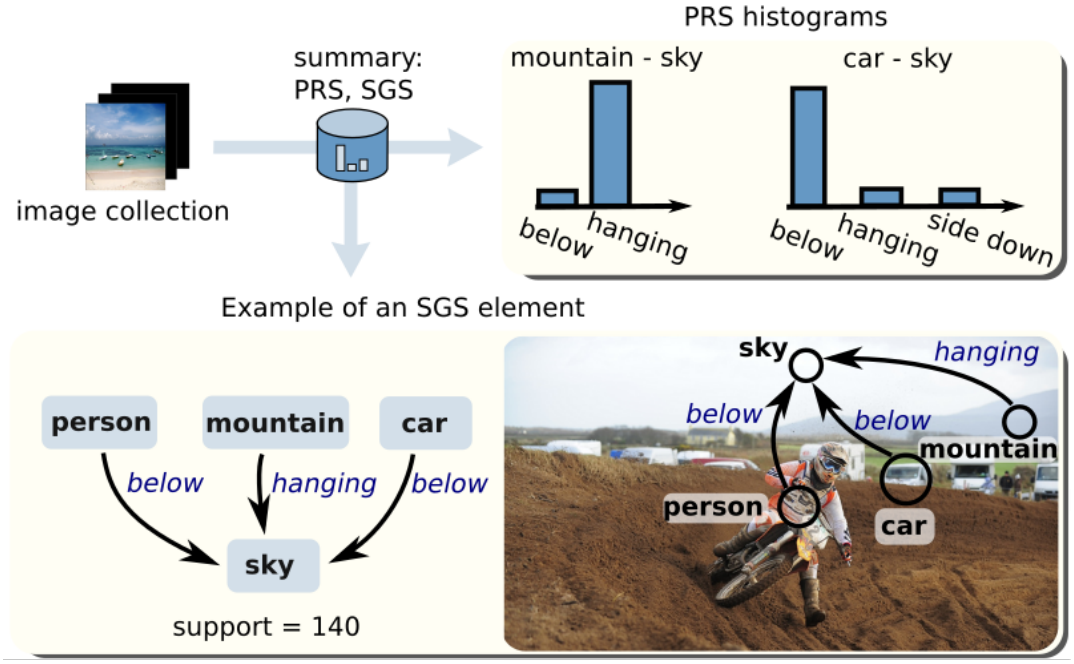


Figure 5.2: Example of summary patterns extracted by SImS from the COCO dataset.

The second type of patterns, called *frequent scene graphs*, represents the core result of SImS and describes common multi-object configurations detected in the input dataset. These patterns extend the concept of frequent pairwise object relationships defined in SAD. We show in the experiments how these scene graphs represent meaningful summaries of the input collection. Additionally, similarly to SAD histograms, they can provide actionable commonsense knowledge for image understanding tasks [2, 97]. The bottom-left part of Figure 5.2 presents an example of SGS graph that involves 4 different objects at the same time. Specifically, it is representative of 140 pictures in the collection (support value), where "mountain" object is *hanging* from "sky", while "person" and "car" are *below* "sky". In the bottom right of Figure 5.2, we finally provide one of the 140 pictures that include this example graph. Unfortunately, the extraction of such semantic patterns (i.e., the SGS) by means of frequent subgraph mining techniques has a high computational complexity and tends to generate redundancies in the output summary. Both issues can be solved by applying an effective preprocessing technique we designed in our work.

We recap the main contributions of SImS in the following.

(i) We define two types of summarization patterns: Frequent pairwise object relationships (PRS), and frequent scene graphs (SGS). Differently from previous work, our summaries are semantics-aware and more interpretable.

(ii) The relative position between objects is described with a novel set of fine-grained labels, defined in Section 3.3.

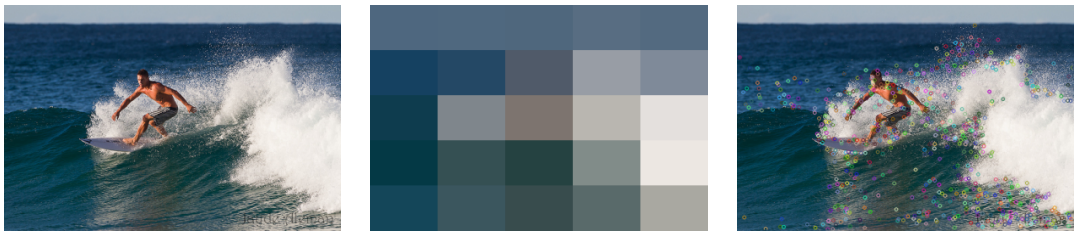
(iii) The scene graphs analyzed by SImS are automatically generated from panoptic segmentation exploiting our relative position classifier.

(iv) We designed a scene graph preprocessing step to remove redundant information and reduce the running time of the frequent subgraph mining process.

We organize the next sections as follows. In Section 5.1 we review previous works on image summarization, highlighting the contribution of the proposed methodology. Next, in Section 5.2, we present the main techniques for frequent subgraph mining, which is a core building block for extracting the SGS in SImS. In Section 5.3 we highlight the main issues with applying frequent subgraph mining on scene graphs without an appropriate preprocessing. The overview of the SImS process is provided in Section 5.4, while the details of its building blocks are described from Section 5.5 to 5.9. The summary evaluation metrics are shown in Section 5.10 and the experiments are finally described in Section 5.11.

5.1 Related works on image summarization

As previously introduced, image collection summarization aims at extracting important patterns to briefly describe huge sets of pictures. A very common way to approach this task consists in selecting a subset of the pictures that are considered to be more important to summarize the collection [19, 105]. These approaches typically describe images with visual features such as color histograms or Scale Invariant Feature Transform (SIFT). Then, feature vectors are used to inspect recurring patterns and extract the most significant images. The generated summary aims at a good *coverage* (i.e., all the patterns and scene types in the collection should be represented) and a high *diversity* (i.e., the images presented in the result should not be redundant in terms of visual aspect and semantic content).



(a) Input image (b) RAC (c) SIFT

Figure 5.3: Extraction of visual features to characterize images.

5.1.1 Extraction of image features

The first way of categorizing previous works in image summarization involves the type of features used to characterize the images. An effective feature extraction should be able to inspect different image aspects such as colors, textures and shapes. These techniques have been widely studied in the earliest years of computer vision, before the development of convolutional neural networks designed to automatically learn the feature extraction process. Instead of exploiting hand-designed features, some summarization techniques rely on more complex ones directly extracted from pretrained convolutional neural networks. In the following, we describe some examples of visual features used in previous works, with the aim of better understanding the semantic gap with respect to the information exploited by SImS.

Color histograms are one of the most well known feature extraction techniques used in literature. Each histogram is a fixed-size vector that describes the image by means of the color distribution of the pixels. This technique is suitable to generate a global feature vector for the image, but unfortunately one single color histograms does not provide spatial information. For this reason, a widely used approach consists in dividing the image in coarse regions or super-pixels, then computing a color histogram for each of them. A variant of this approach is Regional Average Colors (RAC), where the average color is picked instead of the complete color histogram (Figure 5.3b).

To better inspect the presence of shapes inside the image, corner and edge detection algorithms have been designed. Scale Invariant Feature Transform (SIFT) is a technique for detecting keypoints of different types to describe the image (Figure 5.3c). Keypoints are characterized by their position in the image and a feature vector (e.g., 64/128 features) describing the shape that has been recognized, such

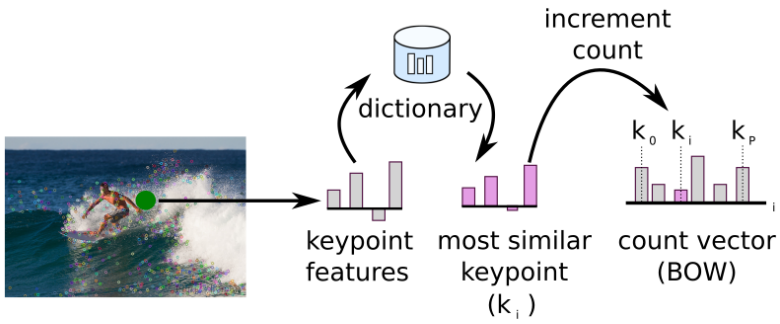


Figure 5.4: Image representation with Bag Of Words (BOW).

as specific types of edges or corners.

Since the number of detected keypoints in a picture is not predefined, the generation of a fixed-size feature vector for each image cannot be immediately performed. The BOW representation (i.e., Bag Of Words) is typically used to solve this problem. Figure 5.4 depicts the main steps to derive BOW vectors from images. Given an input image, the algorithm matches the feature vector of each detected keypoint with the nearest neighbor found in a dictionary of P prototypes. Every image is then described with a count vector (BOW) that specifies the number of matches for each of the P prototypes. The dictionary with prototypes is typically learned by running k-Medoids on a big set of keypoints extracted from the input image collection.

The features described so far can visually characterize a picture. However, visually similar images may have different semantic meanings, while similar images from a semantic point of view can present different visual aspects. Figure 5.5a depicts an example of two visually similar images that present different semantic concepts. Indeed, the picture on the left represents a cottage house, while the one on the right shows a dog house. The usage of contextual information (e.g., the presence of the dog next to the house, with the same size) could help in distinguishing between the two images. Instead, in Figure 5.5b two bedrooms are shown. The visual appearance of the two scenes is quite different, but there are some objects in common, such as bed and pillows, that help in understanding the scene type. As we will describe, SImS takes into consideration also the image content in terms of objects and their positions, which will help in the generation of semantics-aware and interpretable summaries.

(a) Visually similar.

(b) Semantically similar.

Figure 5.5: The semantic gap with visual features.

5.1.2 Image summarization methodologies

Different image collection summarization techniques have been proposed. In general, they share the same definition of summary, which is a subset of the initial collection and can be formalized as follows.

Given a dataset of images $D_{\mathcal{I}}$, the summary is defined as the subset

$$S_{\mathcal{I}} \subseteq D_{\mathcal{I}}, |S_{\mathcal{I}}| <= N$$

where N is the upperbound that specifies the maximum summary size. In the following, we review the main types of image summarization methods that share this formal definition, by providing some example papers that are well known in the community. We start by analyzing methods that only rely on visual features, describing first unsupervised techniques based on clustering, then optimization-based techniques. Finally, we analyze previous works that also introduced semantic analyses on the image content, used together with visual features.

Clustering techniques can be considered as one of the most frequent approaches in literature. Deng et al. [31] use Self Organizing Maps (SOM) [61] to recognize

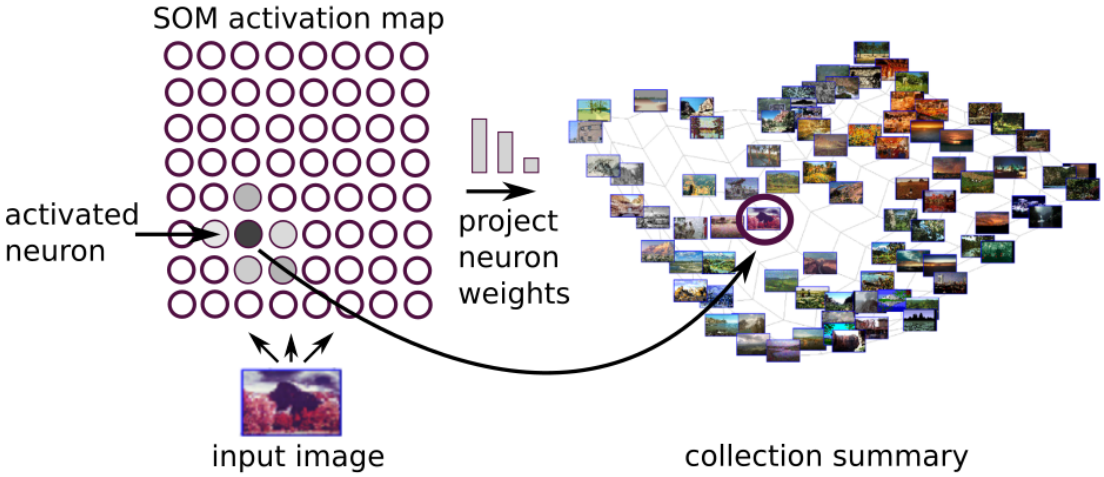


Figure 5.6: Image summarization with Self Organizing Maps [31].

similar images based on visual features. In this approach they use Regional Average Colors (RAC) to describe each image. Specifically, the picture frame is divided into 25 squared non-overlapped blocks and the average color is picked from each of them (Figure 5.3b). They also use a set of Gabor filters [81] to characterize textures in each region. The resulting feature vectors, extracted from all the images in $D_{\mathcal{I}}$, are provided to an 8×8 SOM that is responsible for learning a weight configuration describing the input collection. Finally, the images that generate the highest neuron activations are selected to take part of the summary $S_{\mathcal{I}}$ and displayed to a 2D plane. Specifically, as depicted in Figure 5.6, each summary image is associated with the neuron that shows the highest activation and positioned by projecting the neuron weight vector with Principal Component Analysis (PCA).

Hadi et al. exploit the k-Medoids clustering technique [49] to generate summaries. For the feature extraction they rely on SIFT keypoints, building feature vectors with a Bag Of Words (BOW) representation. After extracting BOW vectors, k-Medoids is run on all the images and the medoids are picked to build the final summary. Other techniques [124] include some preprocessing to filter unimportant SIFT features (e.g., using RANSAC) and exploit a different clustering technique called Affinity Propagation clustering, which does not require presetting the number of representative images.

A different category of approaches define a metric to be optimized when choosing the subset of summary images. Specifically, the subset $S_{\mathcal{I}}$ is selected by optimizing

a function $Q(\bullet)$ that establishes the quality of the summary, under a maximum summary size constraint (N):

$$S_{\mathcal{I}}^{*} = \arg\max_{S_{\mathcal{I}}} Q(S_{\mathcal{I}}), \quad S_{\mathcal{I}} \subseteq D_{\mathcal{I}} \wedge |S_{\mathcal{I}}| <= N$$

An example is the work proposed in [115], where the objective function $Q(\bullet)$ is designed with a mixture of M submodular functions:

$$Q(S_{\mathcal{I}}) = \sum_{m=1}^{M} w_m f_m(S_{\mathcal{I}})$$

Each component in the mixture of functions measures a different quality aspect of the generated summary, such as coverage and diversity. All of the defined submodular functions analyze the distance between images, measured by comparing their feature vectors (e.g., color histograms, SIFT features, super-pixels). Specifically, the coverage of the subset $S_{\mathcal{I}}$, is higher when the summary images are similar to the majority of those in the initial image collection $D_{\mathcal{I}}$. An example (but many others are defined in [115]) of coverage function used in the mixture could be:

$$f_{cov1}(S_{\mathcal{I}}) = \sum_{i \in D_{\mathcal{I}}} \sum_{j \in S_{\mathcal{I}}} s_{i,j}$$

where $s_{i,j}$ is the similarity between two images i and j. Similarly, the diversity of the subset $S_{\mathcal{I}}$ can be measured by inspecting the distances among summary images:

$$f_{div1}(S_{\mathcal{I}}) = \sum_{i \in S_{\mathcal{I}}} \sum_{j \in S_{\mathcal{I}}, i<j} d_{i,j}$$

The final mixture $Q(\bullet)$ considers all the functions defined so far to obtain the best quality summary. Images are selected by optimizing this score with an accelerated greedy algorithm for submodular functions [83].

A second way of defining the summarization task as an optimization method consists in minimizing the reconstruction error of the whole collection starting from the summary. For example, Yang et al. [119] use BOW feature vectors (based on SIFT) to compute linear combinations between summary images in $S_{\mathcal{I}}$ and verify that they are able to reconstruct the feature vectors of the complete set $D_{\mathcal{I}}$. Specifically, the feature vector v_i of an image $i \in D_{\mathcal{I}}$ can be reconstructed with the

following linear combination:

$$v_i = \sum_{j \in S_{\mathcal{I}}} \alpha_{ij} v_j \quad \forall\ i \in D_{\mathcal{I}}$$

where v_j are the feature vectors of the summary images and α_{ij} is the coefficient associated to v_j for reconstructing v_i. The summarization approach aims at minimizing the reconstruction error in L2 norm, defined as:

$$err = \sum_{i \in D_{\mathcal{I}}} ||v_i - \sum_{j \in S_{\mathcal{I}}} \alpha_{ij} v_i||$$

The authors propose to solve the optimization problem with simulated annealing, which avoids local minima and efficiently searches the optimal solution.

As previously described, the pure usage of visual features may yield incorrect or incomplete summaries. We discovered some previous works that can be considered pioneers in introducing semantic features for image summarization. Fan et al. [35] exploit the semantic information contained in textual annotations associated with the images. Specifically, they organize the summarization process in two levels. First, they divide the set of input images in semantic groups based on image descriptions, then they summarize separately the different groups based on visual features. The first step is achieved by considering each image in $D_{\mathcal{I}}$ as a textual document and performing Latent Semantic Analysis (LSA) on its tokens. The result is the definition of a set of semantic topics (e.g., airplanes, pets) and the division of the images in fuzzy clusters based on the agreement with the extracted topics. Each of these semantic clusters is then analyzed and summarized from a visual point of view by means of color histograms and texture descriptors. This step avoids visual redundancies that may be present in groups of images with the same semantic meaning. Therefore, the last summarization step consists in dividing each semantic cluster into a second level of groups, by means of visual features. The images that are closest to the centroids of the second-level groups are considered as representative to summarize each first-level (i.e., semantic) cluster.

Differently to [35], where the semantic analysis is conducted separately from the visual one, another approach [20] proposes a multimodal analysis. Each image is represented with BOW feature vectors that involve both visual and semantic aspects. Hence, the dataset is represented with two matrices (X_v for visual features,

X_t for textual ones) where rows represent the images and columns provide the attribute values. The visual features are encoded with BOW vectors extracted from SIFT keypoints, while textual attributes are defined with count vectors (one attribute for each distinct token). The multimodal analysis consists in applying a latent topic analysis with Non-negative Matrix Factorization (NMF) to X_v and X_t. The result is a matrix that specifies the degree of agreement of each image to the discovered multimodal topics. Finaly, for each of the discovered topics, the top-k most representative images (i.e., those with the highest weights) are selected and added to the summary $S_{\mathcal{I}}$.

The approach proposed by Samani et al. in [98] presents some new important characteristics with respect to the previously analyzed methods. Specifically, they apply an image classifier to the samples in the collection to be summarized. This operation allows obtaining a set of abstract concepts to describe the images without having access to textual tags or captions. However, since the available image classifiers are typically designed to recognize general object categories, these object classes may not be suitable to describe a domain-specific image dataset. For example the generic class "amphitheater" could be translated to "Colosseum", if the system knew that the image dataset is referring to Rome city. To this aim, the authors propose to map class labels with a domain ontology (such as DBpedia) that contains specific knowledge about the image collection to be analyzed. This operation allows a better semantic description by mapping each image with the associated domain concepts. The feature vectors with the mapped concepts are then used to compute a similarity graph between images. The final summary is computed by picking the most relevant images with the graph centrality metric.

Although these methods ([20, 35, 98]) try to involve semantic information in terms of image captions and tags, they do not consider further analyses of the object relationships and they all model the final summary with an image subset. In the next sections, we show how SImS is able to overcome these limitations by bringing the summarization task to a next level in terms of semantic understanding.

The summarization method proposed by Simon et al. in 2007 [103] differentiates from the techniques analyzed so far. The aim of the paper is to summarize large photo collections about world's important sites, such as cities and monuments. They define the concept of scenes and views. A scene S is a physical site in the world (e.g., the Colosseum or the Giza pyramid complex), while a view V

is an image that corresponds to a scene subset in terms of visible 3D keypoints. For example, in the case of a church scene, different views may depict various details such as the outdoor structure or internal elements like columns and frescoes. The proposed summarization approach considers a single scene S and processes a dataset of images ($D_{\mathcal{I},S}$) representing different views of S. The authors encode $D_{\mathcal{I},S}$ with a term-document matrix, where terms (columns) are the 3D keypoints of the scene and documents (rows) are the views described with a 1-hot vector. Specifically, these row vectors highlight which keypoints are visible in each view. The 3D keypoints used as terms are automatically learned by the system and they are computed exploiting SIFT descriptors. The summary images $S_{\mathcal{I},S} \subseteq D_{\mathcal{I},S}$ are selected with a greedy algorithm that minimizes a cost function evaluating diversity and coverage. The authors also define a second method to summarize image collections including multiple scenes by introducing a 2-level hierarchy. First, the different scenes are identified by matching SIFT descriptors between images, then for each scene the term-document matrix is build to define the final summaries. The idea of modeling *scenes* is partially exploited also in our technique, SImS. Indeed, a scene can be thought as a common multi-object pattern that frequently occurs in the collection. For example, a kitchen may be thought as a scene including objects such as “table”, “chair”, “oven”, “sink” configured in a specific way.

5.2 Related works on frequent subgraph mining

SImS performs the extraction of frequent patterns by means of frequent subgraph mining (FSM) techniques. This data mining task has a great variety of applications in other research fields such as biology, chemistry, and web data analysis. The algorithms for frequent subgraph mining can be categorized in different ways [48, 56, 90]. Considering the input type, these techniques can be designed to analyze a single big graph, or several smaller ones. The input graphs can be directed or undirected, dynamic or static. Also the output type can change with the algorithm. For example, exact FSM techniques return all the frequent subgraphs, while inexact methods just return a subset. Finally, FSM techniques can be categorized according to the algorithmic approach, such as Apriori- or pattern-growth-based.

In our work, we considered two state-of-the-art algorithms to mine frequent patterns from the labeled scene graphs. The first one is gSpan [118], an exact FSM

technique based on a Depth First Search approach. This algorithm requires a low amount of memory and it is suitable for very large graph collections. The second method is SUBDUE [26], an inexact algorithm based on graph compression. In our experiments, we will show summarization results exploiting both gSpan and SUBDUE, highlighting their pros and cons.

5.3 Frequent Scene Graph mining

The Scene Graph Summary (SGS) represents the core output of our semantic image summarization method. As anticipated in the previous sections, we extract frequent scene graphs by means of FSM techniques. Unfortunately, trying to directly use FSM algorithms on the scene graphs of a specific image collection has shown to suffer from the three following issues: (i) a very long *running time*, (ii) the presence of *high-entropy relationships* and (iii) the presence of *repeated items*.

5.3.1 Running time

With a preliminary analysis (see Section 5.11.2) we note that FSM algorithms, specifically SUBDUE and gSpan, take several minutes or even hours to produce the result when applied to big collections such as the Microsoft COCO dataset [60]. Moreover, gSpan has a variable running time that depends on its hyperparameter *minsup*. This value specifies the minimum percentage of transactions where a scene graph should occur to be included in the final result. Since gSpan is a DFS-based algorithm, the lower is *minsup* the higher is the running time to generate the summary. Conversely, with higher values of *minsup*, more patterns are excluded and the search space drastically reduces. Unfortunately, the excluded patterns, even if with low frequency, may sometimes represent interesting insights about the image collection. Less frequent, but representative, scenes are likely to be present in large image collections and for this reason a trade-off between running time and the number of discovered patterns should be found.

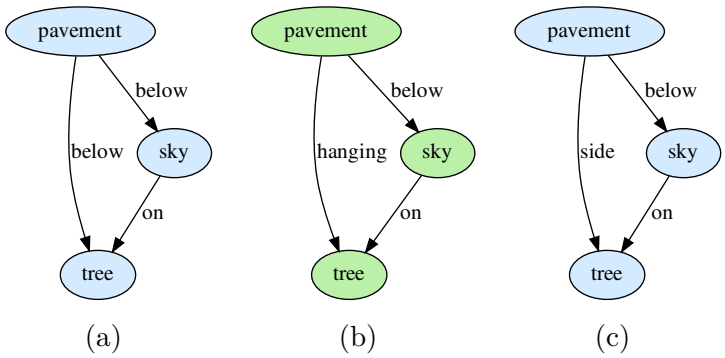


Figure 5.7: Example of frequent scene graphs presenting high-entropy relationships.

5.3.2 High-entropy relationships

The second drawback that emerges when applying FSM algorithms to scene graphs is the presence of redundant information in the output summary. As described in Section 5.1.2 summarization methods should generate non-redundant summaries, characterized by a high coverage with respect to the initial image collection. In complex image datasets, the large number of distinct object classes and the presence of many objects inside the same image brings to crowded scene graphs with irrelevant information. We call *high-entropy* relationships the edges that connect a pair of two specific classes if their relative position takes many different values in the input scene graphs (i.e., the objects with these two classes do not have a preferred relative position). These object pairs damage the final result, as they will contribute to obtain many similar frequent graphs, where the same object classes are connected by slightly different relationships.

For example, the object pair "pavement, tree" is characterized by a high-entropy relationship, as it appears in the COCO dataset with the "below", "hanging", or "side" relative positions. Figures 5.7a, 5.7b, and 5.7c show three examples of frequent scene graphs, generated by an FSM algorithm, where the object pair "pavement, tree" appears in the same context, but with a different position relationship. Since all the other nodes and edges in the two graphs are the same, the presence of both graphs in the SGS does not provide useful information as it reduces diversity.

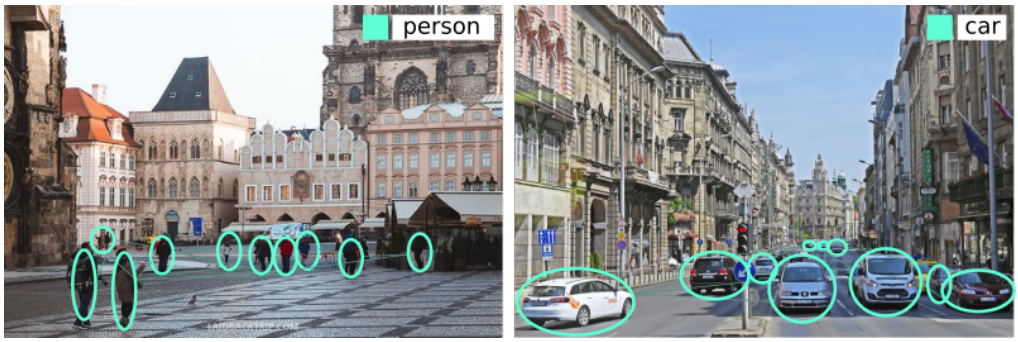
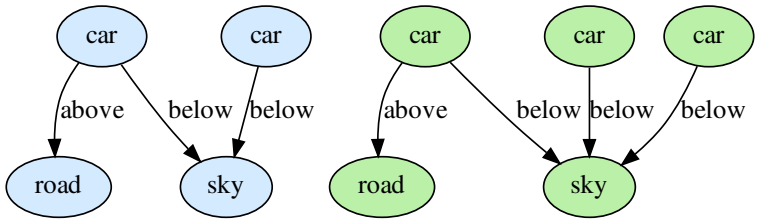


Figure 5.8: Example of crowded images with repeated items.



(a) (b)

Figure 5.9: Example of frequent scene graphs presenting repeated items.

5.3.3 Repeated items

Some object classes typically occur multiple times in the same image. For example, a picture representing a city square will contain many people, while a road scene will have many cars inside (Figure 5.8). The scene graphs associated with these images contain many nodes with the same class label and many low-informative relationships. After applying FSM on the scene graphs, the presence of repeated items contributes to obtain redundant frequent graphs with the same item included multiple times. Moreover, the summary could contain graphs with the same informative content, which only differ for the number of repetitions of a specific object class. Figures 5.9a and 5.9b show two of these frequent scene graphs, where the object "car" occurs two times in the first one and three times in the second one. Both graphs refer to a city scene, with "sky", "road" and "car" objects. From a semantic point of view the two graphs are equivalent and should not be present together in the same SGS to increase diversity.

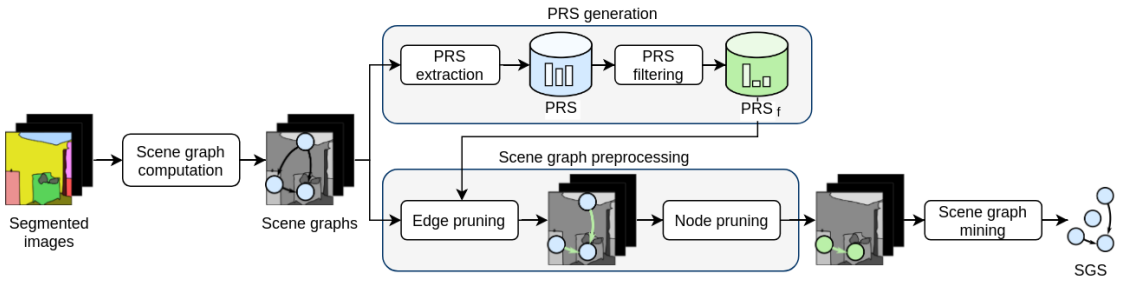


Figure 5.10: SImS architecture.

5.4 Semantic Image Summarization

SImS generates the output summaries, namely the PRS and the SGS, by applying frequent subgraph mining to the input scene graphs. We provide in Section 5.5 the formal definitions of scene graphs and of the two summary types. Afterwards, in Section 5.6-5.9 we describe the algorithmic component of each part of our algorithm. Figure 5.10 depicts the overview of SImS, whose building blocks are briefly outlined in the following.

Scene graph computation. This step is responsible for the transformation of a segmented image into a scene graph. Similarly to SAD, the process involves the relative position computation algorithm described in Section 3.3, which is applied to all the images in the collection being summarized.

Pairwise Relationship Summary generation. This building block inspects frequent pairwise object relationships in terms of relative position. The algorithm takes as input the collection of scene graphs generated at the previous stage, then collects the statistics about each object class pair in a similar way we defined for SAD (see Section 4.3). The summarized information, in the form of histograms, is stored in the PRS.

Scene graph preprocessing. As specified in Section 5.3, the scene graphs must be further preprocessed to reduce running time and avoid redundancies in the result. The preprocessing of scene graphs works by exploiting the summarized knowledge in the PRS.

Scene graph mining. During this step, SImS applies a frequent subgraph mining algorithm to the preprocessed scene graphs in order to generate the final Scene Graph Summary (SGS).

Label	Description
above	s is above r without contact
below	s is below r without contact
on	s is on top of r with contact
hanging	s is below r with contact
side	s and r are not vertical aligned
side-up	s and r are not vertical aligned, s is in a higher position
side-down	s and r are not vertical aligned, s is in a lower position
inside	s pixels are inside r shape
around	s pixels are around r shape

Table 5.1: Relative position labels for an edge connecting a subject-reference pair.

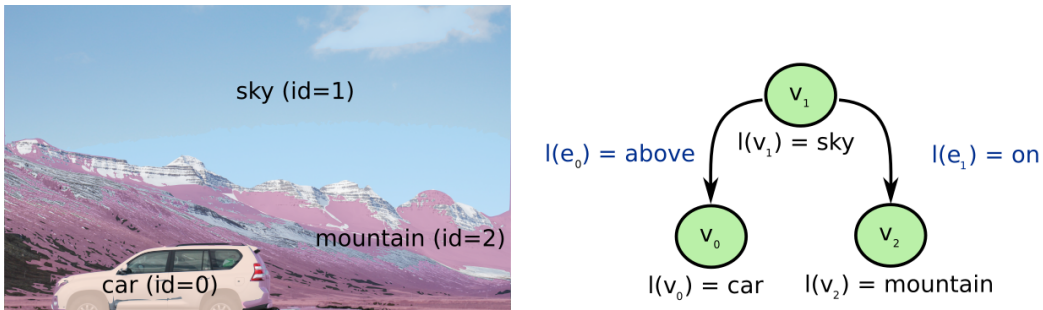
5.5 Summarization patterns

The PRS and the SGS summaries are designed to characterize frequent object relationships occurring in the input collection. The PRS describes pairwise relationships by means of discrete probability distributions, while the SGS exploits the *scene graph* data structure. Scene graphs are used not only in the SGS, but also in the first stage of the algorithm to represent the input images.

We define a scene graph with the directed graph:

$$G = (V, E, l_v, l_e)$$

where V is the set of vertices, E are the edges, $l_v(\bullet)$ is a function that assigns a class label to each vertex, and $l_e(\bullet)$ is a function assigning a relationship type to each edge. Every vertex $v \in V$ is associated to an object instance in the panoptic segmentation of a specific image, while $l_v(v)$ specifies its object class (e.g., "ship", "sky"). An edge connecting a pair of objects represents instead their position relationship. Specifically, two object instances $s \in V$ (subject), $r \in V$ (reference) are linked by a relationship if there exists an edge $e = (s, r)$, $e \in E$. The relationship type is specified by the edge label $l_e(e)$ and corresponds to a specific relative position among the 9 labels we defined in Section 3.3. We recall these relationships with a summary in Table 5.1. For a better understanding of this formal definition, we depicted in Figure 5.11 a visual example of scene graph associated to a segmented image with three object instances (i.e., "car", "sky", "mountain"). In this example,



(a) Input image, with panoptic segmentation.

(b) Associated scene graph.

Figure 5.11: Example of image representation by means of scene graphs.

the "sky" object is connected with "car" according to the relationship "above" and it is also connected with "mountain" following the relationship "on".

The main summary generated by SImS, called SGS, is defined as a set of frequent multi-object patterns represented with scene graphs. Specifically, it can be formalized with:

$$SGS = \{G_1, \ldots, G_i, \ldots, G_{|SGS|}\}$$

where each G_i is a frequent scene graph derived from the input collection by means of an FSM algorithm such as gSpan or SUBDUE, and $|SGS|$ is the number of summary graphs.

The Pairwise Relationship Summary is instead composed of a set of patterns inspecting common relative position relationships found among the object pairs in the collection under analysis. The probabilities of finding each specific relative position label for a give class pair is modeled in the form of histograms (i.e., discrete probability distributions).

Let $s_l \in L$ (subject label), $r_l \in L$ (reference label) be two object classes, where L is the set of object class labels available in the input collection. The discrete probability distribution that models the likelihoods of the position labels for two objects with classes s_l, r_l is defined with the following *histogram*:

$$h(s_l, r_l) = \{P(p_1|s_l, r_l), ..., P(p_i|s_l, r_l), ..., P(p_n|s_l, r_l)\}$$

where each value $P(p_i|s_l, r_l)$ reports the likelihood that a subject with class s_l and

a reference with class r_l are located with a relative position p_i.

To make an example, the histogram:

$$h(\textit{floor}, \textit{tv}) = \{P(\textit{side-down}|\textit{floor}, \textit{tv}) = 0.11, P(\textit{below}|\textit{floor}, \textit{tv}) = 0.82, ...\}$$

allows understanding that the input collection contains images where "floor" and "tv" co-occur and 82% of the times "floor" is below (without contact) the "tv". The lower likelihood values in the histogram are omitted for brevity.

The Pairwise Relationship Summary (PRS) is then defined as the set of histograms that are generated from the input collection by considering all the class pairs. More formally:

$$PRS = \{h(s_l, r_l) \mid (s_l, r_l) \in L \times L\}$$

where the cartesian product $L \times L$ represents all the possible class pairs.

5.6 Scene graph computation

This processing step consists of the transformation of an image labeled with panoptic segmentation $(\mathcal{L}, \mathcal{Z})$ into a scene graph. The object identifiers in the panoptic segmentation matrix $\mathcal{Z}$ (see Section 2.1.5) are exploited to form the different scene graph nodes $v \in V$. Afterwards, the matrix containing object class labels (i.e., $\mathcal{L}$) is used to define the function $l_v(\bullet)$ mapping each vertex id to the associated class. Finally, every pair of vertices is linked by the edges $e \in E$, each labeled with the function $l_e(\bullet)$ that exploits the output of our relative position classifier (see Section 3.3).

5.7 Pairwise Relationship Summary generation

After the generation of scene graphs, their information is used to build the PRS by considering common pairwise patterns. With a process similar to the one described for SAD, given each class pair s_l, r_l, we count the percentage of times that two objects with these classes appear at a specific relative position. The pseudocode of this process is described in Algorithm 4.

Algorithm 4 SImS: PRS extraction

Input: Scene graphs $\mathcal{G}_I$, $minsup_h$, $maxentr_h$
Output: filtered patterns PRS_f

1: $PRS = \{\}$
2: **for all** $(V_i, E_i, l_{v,i}, l_{e,i})$ in $\mathcal{G}_I$ **do**
3: **for all** $e = (s, r)$ **in** E_i **do**
4: $s_l = l_{v,i}(s), r_l = l_{v,i}(r)$
5: $p = l_{e,i}(e)$
6: $h(s_l, r_l) = getOrCreateHistogram(PRS, s_l, r_l)$
7: $increment(h(s_l, r_l), p)$
8: **end for**
9: **end for**
10:
11: **for all** $h(s_l, r_l)$ in PRS **do**
12: $sup(h(s_l, r_l)) = sum(h(s_l, r_l))$
13: $h(s_l, r_l) = h(s_l, r_l)/sup(h(s_l, r_l))$
14: $ent(h(s_l, r_l)) = entropy(h(s_l, r_l))$
15: **end for**
16:
17: $PRS_f = filter(PRS, minsup_h, maxentr_h)$
18: **return** PRS_f

The main loop (line 2) iterates on all the scene graphs of the input collection, while the inner loop (line 3) iterates on all the edges in the selected graph. From each edge, SImS extracts the information of the subject and reference class labels (line 4), then the relative position $l_{e,i}(e)$ (line 5). Afterwards, it retrieves (or creates if it does not exist) from the PRS the histogram associated to the class pair (line 6) and increments the count corresponding to position p (line 7). At the end of the main loop, it normalizes the histogram counts to obtain the discrete probability distributions describing the frequent pairwise patterns (line 11). The normalization is made by dividing the histogram values with the number of object pairs that updated the histogram counts (i.e., their *support*).

Finally, the PRS histograms are filtered to keep only the most relevant information and reduce the summary size. The filtering process is based on two measures that describe the quality of each histogram: *support* and *entropy*. The support of the class pair is computed on histogram counts before normalization (line 12)

and measures the reliability of the collected information. Indeed, the more training samples represent the histogram, the higher is the likelihood that the inferred pattern is less affected by noise.

Entropy is instead a quantity defined in information theory for discrete probability distributions [101]. It measures the average level of information present in the distribution. Let $h(s_l, r_l)$ be a histogram distribution in the PRS and $P(p_i|s_l, r_l)$ be the likelihood associated with position p_i. The information content carried by an event where two objects with classes s_l, r_l satisfy the position p_i is:

$$I(p_i|s_l, r_l) = -log(P(p_i|s_l, r_l))$$

Indeed, when $P(p_i|s_l, r_l)$ is low and the event is verified (i.e., s_l, r_l satisfy position p_i), the quantity of information is higher because the event is unexpected. Entropy averages the information content for all the possible events (i.e., positions p_i in our case) in the histogram distribution. It is defined as:

$$entropy(h(s_l, r_l)) = E[-log(P(p_i|s_l, r_l)] = -\sum_i P(p_i|s_l, r_l) log(P(p_i|s_l, r_l))$$

The product $-P(p_i|s_l, r_l)log(P(p_i|s_l, r_l)$ is lower when $P(p_i|s_l, r_l)$ is either 0 or 1. Instead, it is higher when $P(p_i|s_l, r_l)$ is close to 0.5. This means that unbalanced histograms that present few likelihoods with values close to 1 and many values close to 0 will have a lower entropy. Conversely, a histogram where all the positions p_i share almost the same likelihood, will present a higher entropy. In our analysis we are interested in keeping the histograms with a low entropy, which describe object pairs satisfying a small predefined set of relative positions.

According to this reasoning, SImS filters out the histograms that have a high support and a low entropy value (line 17), storing the result in PRS_f. The threshold values $minsup_h$ and $maxentr_h$ will be discussed in Section 5.11.1.

5.8 Scene graph preprocessing

In Section 5.3 we inspected the main drawbacks that occur by applying Frequent Subgraph Mining directly to the scene graphs. The scene graph preprocessing described in this section aims at reducing the FSM running time and avoiding the

Algorithm 5 SImS: scene graph preprocessing

Input: Scene graphs $\mathcal{G}_I$, PRS_f
Output: Preprocessed scene graphs

for all $(V_i, E_i, l_{v,i}, l_{e,i})$ in $\mathcal{G}_I$ **do**
  for all $e = (s, r)$ **in** E_i **do**
    if $h(l_{v,i}(s), l_{v,i}(r)) \notin PRS_f$ **then**
      $E_i = E_i \setminus e$
    end if
  end for

  $V = copy(V_i)$
  while $||V|| > 0$ **do**
    $v_i = pop(V)$
    for all $v_j \in V$ **do**
      if $equivalent(v_i, v_j)$ **then**
        $V = V \setminus v_j$
        $V_i = V_i \setminus v_j$
        $E_i = E_i \setminus \{\forall e \in E_i \mid r = v_j \vee s = v_j\}$
      end if
    end for
  end while
end for
return $\mathcal{G}_I$

presence of redundancies (i.e., repeated items, high-entropy relationships) in the final summary. To this aim, we designed the *edge pruning* and the *node pruning* steps that are applied directly after the generation of the scene graphs. The details of these two steps are provided in the next subsections, while Algorithm 5 shows the pseudocode for the whole preprocessing pipeline. The effectiveness of the proposed preprocessing, in terms of summary quality and reduction of running time, is accurately verified in Section 5.11.2.

5.8.1 Edge pruning

This part of the preprocessing has the goal of reducing the presence of high-entropy relationships. This is achieved by exploiting the information collected in the PRS_f summary. Specifically, high-entropy relationships are defined to be the edges adjacent to each pair of vertices s, r whose class labels (i.e., $l_v(s), l_v(r)$) are associated to a histogram $h(l_v(s), l_v)$ that is not relevant (i.e., due to high entropy or

low support). More formally, an edge $e = (s, r) \in E$ of a scene graph $G(V, E, l_v, l_e)$ is deemed to be a high-entropy relationship if:

$$h(l_v(s), l_v(r)) \notin PRS_f$$

Therefore, the input scene graphs are pruned by removing all the edges with high entropy (lines 2-4 of Algorithm 5). The lower number of edges reduces running time during FSM, whilst maintaining the important information that will contribute to the final SGS.

5.8.2 Node pruning

This second part of the preprocessing algorithm aims at detecting repeated items in the scene graphs. The process is performed by comparing each single node with the others in the same graph. Specifically, we deem two nodes to be equivalent if they share the same object class and they are connected with the same relationship types to other objects in the image. An example of equivalent nodes may be the presence of many cars that are all positioned "on road" and "below sky" in the same city scene. Including of all these nodes is not necessary for the summary, hence only one of them will be kept by the algorithm. In the following, we first show the definition of equivalent nodes, then detail the node pruning process by presenting its pseudocode.

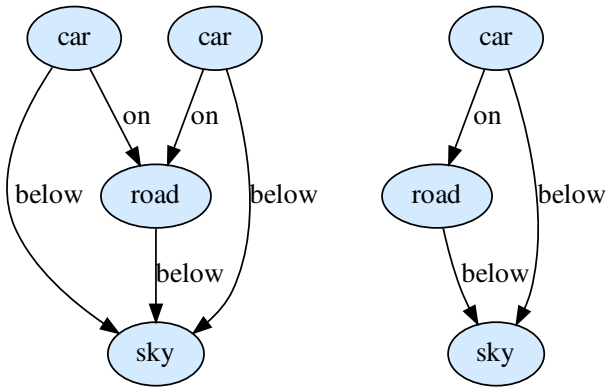
The definition of equivalent nodes requires two functions that describe the outbound and the inbound edges of each vertex v.

Definition 4 (outbound edge description)**.** *Let $v \in V$ be a vertex in a scene graph $G = (V, E, l_v, l_e)$. Outbound edges are described with the following set:*

$$out(v) = \{(l_e(e), l_v(v_k)) \mid e = (v, v_k) \in E\}$$

where e are the outbound edges of v with outbound nodes v_k.

By means of this function, the labels of the outbound edges and the outbound nodes are collected in the set of tuples $(l_e(e), l_v(v_k))$. A symmetric definition provides the description of inbound edges.



(a) Equivalent nodes. (b) Pruned graph.

Figure 5.12: Node pruning.

Definition 5 (inbound edge description)**.** *Let $v \in V$ be a vertex in a scene graph $G = (V, E, l_v, l_e)$. Inbound edges are described with the following set:*

$$in(v) = \{(l_e(e), l_v(v_k)) \mid e = (v_k, v) \in E\}$$

where e are the inbound edges of v with inbound nodes v_k.

The concept of node equivalence exploits the previous definitions. Specifically, two nodes are equivalent if they share the same class label and the same description of inbound and outbound nodes.

Definition 6 (node equivalence)**.** *Let $v_i, v_j \in V$ be two nodes. They are equivalent if:*

$$l_v(v_i) = l_v(v_j) \wedge in(v_i) = in(v_j) \wedge out(v_i) = out(v_j)$$

With this definition, we can detect and remove nodes that share the same semantic information (i.e., the same object class and the same labels for inbound and outbound edges). A visual example of equivalent nodes is provided in Figure 5.12a, where two objects with label "car" are positioned in the same way with respect to "road" and "sky". After node pruning, the result is shown in Figure 5.12b: Only one of the two "car" objects has been kept in the graph.

The process for removing equivalent nodes is detailed in Algorithm 5. In line 8 the procedure copies into V the set of vertices of the analyzed graph. The set V is exploited as auxiliary data structure to track all the processed vertices: When it becomes empty (line 9), the node pruning process is complete. Instead, the original set V_i will be updated at each iteration, by removing the unnecessary nodes according to the pruning algorithm. The procedure iterates by removing from V one element (i.e., v_i) at a time (lines 9, 10). The selected vertex v_i is compared to all the remaining vertices in V, to search for equivalent nodes that will be removed. Specifically, in lines 13-14 each equivalent node v_j is removed from both the copy V (to avoid being processed again in the next while-loop iterations) and V_i (the original scene graph). All the edges $e \in E_i$ that are adjacent to v_j are removed from the graph, as well (line 15).

Before describing the final step of SImS that involves scene graph mining, we highlight the reason why node pruning is applied after edge pruning and not viceversa. The node equivalence condition requires a complete matching of the edge labels. This condition is difficult to be verified when the scene graphs present many high-entropy relationships. For example, two nodes may share exactly the same edge labels with the exception of a single high-entropy relationship that assumes two different values. Node pruning benefits from the previous application of edge pruning, which removes high-entropy relationships and increases the probability of finding equivalent nodes.

5.9 Scene graph mining

The Scene Graph Summary is generated from the preprocessed scene graphs by means of a state-of-the-art Frequent Subgraph Mining algorithm. In our experimental setting we inspected the usage of both gSpan and SUBDUE, which are designed to work with labeled graphs. Specifically, in Section 5.11.2 we analyze the obtained results, by varying the FSM algorithm configuration. The frequent graphs extracted with FSM are stored in the SGS, which is the final summary provided by SImS. The SGS also contains the support value of each frequent graph, allowing a better understanding of its importance to represent the input images.

In order to obtain a more complete visual representation of the final summary, SImS associates each frequent graph with an example image. This operation is

performed by comparing each G_i in the SGS with the scene graphs associated to the images in the input collection. Specifically, SImS extracts from the input collection the smallest super-graph of each pattern G_i (i.e., it only includes the important information to represent the frequent pattern). The images associated to the selected super-graphs are collected for depicting the final visual summary.

5.10 Evaluation methodology

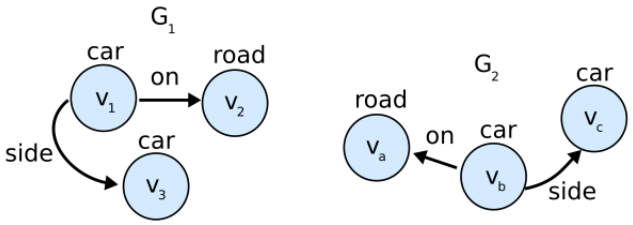
As anticipated in Section 5.1, image collection summarization methods must be able to provide summaries without redundancies and with a high representativity of the input collection. To this aim, in this work we analyze SImS result by means of two frequently adopted metrics: *coverage* [98, 106] and *diversity* [20, 115].

Since the dataset under analysis does not come with ground-truth summaries, the evaluation techniques such as ROUGE [75], V-ROUGE [115] and VERT [74] cannot be applied. Specifically, ROUGE is a set of recall-based metrics that analyze the overlap between the predicted summary and a reference summary (i.e., the ground truth). These metrics are designed to compute the percentage of n-grams or skip-grams matching with the reference. V-ROUGE implements the same principles of ROUGE in the computer vision field. In particular, skip-grams and n-grams are substituted by visual words that describe the image content (e.g., SIFT descriptors, color histograms or super-pixels). Finally, VERT is designed to evaluate video summaries in the form of ranked keyframes. This metric measures the precision of the position of each keyframe in the predicted ranking with respect to the reference ones.

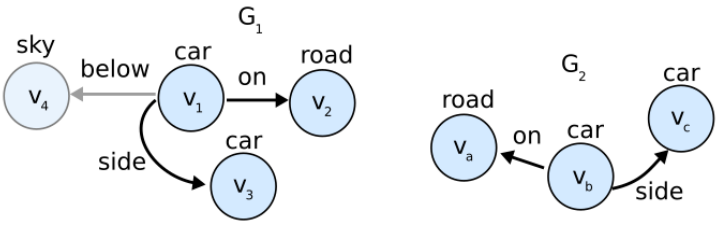
SImS output differs from standard summaries as it consists of a set of frequent scene graphs. For this reason, also the standard definitions of coverage and diversity cannot be directly used. Hence, we propose in the following a slight modification of these two metrics to fit our case study.

We define *coverage* as a value in range $[0, 1]$, which is higher when the final summary is sufficiently representative of the scene graphs in the input collection. Our coverage definition is based on the subgraph isomorphism property [27].

Graph isomorphism can be used to identify equivalent graphs that carry the same information. Let V_i and V_j be the set of vertices of two graphs G_i and G_j,



(a) Example of isomorphic graphs.



(b) Graph G_2 is subgraph isomorphic to G_1.

Figure 5.13: Subgraph isomorphism.

respectively. Graph isomorphism is a bijection $f : V_i \rightarrow V_j$ that maps the vertices of the two graphs and satisfies the following property. Let $v_x, v_y \in V_i$ be two vertices of G_i. The corresponding nodes of G_j $(f(v_x), f(v_y) \in V_j)$ must be adjacent (i.e., connected by an edge) if and only if v_x, v_y are adjacent in G_i.

In the case of labeled graphs the bijection f must also satisfy the property $l_v(v_x) = l_v(f(v_x)) \wedge l_v(v_y) = l_v(f(v_y))$ (i.e., the mapped nodes must have the same label). Furthermore, when considering node adjacency, the label of the edge e_i connecting v_x and v_y must be the same of the one connecting $f(v_x)$ with $f(v_y)$.

Figure 5.13a shows two isomorphic graphs (G_1, G_2), where the bijection function maps the nodes in the following way: $f(v_1) = v_b, f(v_2) = v_a, f(v_3) = v_c$. Note how the class labels of the mapped vertices are the same (i.e., both v_1 and v_b belong to class *car*) and the edge labels among adjacent nodes are identical (i.e., the edge that connects v_1 to v_2 and the one connecting v_b to v_a have both label *on*).

Subgraph isomorphism extends the previous definition to identify when a graph is a subset of another one. Specifically, subgraph isomorphism applies to two graphs G_i, G_j when at least a subgraph of G_i is isomorphic to G_j. Figure 5.13b depicts two graphs satisfying this property. Specifically, graph G_2 is isomorphic to the subgraph of G_1 including nodes v_1, v_2, v_3.

As previously mentioned, our metric for evaluating the summary coverage is

based on subgraph isomorphism. Specifically, a scene graph in the input collection is *represented* by the summary if at least one of its subgraphs is isomorphic to a graph in the SGS. Every image in the input collection should be represented to obtain a full coverage.

Definition 7. *(represented graph). Let G_i be an arbitrary scene graph in the input collection $\mathcal{G}_I$. G_i is represented in the SGS if the following function is true:*

$$represented(G_i, SGS) = \begin{cases} 1, & if\ \exists\ G_j \subseteq G_i \mid G_j \in SGS \\ 0, & otherwise \end{cases}$$

where the operator $\subseteq$ indicates subgraph isomorphism.

Definition 8. *(coverage). Let SGS be a summary and $\mathcal{G}_I$ the input scene graphs. We define SGS coverage with respect to $\mathcal{G}_I$ as:*

$$coverage(SGS, \mathcal{G}_I) = \sum_{G_i \in \mathcal{G}_I} represented(G_i, SGS)/|\mathcal{G}_I|$$

where $|\mathcal{G}_I|$ is the number of images in $\mathcal{G}_I$.

Coverage values range from $|SGS|/|\mathcal{G}_I|$ to 1. Higher values are given for better quality summaries.

To assess redundancy in the provided summary, *diversity* specifies the average dissimilarity between graphs in the SGS. A high diversity should be presented by a good quality summary. We define two different metrics: (i) *node diversity*, and (ii) *edge diversity*.

Remember that the output graphs in the SGS are characterized by the presence of low-entropy relationships thanks to the edge pruning step (see Section 5.8). For this reason, two graphs with the same nodes may be considered semantically equivalent. In the case of node diversity, graph dissimilarities can be reasonably measured by only analyzing node labels. Specifically, the dissimilarity between two graphs can be defined as the complement of Intersection over Union (IoU) between their node labels.

Definition 9. *(node dissimilarity). Let G_i, G_j be two graphs in the SGS. Their node dissimilarity is defined as:*

$$nd(G_i, G_j) = 1 - \frac{|l_v(G_i) \cap l_v(G_j)|}{|l_v(G_i) \cup l_v(G_j)|} \tag{5.1}$$

where $l_v(G_i), l_v(G_j)$ represent the node labels of G_i and G_j respectively.

Node diversity is finally computed as the average node dissimilarity of SGS graphs.

Definition 10. *(node diversity). Let SGS be a scene graph summary. Its node diversity is defined as:*

$$diversity_n(SGS) = \begin{cases} \sum_{G_i, G_j \in SGS, i<j} \frac{nd(G_i, G_j)}{|SGS| * (|SGS|-1)/2} & if |SGS| > 1 \\ 1 & otherwise \end{cases} \tag{5.2}$$

where G_i and G_j are summary graphs.

This metric ranges in [0, 1]. Higher values imply a better summary because the summary graphs contain non-redundant information.

We define a second metric, edge diversity, which takes into consideration edge labels when comparing graphs. This metric exploits a function that describes the list of edges in a scene graph with a set of tuples.

Definition 11. *(edge description). Let G_i be a scene graph in the SGS. Its edges are described with:*

$$edges(G_i) = \{(l_v(v_j), l_e(e), l_v(v_k)) \mid e = (v_j, v_k) \in E\} \tag{5.3}$$

where e is an edge of G_i connecting the nodes v_j, v_k, while $l_v(v_j)$ and $l_v(v_k)$ are the node labels, and $l_e(e)$ is the edge label.

Similarly to node dissimilarity, we define the edge dissimilarity and edge diversity by exploiting edge descriptions.

Definition 12. *(edge dissimilarity). Let G_i, G_j be two graphs in the SGS. Their edge dissimilarity ed is defined as:*

$$ed(G_i, G_j) = 1 - \frac{|edges(G_i) \cap edges(G_j)|}{|edges(G_i) \cup edges(G_j)|} \tag{5.4}$$

Definition 13. *(edge diversity). Let SGS be a scene graph summary. Its edge diversity is defined as:*

$$diversity_e(SGS) = \begin{cases} \sum_{G_i,G_j \in SGS, i<j} \frac{ed(G_i,G_j)}{|SGS|*(|SGS|-1)/2} & if |SGS| > 1 \\ 1 & otherwise \end{cases} \tag{5.5}$$

where G_i and G_j are summary graphs.

5.11 Experimental evaluation

The Microsoft COCO dataset [60] is suitable for our experimental evaluation, as its images are labeled with panoptic segmentation. COCO contains 118K pictures annotated with 53 stuff classes and 80 object labels. It also presents a great variety of contents and represented scenes, with a high number of labeled objects for each image. We used its ground-truth labels to analyze the effectiveness of our summarization method regardless of the panoptic segmentation model. Other datasets including panoptic annotations, such as Cityscapes [28] and ADE20K [125], are less suitable due to the lower variety of represented scenes (e.g., only road pictures or indoor ones), but could be analyzed with appropriate changes on the input data preprocessing pipeline. From a technical point of view, the only drawback of changing dataset and applying SImS is related to the design of the preprocessing steps for reading the panoptic annotations, which may be encoded with different formats.

We organize the next subsections as follows. First, we evaluate the results of the PRS extraction step on the COCO images. Afterwards, we will assess the effectiveness of the scene graph preprocessing steps (i.e., node and edge pruning). Next, we will move on with the analysis of different frequent subgraph mining algorithms and configurations, focusing on summary quality and running time. Finally, we will show a qualitative and quantitative comparison with previous summarization techniques.

All the experiments presented in this section are run with the following hardware configuration: Intel Xeon Gold 6140, CPU @ 2.30GHz, RAM 40 GB. Our summarization algorithm is fully implemented in Python 3, while the frequent subgraph mining process relies on available online C implementations [26, 118]. The

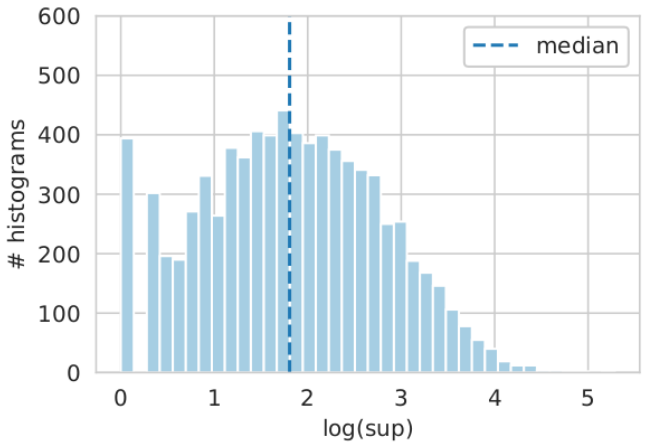


Figure 5.14: Support distribution of PRS histograms.

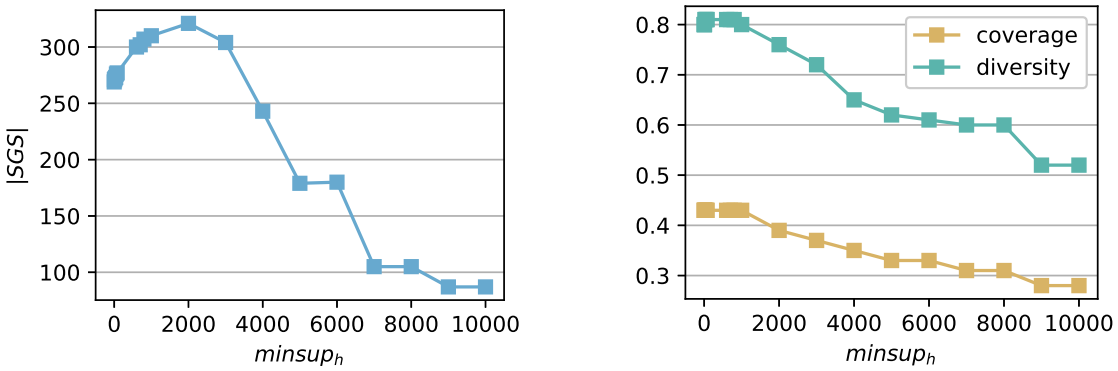
complete source code of SImS is provided in our GitHub repository:
https://github.com/AndreaPasini/SImS

5.11.1 Pairwise Relationship Summary generation

The Pairwise Relationship summary is built directly after creating the scene graphs on the whole COCO dataset. The extraction of the scene graphs, containing 5M relationships in total, takes 4 hours, while the PRS is generated from the scene graphs in about 20 seconds. The resulting PRS contains 7867 histograms, whose support distribution is depicted in Figure 5.14. Since the curve is positively skewed, to obtain a gaussian-shaped distribution and increase readability, we have shown the horizontal axis values in log-scale.

The high number of histograms in the PRS is reduced by means of the PRS filtering phase, described in Section 5.7. Specifically, the $minsup_h$ and the $maxentr_h$ thresholds are imposed to select the most relevant histograms. We enforce $minsup_h$ to improve the running time of the frequent subgraph mining process. Setting higher values allows reducing the outlier histograms in the PRS_f (i.e., those supported by few elements) and decreasing the FSM time. To identify feasible values for this threshold, we performed a sensitivity analysis by inspecting the range $[0, 10000]$ (which corresponds to $[0\%, 5\%]$ with a relative support). The sensitivity analysis is conducted on the full COCO dataset, running gSpan with $minsup = 0.01$ as graph mining method. We consider the effect of $minsup_h$ on both the SGS size and the

two summary evaluation metrics, namely coverage and diversity.



(a) Number of SGS graphs.

(b) SGS quality.

Figure 5.15: $minsup_h$ sensitivity.

Figure 5.15a depicts the number of summary histograms while varying $minsup_h$. With increasing values of the threshold, the summary size climbs from 269 to a maximum of 321 at $minsup_h = 2000$. The increase is due to the elimination of outlier relationships (i.e., supported by few items) that helps the node pruning step to obtain simpler graphs. Indeed, a better simplification of the input graphs implies a higher presence of frequent patterns. Instead, when the value of $minsup_h$ grows too much, the number of graphs decreases as the removal of nodes and edges from the input images starts causing the presence of empty graphs (which are not considered in the FSM process).

The sensitivity of the SGS quality with respect to changes of $minsup_h$ is shown in Figure 5.15b. The presence of a very stable plateau for both coverage and diversity can be immediately seen for $minsup_h$ in $[0, 1000]$. Specifically, coverage remains fixed to 0.43, while diversity fluctuates between 0.80 and 0.81. Note that if diversity and coverage are approximately constant in range $[0, 1000]$, the number of SGS graphs increases, instead. This entails that the SGS graphs that are included in the result while increasing $minsup_h$ are meaningful (i.e., they do not reduce diversity), but they are representative of the same images (i.e., coverage does not increase). Finally, once we increase additionally $minsup_h$ after the plateau, both coverage and diversity start to slightly decrease. For this reason, choosing values in $[0, 1000]$ is a good choice from the summary quality point of view, remembering that the higher is $minsup_h$ and the lower is the running time. In all the following

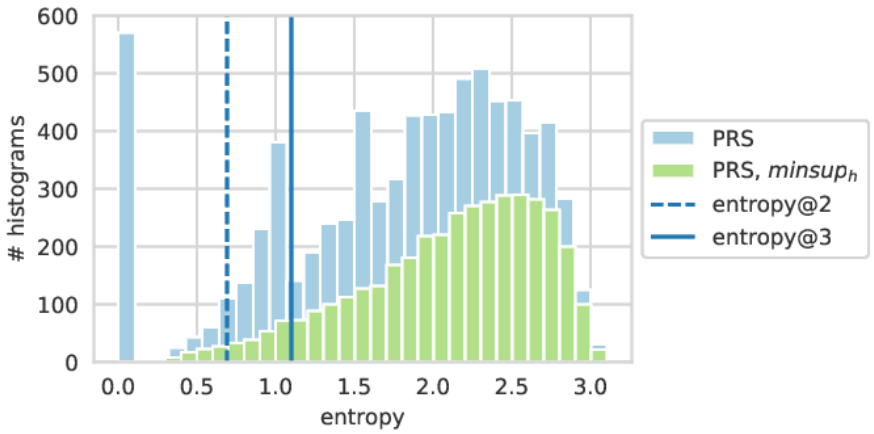


Figure 5.16: Entropy distribution of PRS histograms.

experiments in this section we conventionally fixed $minsup_h = 64$, computed with the median of the support distribution in log-scale (see Figure 5.14).

Focusing again on the PRS characteristics, Figure 5.16 provides the distribution of the histogram entropy before and after applying the $minsup_h$ threshold. The chart highlights how this filter removes the bin with close-to-zero entropy histograms, which are typically characterized by a low support (i.e., few noisy sample pairs).

PRS histograms are further filtered with the $maxentr_h$ threshold that allows selecting those with a representative distribution concentrated on few relative positions. For this purpose, we consider two reference histograms characterized by only 2 and 3 non-zero elements among the 9 position relationships. We calculate their entropy values, denoted as *entropy*@2 and *entropy*@3, and use them as reference for choosing $maxentr_h$. Figure 5.16 shows the position of these two reference values with vertical lines. Many of the collected histograms fall beyond the two maximum entropy values, as they contain high-entropy relationships. Specifically, with $minsup_h = 64$, $maxentr_h = entropy@2$ we obtain 77 histograms in PRS_f (i.e., 0.9% of the original ones). Instead, if we set $maxentr_h$ to *entropy*@3, the number of histograms in PRS_f is 277 (3%) . The value *entropy*@3 proves to be a better trade-off between the histogram representativity and their number. Hence, we select this threshold value for the following experiments.

To conclude the analysis of the PRS, Figure 5.17 provides a qualitative overview

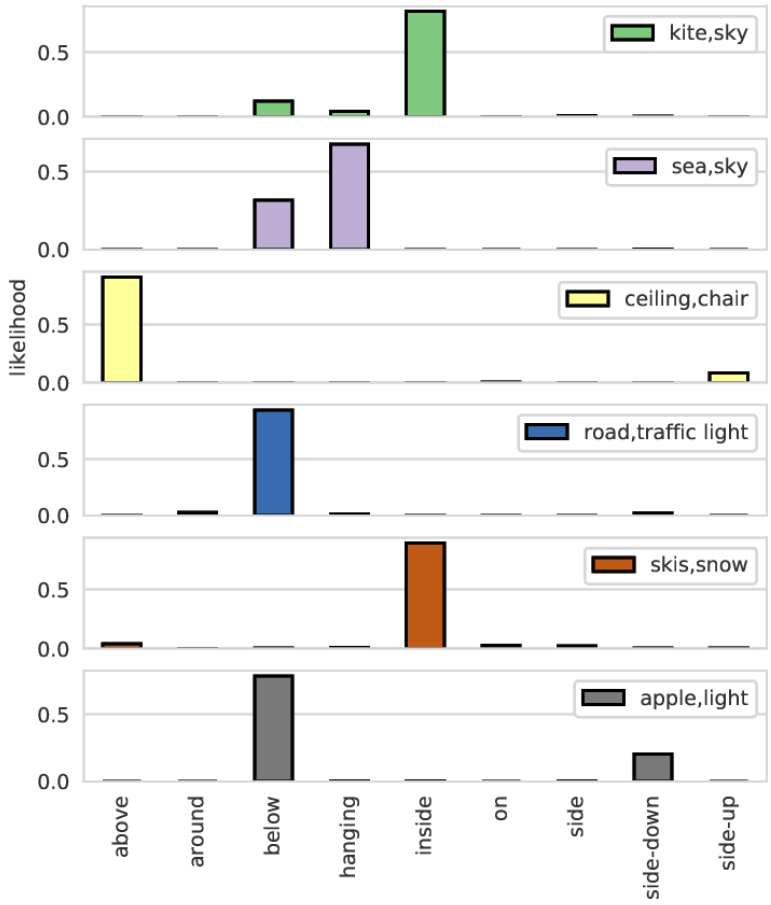


Figure 5.17: Example histograms in the PRS_f.

by showing some example histograms. The distributions in this chart show, for example, that the pair "sea, sky" satisfies the relationships *below* and *hanging*, while the pair "ceiling, chair" is mostly characterized by the relationship *above*.

5.11.2 Scene Graph Summary generation

In this section we conduct an ablation study of the building blocks of SImS. The analysis focuses on different aspects, such as running time, summary size, and quality. We separately assess (i) the graph preprocessing step, by turning on/off the edge and node pruning procedures, (ii) the FSM algorithm, changing between gSpan and SUBDUE, and (iii) the value of *minsup* in the case of gSpan. Table 5.2 reports the running time of the FSM process and some statistics about the output graphs. The first three lines of the table show the results for gSpan

Configuration					*Statistics*			
Config.	Alg.	Minsup	Edge pruning	Node pruning	Scene graph mining time	N. graphs	Avg. N. nodes	Std. N. nodes
1	gSpan	0.010	N	N	15h 55m	6184	5.29	2.05
2	gSpan	0.010	Y	N	4h 30m	186	4.20	2.22
3	gSpan	0.010	Y	Y	**3s**	276	3.11	0.97
4	gSpan	0.001	N	N	-			
5	gSpan	0.001	Y	Y	7s	9865	5.24	1.80
6	SUBDUE	-	Y	N	12h	184	19.89	8.94
7	SUBDUE	-	Y	Y	17m	48	6.15	2.35

Table 5.2: SGS generation results on whole COCO training set (118K images).

with *minsup* = 0.01, evaluating separately the graph preprocessing steps. The fourth and fifth lines decrease *minsup* to 0.001, while the last two lines provide the outcomes for SUBDUE.

Focusing on the running time of the frequent subgraph mining process, the first line of Table 5.2 confirms that the application of gSpan without any preprocessing takes a very long time (i.e., 16 hours). If we introduce edge pruning, the algorithm takes 4 hours and 30 minutes (config. 2), while, by activating also node pruning, it only requires 3 seconds. With a lower value of *minsup* (i.e., 0.001 in config. 4) and without any preprocessing, the FSM algorithm does not finish within 2 days. On the contrary, if we activate the two preprocessing steps, FSM converges in 7 seconds (config. 5 in Table 5.2). A similar conclusion can be drawn from config. 6 and 7, using SUBDUE. Indeed, the node pruning procedure manages to reduce the processing time from 12 hours to 17 minutes. Finally, these results show that gSpan presents the best time performance among the two FSM algorithms.

Let us inspect the effects of graph preprocessing on the amount of frequent graphs and the number of nodes. Table 5.2 provides the *Avg. N. nodes* and *Std. N. nodes* columns, which indicate the average and the standard deviation of the number of nodes in the SGS graphs. With gSpan, the edge pruning step diminishes the number of graphs from 6184 (config. 1) to 186 (config. 2). This is a first indicator of its ability of reducing redundancies in the SGS. Conversely, the application of node pruning slightly increases the amount of frequent graphs (from 186 in config. 2 to 276 in config. 3). Indeed, since the average number of nodes tends to decrease (from 5.29 in config. 1 to 3.11 in config. 3), the simplification of the scene graphs helps the FSM process to detect a higher number of frequent graphs. Instead,

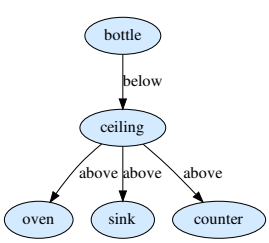
Configuration					*Statistics*	
Config.	Alg.	Minsup	Edge pruning	Node pruning	Coverage	Node diversity
1	gSpan	0.010	N	N	0.43	0.60
2	gSpan	0.010	Y	N	0.43	0.69
3	gSpan	0.010	Y	Y	0.43	**0.81**
4	gSpan	0.001	N	N	-	
5	gSpan	0.001	Y	Y	**0.48**	0.75
6	SUBDUE	-	Y	N	0.24	0.35
7	SUBDUE	-	Y	Y	0.24	0.38

Table 5.3: SGS generation results on whole COCO training set (118K images).

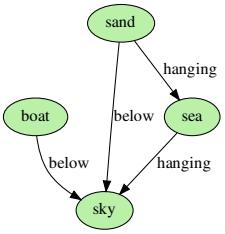
when decreasing the value of *minsup* to 0.001 (config. 5), the number of graphs increases (9865) and their average number of nodes climbs to 5.24. Bigger graphs can be more interesting to describe complex frequent scenes in the input images. SUBDUE graphs are few in number (48 in config. 7), but they tend to have a higher number of nodes (avg = 6.15) due to the nature of FSM algorithm.

To conclude this ablation analysis, we provide the values of coverage and node diversity in Table 5.3, computed as shown in Section 5.10. In the case of gSpan with *minsup* = 0.01 (config. 1-3), the value of coverage is fixed to 0.43 when varying the preprocessing configuration. This implies that node and edge pruning are able to simplify the FSM process, whilst maintaining coverage. Moreover, since the preprocessing step is designed to reduce redundancies in the SGS, the node diversity increases from 0.60 to 0.81. With *minsup* = 0.001 coverage increases to 0.48, while diversity becomes slightly lower (0.75) due to the higher number of graphs. Finally, SUBDUE proves to be the worst also in terms of summary quality. Indeed, it only reaches coverage 0.24 and diversity 0.38. Low coverage is due to the generation of bigger, hence more specific summary graphs. Indeed, this algorithm aims at finding the graphs which allow the best compression when they are substituted in the input collection with a single representative node. Also diversity is penalized by the presence of bigger graphs with few differences in terms of node labels.

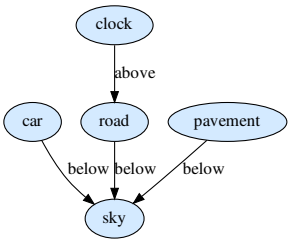
In conclusion, when applying SImS to the whole COCO dataset, config. 3 and config. 5 proved to be the best settings in terms of coverage, diversity and running time, also demonstrating the effectiveness of the preprocessing steps. Some



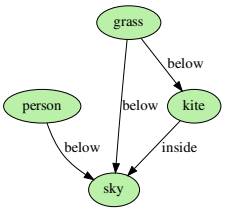
(a) Indoor scene, sup=138



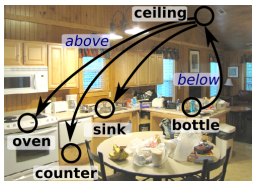
(b) Seaside, sup=140



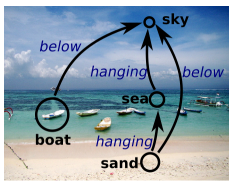
(c) City, sup=145



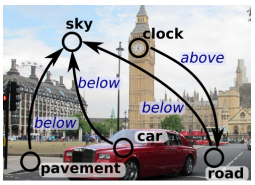
(d) Outdoor scene, sup=406



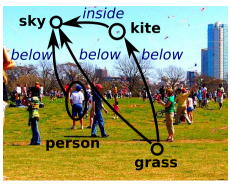
(e)



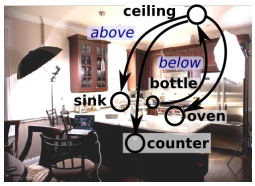
(f)



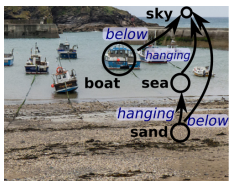
(g)



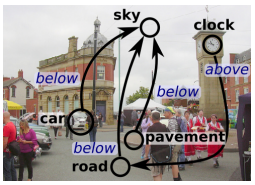
(h)



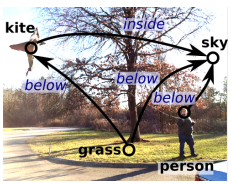
(i)



(j)



(k)



(l)

Figure 5.18: Examples of frequent graphs and represented images, minsup=0.001 (config. 5).

examples of scene graphs in the SGS generated by config. 5 are depicted in Figure 5.18 (a-d), while Figure 5.18 (e-l) shows some images extracted from the input collection that are represented by the previous scene graphs.

5.11.3 Comparison with other summarization techniques

We proceed the evaluation of SImS by comparing its results with a widely used summarization baseline. Since there are no available online resources, we implemented the technique proposed in [49], based on k-Medoids clustering applied to visual features. Following the standard implementation, we extract SIFT features from the input images and convert them to feature vectors of size 1000 by means of the Bag Of Words methodology described in Section 5.1.2. The k-Medoids clustering algorithm is then applied to the features matrix with different values of k, to inspect several summary sizes. The implementation of this baseline technique is available in our online repository.

Since k-Medoids is not scalable to big data collections, we conducted the comparison with SImS on two subsets of COCO. The images of the subsets are chosen based on their topic. Specifically, we selected the pictures according to the textual captions provided in the Microsoft COCO annotations. The first subset (i.e., Subset 1) includes the images whose caption contains the words "skiing" or "driving". Hence, this first experiment focuses on the inspection of common patterns among images representing a specific *action*. The second subset (i.e., Subset 2) is built by selecting images whose caption includes either "garden" or "church", focusing this time on the description of two different *places* that can be found in COCO. The two subsets contain 4865 and 890 images, respectively. Other subsets with different topics can be analyzed without significant changes in the final results.

The choice of exploiting image captions to build the subset, instead of considering scene graph object classes, allows a fair comparison between the two summarization methods. In particular, it avoids bias caused by a selection based on the same information contained in the scene graphs used by SImS, but that is not available to k-Medoids.

To compare the two summarization methods, we enforce the number of output summary elements to a fixed value k. In the case of k-Medoids, we use k to choose the number of medoids of the clustering process, while for SImS we exploit this value to select the top-k frequent graphs in the SGS. In the experiments, we analyzed k values in range $[2, 20]$. SImS algorithm is run with the hyperparameter setting shown in config. 3 (Table 5.2). The evaluation of the two techniques is performed by means of coverage and diversity of the output summaries, following the definitions

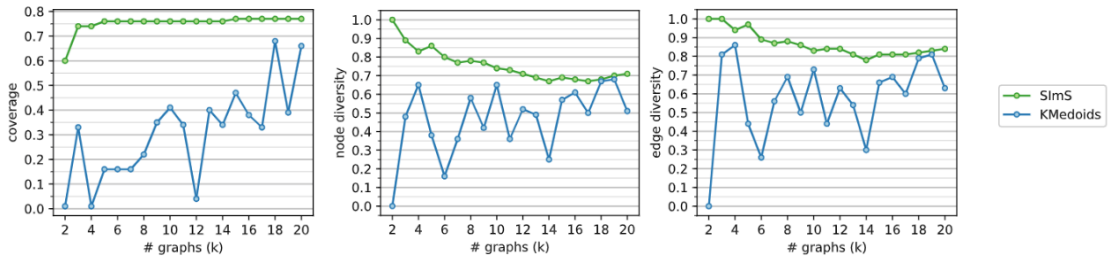


Figure 5.19: Quantitative comparison between SImS and k-Medoids on COCO Subset 1 (driving-skiing, 4865 images).

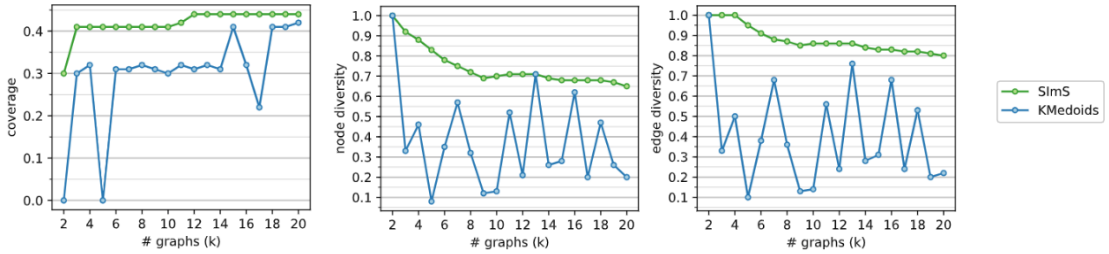


Figure 5.20: Quantitative comparison between SImS and k-Medoids on COCO Subset 2 (garden-church, 890 images).

presented in Section 5.10. Since k-Medoids does not output scene graphs, we adopt the following procedure for a fair comparison. We collect the k output images of k-Medoids, while for SImS we pick the k images from the output scene graphs, using the methodology described in Section 5.9. Afterwards, we extract the scene graphs from these two sets of images, following the algorithm shown in Section 5.6. Finally, we extract the most meaningful information with the node and edge pruning steps, then we compute coverage and diversity on the obtained scene graphs.

Figure 5.19 and 5.20 show the values of coverage and diversity for the two methods on Subset 1 and Subset 2, respectively. Due to the long running time, the results provided by k-Medoids were obtained with a single run of the algorithm and a fixed random seed. For Subset 1 both methods provide a higher coverage with respect to Subset 2. In Figure 5.19, SImS easily reaches a high coverage with few graphs ($k = 5$, coverage=0.76), while k-Medoids can touch coverage 0.68 only with $k = 18$. Also in Figure 5.20, SImS shows the best coverage for all the summary sizes. With $k = 12$ it reaches its best coverage value (0.44). Moreover, SImS node and edge diversity are always higher or equal to the ones of k-Medoids in both subsets. Typically, SImS node diversity tends to decrease with a higher number of

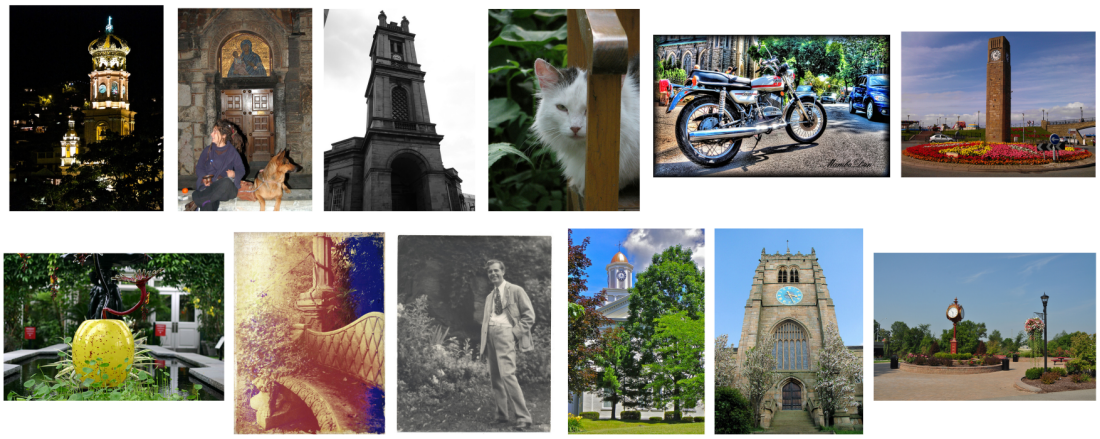
graphs. In Subset 1 node diversity touches its minimum value (0.67) at $k = 14$, while k-Medoids can only reach a maximum of 0.68 at $k = 19$. A similar trend is described by edge diversity, which also inspects the information carried by edge labels. Finally, SImS shows better diversity also in Subset 2.

We conclude the analysis with a qualitative comparison between the two methods. Figure 5.21 and 5.22 provide the output images for Subset 2 (with $k = 5$) and Subset 3 (with $k = 12$), respectively. In Figure 5.21, both methods depict images containing people skiing and cars/trucks. SImS also highlights the most significant objects in these images, providing the frequent patterns. K-Medoids shows a lower coverage as it includes objects with a lower frequency in the input collection. For example, a picture with a man on a carriage (i.e., driving topic) appears in Figure 5.21a. This object is not included in SImS results, as its class is not frequent in the dataset. In Subset 2, SImS highlights how cars and clock towers are frequent

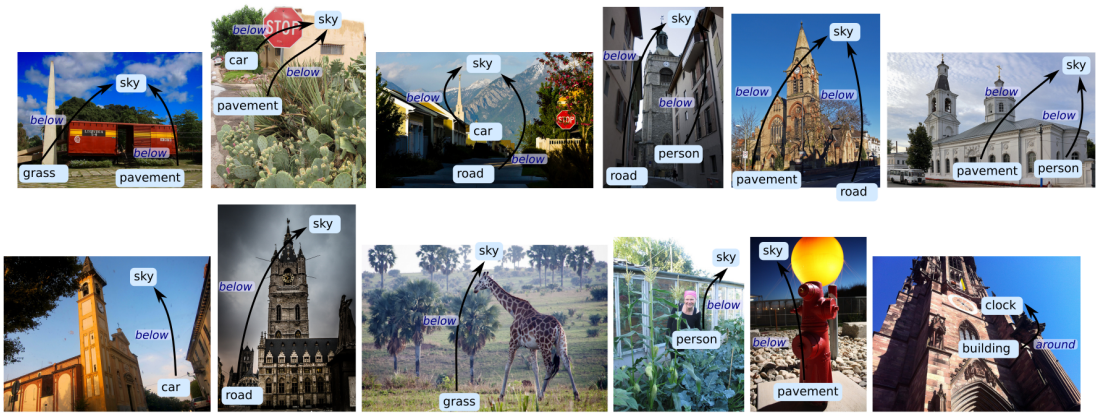
(a) KMedoids summary.

(b) SImS summary.

Figure 5.21: Qualitative comparison on COCO Subset 1 (“skiing”, “driving”), with 5 summary elements.



(a) KMedoids summary.



(b) SImS summary.

Figure 5.22: Qualitative comparison on COCO Subset 2 (“church”, “garden”), with 12 summary elements.

items appearing in the collection (Figure 5.22b), which is not immediately visible from the k-Medoids images (Figure 5.22a).

Chapter 6

Conclusion and future works

This thesis contributes to the state-of-the-art in image understanding by providing novel approaches based on semantics. In these chapters, we claimed and proved that an in-depth analysis of the objects inside images and their semantic relationships is fundamental to enhance the comprehension of visual data. We divided our discussion into three correlated topics: (i) Detection of the relative position between objects, (ii) anomaly detection in semantic segmentation, and (iii) image collection summarization.

In Chapter 3, we laid the foundation for the automatic derivation of object relationships. Specifically, we focused on the definition of 9 fine-grained relative position labels, capable of, for example, distinguishing between "on with" and "on without" contact, or between "side, side-up, side-down". Our methodology demonstrated to be able of analyzing the relationships between object shapes, without simplifying them to centroids or bounding boxes, as in previous works. To this aim, our technique was designed to process either semantic segmentation or panoptic segmentation annotations, which provide a very detailed description of the object shapes. Since semantic segmentation does not include any information about object instances, we proposed to inspect them by means of a connected-components detection algorithm. The labeled instances were then exploited to derive a novel set of string-based features, which are able to capture fine details about the shapes of the different objects. Furthermore, bounding-box-based features integrated the string representation to identify the object positions when they are not vertically aligned. The final relative position classifier, based on random forests, achieved high F-score

and proved to be capable of correctly distinguishing between the previously defined fine-grained relationships.

Object relationships, such as relative position, size, and co-occurrence, can be exploited for identifying possible classification errors made by segmentation neural networks. In Chapter 4 we presented SAD (Semantic Anomaly Detection), a technique for anomaly detection based on an automatically-derived, interpretable, knowledge base. Our novel methodology inspects a set of ground-truth images, labeled with semantic segmentation, and extracts a set of interpretable rules describing the average behavior of the different objects. These rules, stored in the knowledge base, were defined by means of histograms representing discrete probability distributions. After the training process, the knowledge base was exploited to inspect anomalies in semantic segmentation results. Specifically, object pairs that do not follow the behavior of normal instances are deemed to be possible classification errors. We defined and tested three anomaly detection methods, showing that we are able to identify objects with a low pixel accuracy predicted by the segmentation model.

The last topic of this thesis was focused on a further application of object relationships. In Chapter 5, we demonstrated the usage of scene graphs, specifying object relative positions, in a research field called image collection summarization. We proposed SImS (Semantic Image Summarization), which can automatically build a novel type of summaries based on scene graphs. First, the relative position classifier defined in Chapter 3 was exploited to build scene graphs starting from panoptic segmentation, where nodes represent objects and edges model the position relationships. Afterwards, SImS extracts two types of patterns: (i) the PRS (Pairwise Relationship Summary), inspecting common pairwise object relationships, and (ii) the SGS (Scene Graph Summary), inspecting frequent multi-object patterns. The application of frequent subgraph mining techniques to the scene graphs showed to be an effective way of deriving summarization patterns from the input collection. We demonstrated that our technique, relying on the semantic description of the image content, is able to overcome previous methods in terms of coverage and diversity. Moreover, the final results provided by SImS enjoy a higher interpretability, thanks to the presentation of frequent scene graphs paired with example images. Finally, the designed preprocessing technique, including edge and node pruning, has proved to be effective in reducing running time and improving summary diversity.

Future works may extend the semantic content of the object relationships used by SAD and SImS. For example, the semantics of our graphs, nowadays focused on position relationships, could be integrated by actions (e.g., "racket hits ball") and object properties (e.g., "a grey van"). In the case of SAD, also external ontologies could be used as prior knowledge to enhance the content of the knowledge base and model more complex object relationships. The SAD approach, which is a general methodology for detecting anomalies, could be also experimented in other domains, e.g., textual documents. A further possible application of SAD could be the prediction of the quality of segmented images based on the detected anomalies, useful to assess predictions when ground truth is not available. In the case of SImS, further improvements could be brought by post-processing the SGS summaries to enhance coverage and diversity. Also other semantic approaches for pairing the images with the output frequent scene graphs could improve the results. Finally, semantic hierarchies among the SGS graphs could be introduced for a better and interactive visualization.

Bibliography

[1] Tarek Abbes, Adel Bouhoula, and Michael Rusinowitch. “Efficient decision tree for protocol analysis in intrusion detection”. In: *International Journal of Security and Networks* 5.4 (2010), pp. 220–235.

[2] Somak Aditya, Yezhou Yang, and Chitta Baral. “Integrating knowledge and reasoning in image understanding”. In: *arXiv preprint arXiv:1906.09954* (2019).

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. “Mining association rules between sets of items in large databases”. In: *Acm sigmod record*. Vol. 22. 2. ACM. 1993, pp. 207–216.

[4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. “A survey of network anomaly detection techniques”. In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[5] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. “A survey of anomaly detection techniques in financial domain”. In: *Future Generation Computer Systems* 55 (2016), pp. 278–288.

[6] Oron Ashual and Lior Wolf. “Specifying object attributes and relations in interactive scene generation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4561–4569.

[7] Muhammad Nabeel Asim et al. “A survey of ontology learning techniques and applications”. In: *Database* 2018 (2018).

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[9] Moshe Bar. "Visual objects in context". In: *Nature Reviews Neuroscience* 5.8 (2004), pp. 617–629.

[10] Daniel Barbará et al. "ADAM: a testbed for exploring the use of data mining in intrusion detection". In: *ACM Sigmod Record* 30.4 (2001), pp. 15–24.

[11] Sean Bechhofer et al. "OWL web ontology language reference". In: *W3C recommendation* 10.02 (2004).

[12] Sean Bell et al. "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2874–2883.

[13] Irving Biederman, Robert J Mezzanotte, and Jan C Rabinowitz. "Scene perception: Detecting and judging objects undergoing relational violations". In: *Cognitive psychology* 14.2 (1982), pp. 143–177.

[14] MHT de Boer et al. "Applying semantic reasoning in image retrieval". In: (2015).

[15] Oren Boiman and Michal Irani. "Detecting irregularities in images and in video". In: *International journal of computer vision* 74.1 (2007), pp. 17–31.

[16] Sarah Boslaugh. *Statistics in a nutshell: A desktop quick reference*. " O'Reilly Media, Inc.", 2012.

[17] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[18] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "Coco-stuff: Thing and stuff classes in context". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1209–1218.

[19] Jorge Camargo, Fabio González, and Rodolfo Torres. "Visualization, summarization and exploration of large collections of images: State of the art". In: *Latin-American Conference On Networked and Electronic Media. LACNEM*. 2009.

[20] Jorge E Camargo and Fabio A González. "Multimodal latent topic analysis for image collection summarization". In: *Information Sciences* 328 (2016), pp. 270–287.

[21] Shi-Kuo Chang et al. “An intelligent image database system”. In: *IEEE Transactions on Software Engineering* 14.5 (1988), pp. 681–688.

[22] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[23] Xinlei Chen and Abhinav Gupta. “Spatial memory for context reasoning in object detection”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4086–4096.

[24] Myung Jin Choi et al. “Exploiting hierarchical context on a large database of object categories”. In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conf. on*.

[25] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. “Multi-column deep neural networks for image classification”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[26] Diane J Cook and Lawrence B Holder. “Substructure discovery using minimum description length and background knowledge”. In: *Journal of Artificial Intelligence Research* 1 (1993), pp. 231–255.

[27] Luigi P Cordella et al. “A (sub) graph isomorphism algorithm for matching large graphs”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.10 (2004), pp. 1367–1372.

[28] Marius Cordts et al. “The Cityscapes Dataset”. In: *CVPR Workshop on The Future of Datasets in Vision*. 2015.

[29] Gabriela Csurka et al. “What is a good evaluation measure for semantic segmentation?.” In: *BMVC*. Vol. 27. Citeseer. 2013, p. 2013.

[30] Stanislas Dehaene. *Consciousness and the brain: Deciphering how the brain codes our thoughts*. Penguin, 2014.

[31] Da Deng. “Content-based image collection summarization and comparison using self-organizing maps”. In: *Pattern Recognition* 40.2 (2007), pp. 718–727.

[32] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[33] Santosh K Divvala et al. “An empirical study of context in object detection”. In: *2009 IEEE Conference on computer vision and Pattern Recognition*. IEEE. 2009, pp. 1271–1278.

[34] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[35] Jianping Fan et al. “A novel approach to enable semantic and visual image summarization for exploratory image search”. In: *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. 2008, pp. 358–365.

[36] Martin A Fischler and Robert A Elschlager. “The representation and matching of pictorial structures”. In: *IEEE Transactions on computers* 100.1 (1973), pp. 67–92.

[37] Matthew Fisher, Manolis Savva, and Pat Hanrahan. “Characterizing structural relationships in scenes using graph kernels”. In: *ACM SIGGRAPH 2011 papers*. 2011, pp. 1–12.

[38] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. “An approach to spacecraft anomaly detection problem using kernel feature space”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 401–410.

[39] Carolina Galleguillos and Serge Belongie. “Context based object categorization: A critical survey”. In: *Computer vision and image understanding* 114.6 (2010), pp. 712–722.

[40] Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. “Object categorization using co-occurrence, location and appearance”. In: *Computer Vision and Pattern Recognition (CVPR), 2008. IEEE Conf. on*. IEEE, pp. 1–8.

[41] Spyros Gidaris and Nikos Komodakis. “Object detection via a multi-region and semantic segmentation-aware cnn model”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1134–1142.

[42] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[43] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[45] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. In: *AI magazine* 38.3 (2017), pp. 50–57.

[46] Palash Goyal et al. “Cross-modal Learning for Multi-modal Video Categorization”. In: *arXiv preprint arXiv:2003.03501* (2020).

[47] Riccardo Guidotti et al. “A survey of methods for explaining black box models”. In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[48] Büsra Güvenoglu and Belgin Ergenç Bostanoglu. “A qualitative survey on frequent subgraph mining”. In: *Open Computer Science* 8.1 (2018), pp. 194–209.

[49] Youssef Hadi, Fedwa Essannouni, and Rachid Oulad Haj Thami. “Video summarization by k-medoid clustering”. In: *Proceedings of the 2006 ACM symposium on Applied computing*. 2006, pp. 1400–1401.

[50] Michael A Hayes and Miriam AM Capretz. “Contextual anomaly detection in big sensor data”. In: *Big Data (BigData Congress), 2014 IEEE Int. Congress on*. IEEE. 2014, pp. 64–71.

[51] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[52] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[53] David Heckerman. “The certainty-factor model”. In: *Encyclopedia of Artificial Intelligence,* (1992), pp. 131–138.

[54] Derek Hoiem et al. “Recovering occlusion boundaries from a single image”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[55] Han Hu et al. “Relation networks for object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3588–3597.

[56] Chuntao Jiang, Frans Coenen, and Michele Zito. “A survey of frequent subgraph mining algorithms”. In: *The Knowledge Engineering Review* 28.1 (2013), pp. 75–105.

[57] Justin Johnson, Agrim Gupta, and Li Fei-Fei. “Image generation from scene graphs”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1219–1228.

[58] Gunhee Kim, Seungwhan Moon, and Leonid Sigal. “Joint photo stream and blog post summarization and exploration”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3081–3089.

[59] Alexander Kirillov et al. “Panoptic feature pyramid networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6399–6408.

[60] Alexander Kirillov et al. “Panoptic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 9404–9413.

[61] Teuvo Kohonen. *Self-organizing maps*. Vol. 30. Springer Science & Business Media, 2012.

[62] Ranjay Krishna et al. “Visual genome: Connecting language and vision using crowdsourced dense image annotations”. In: *International Journal of Computer Vision* 123.1 (2017), pp. 32–73.

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

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[65] L’ubor Ladický et al. “Associative hierarchical crfs for object class image segmentation”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 739–746.

[66] L'ubor Ladický et al. "What, where and how many? combining object detectors and crfs". In: *European conference on computer vision*. Springer. 2010, pp. 424–437.

[67] Rikard Laxhammar, Goran Falkman, and Egils Sviestins. "Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator". In: *Information Fusion, 2009. FUSION'09. 12th Int. Conf. on*. IEEE. 2009, pp. 756–763.

[68] Justin Lazarow et al. "Learning instance occlusion for panoptic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10720–10729.

[69] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[70] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[71] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[72] Yong Jae Lee and Kristen Grauman. "Object-graphs for context-aware visual category discovery". In: *IEEE transactions on pattern analysis and machine intelligence* 34.2 (2012), pp. 346–358.

[73] Jianan Li et al. "Attentive contexts for object detection". In: *IEEE Transactions on Multimedia* 19.5 (2016), pp. 944–954.

[74] Yingbo Li and Bernard Merialdo. "VERT: automatic evaluation of video summaries". In: *Proceedings of the 18th ACM international conference on Multimedia*. 2010, pp. 851–854.

[75] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[76] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[77] Qi Liu et al. “Unsupervised detection of contextual anomaly in remotely sensed data”. In: *Remote Sensing of Environment* 202 (2017), pp. 75–87.

[78] Yong Liu et al. “Structure inference net: Object detection using scene-level context and instance-level relationships”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6985–6994.

[79] Vijay Mahadevan et al. “Anomaly detection in crowded scenes”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1975–1981.

[80] Tomasz Malisiewicz and Alyosha Efros. “Beyond categories: The visual memex model for reasoning about object relationships”. In: *Advances in neural information processing systems*. 2009, pp. 1222–1230.

[81] Bangalore S Manjunath and Wei-Ying Ma. “Texture features for browsing and retrieval of image data”. In: *IEEE Transactions on pattern analysis and machine intelligence* 18.8 (1996), pp. 837–842.

[82] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[83] Michel Minoux. “Accelerated greedy algorithms for maximizing submodular set functions”. In: *Optimization techniques*. Springer, 1978, pp. 234–243.

[84] “MIT Scene Parsing Benchmark, http://sceneparsing.csail.mit.edu/”. In: (2016).

[85] Heesoo Myeong, Ju Yong Chang, and Kyoung Mu Lee. “Learning object relationships via graph-based context model”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2727–2734.

[86] Aude Oliva and Antonio Torralba. “Building the gist of a scene: The role of global image features in recognition”. In: *Progress in brain research* 155 (2006), pp. 23–36.

[87] Devi Parikh, C Lawrence Zitnick, and Tsuhan Chen. “Exploring tiny images: The roles of appearance and contextual information for machine and human object recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.10 (2011), pp. 1978–1991.

[88] Andrea Pasini and Elena Baralis. "Detecting Anomalies in Image Classification by Means of Semantic Relationships". In: *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE. 2019, pp. 231–238.

[89] Andrew Rabinovich et al. "Objects in context". In: *Computer vision, 2007. ICCV 2007. IEEE 11th int. conf. on*. IEEE. 2007, pp. 1–8.

[90] T Ramraj and R Prabhakar. "Frequent subgraph mining algorithms-a survey". In: *Procedia Computer Science* 47 (2015), pp. 197–204.

[91] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[92] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[93] Wei Ren, Maneesha Singh, and Sameer Singh. "Image retrieval using spatial context". In: *Proceedings of the 9th international workshop on systems, signals and image processing*. 2002, pp. 44–49.

[94] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[95] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[96] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[97] Fereshteh Sadeghi, Santosh K Kumar Divvala, and Ali Farhadi. "Viske: Visual knowledge extraction and question answering by visual verification of relation phrases". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1456–1464.

[98] Zahra Riahi Samani and Mohsen Ebrahimi Moghaddam. "A knowledge-based semantic approach for image collection summarization". In: *Multimedia Tools and Applications* 76.9 (2017), pp. 11917–11939.

[99] Brigit Schroeder, Subarna Tripathi, and Hanlin Tang. "Triplet-Aware Scene Graph Embeddings". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[100] Sebastian Schuster et al. "Generating semantically precise scene graphs from textual descriptions for improved image retrieval". In: *Proceedings of the fourth workshop on vision and language*. 2015, pp. 70–80.

[101] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[102] Edward H Shortliffe and Bruce G Buchanan. "A model of inexact reasoning in medicine". In: *Mathematical biosciences* 23.3-4 (1975), pp. 351–379.

[103] Ian Simon, Noah Snavely, and Steven M Seitz. "Scene summarization for online image collections". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

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

[105] Anurag Singh and Deepak Kumar Sharma. "Image Collection Summarization: Past, Present and Future". In: *Data Visualization and Knowledge Engineering*. Springer, 2020, pp. 49–78.

[106] Pinaki Sinha. "Summarization of archived and shared personal photo collections". In: *Proceedings of the 20th international conference companion on World wide web*. 2011, pp. 421–426.

[107] John R Smith et al. "Decoding image semantics using composite region templates". In: *Content-Based Access of Image and Video Libraries, 1998. Proceedings. IEEE Workshop on*. IEEE. 1998, pp. 9–13.

[108] Robyn Speer, Joshua Chin, and Catherine Havasi. "Conceptnet 5.5: An open multilingual graph of general knowledge". In: *arXiv preprint arXiv:1612.03975* (2016).

[109] Hao Su, Jia Deng, and Li Fei-Fei. "Crowdsourcing annotations for visual object detection". In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[110] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[111] 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.

[112] Pang-Ning Tan et al. *Introduction to data mining, 2nd Edition*. 2018.

[113] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. "Big data analytics for network anomaly detection from netflow data". In: *Computer Science and Engineering (UBMK), 2017 Int. Conf. on*. IEEE. 2017, pp. 592–597.

[114] Subarna Tripathi et al. "Compact scene graphs for layout composition and patch retrieval". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[115] Sebastian Tschiatschek et al. "Learning mixtures of submodular functions for image collection summarization". In: *Advances in neural information processing systems*. 2014, pp. 1413–1421.

[116] Huiyu Wang et al. "MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers". In: *arXiv preprint arXiv:2012.00759* (2020).

[117] Yuwen Xiong et al. "Upsnet: A unified panoptic segmentation network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8818–8826.

[118] Xifeng Yan and Jiawei Han. "gspan: Graph-based substructure pattern mining". In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE. 2002, pp. 721–724.

[119] Chunlei Yang et al. "Image collection summarization via dictionary learning for sparse representation". In: *Pattern Recognition* 46.3 (2013), pp. 948–961.

[120] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[121] Xingyu Zeng et al. “Crafting gbd-net for object detection”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.9 (2017), pp. 2109–2123.

[122] Zheng Zhang et al. “HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification”. In: *Proc. IEEE Workshop on Information Assurance and Security*. 2001, pp. 85–90.

[123] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890.

[124] Ye Zhao, Richang Hong, and Jianguo Jiang. “Visual summarization of image collections by fast RANSAC”. In: *Neurocomputing* 172 (2016), pp. 48–52.

[125] Bolei Zhou et al. “Scene parsing through ADE20k dataset”. In: *Proc. CVPR*. 2017.

[126] Bolei Zhou et al. “Scene parsing through ade20k dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 633–641.

[127] Yousong Zhu et al. “Couplenet: Coupling global structure with local parts for object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4126–4134.

[128] Honglei Zhuang et al. “Identifying semantically deviating outlier documents”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 2748–2757.

[129] C Lawrence Zitnick and Devi Parikh. “Bringing semantics into focus using visual abstraction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3009–3016.