UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

Master's degree in ICT for Internet and multimedia

OBJECT RECOGNITION IN DYNAMIC ENVIRONMENTS FOR BILLBOARD DETECTION AND CAMPAIGN VALIDATION

Supervisor
Professor Pietro Zanuttigh

Company tutor
Dott. Francesco Coccia

Master Candidate
Alberto Grimaldi

Academic Year
2022 – 2023

Abstract

This thesis work was developed during an internship carried out at *Technology Reply S.R.L.*
The aim of this work is the development of an automatic detection and validation system for advertising posters. Starting from a set of images showing advertising panels belonging to a single campaign, the target is to identify the outlier, i.e., the advertising panel that is supposed to belong to the same campaign but has been mislabelled. The system was divided into two macro modules according to the task at hand: the object detection module and the similarity estimation and outlier detection module. The object detection module was used to detect the advertising panels in the images and isolate them from the surrounding context, thus focusing attention on the most important part of the image, i.e., the billboard. The similarity estimation and outlier detection module was the most difficult to implement. It consists of a series of blocks assigned to different functions, in which the image is reduced to a feature descriptors vector, compared with all others to measure its similarity, analysed and then classified as an outlier or non-outlier. Several tests were conducted which showed a good ability of the system to classify images but also revealed critical issues to be addressed in any future developments.

Sommario

Questa progetto di tesi è stato sviluppato durante uno stage svolto presso l'azienda *Technology Reply S.R.L.*
L'obiettivo di questo lavoro è lo sviluppo di un sistema di rilevazione e validazione automatica dei manifesti pubblicitari. Partendo da un insieme di immagini che mostrano pannelli pubblicitari appartenenti a una singola campagna, l'obiettivo è quello di identificare l'outlier, cioè il pannello pubblicitario che dovrebbe appartenere alla stessa campagna ma che è stato in realtà etichettato in modo errato. Il sistema è stato suddiviso in due macro-moduli in base al compito da svolgere: il modulo di rilevamento degli oggetti e il modulo di stima della somiglianza e rilevamento degli outlier. Il modulo di rilevamento degli oggetti è stato utilizzato per individuare i pannelli pubblicitari nelle immagini e isolarli dal contesto circostante, concentrando così l'attenzione sulla parte più importante dell'immagine, ovvero il cartellone pubblicitario. Il modulo di stima della somiglianza e di rilevamento degli outlier è stato il più difficile da implementare. È costituito da una serie di blocchi aventi diverse funzioni, in cui l'immagine viene ridotta a un vettore di descrittori di caratteristiche, confrontata con tutte le altre per misurarne la somiglianza, analizzata e quindi classificata come outlier o non outlier. Sono stati condotti diversi test che hanno mostrato una buona capacità del sistema di classificare le immagini, ma hanno anche rivelato criticità da affrontare in eventuali sviluppi futuri.

Contents

Chapter 1

Introduction

The interest in artificial intelligence has grown considerably in recent years on behalf of the scientific community. According to the latest AI Annual Index Report of Stanford University, after growing only slightly from 2010 to 2015, the number of AI journal publications grew 2.3 times since 2015. From 2020 to 2021, they increased by 14.8%.

Computer vision is a field of Artificial Intelligence that allows computers and systems to derive meaningful information from digital images, videos, and other visual inputs and take actions or make recommendations based on that information. If artificial intelligence enables computers to think, computer vision enables them to see, observe, and understand (IBM Corporation, 2023). There are numerous real-world uses of computer vision technology, including autonomous driving, crowd surveillance, sports analytics, and the development of video games. There are many different problems to be solved in this area, which is where academic and industrial research has been focusing most of its efforts recently. Just to give a few examples, it is worth mentioning the Image Classification task which is the ability of machines to

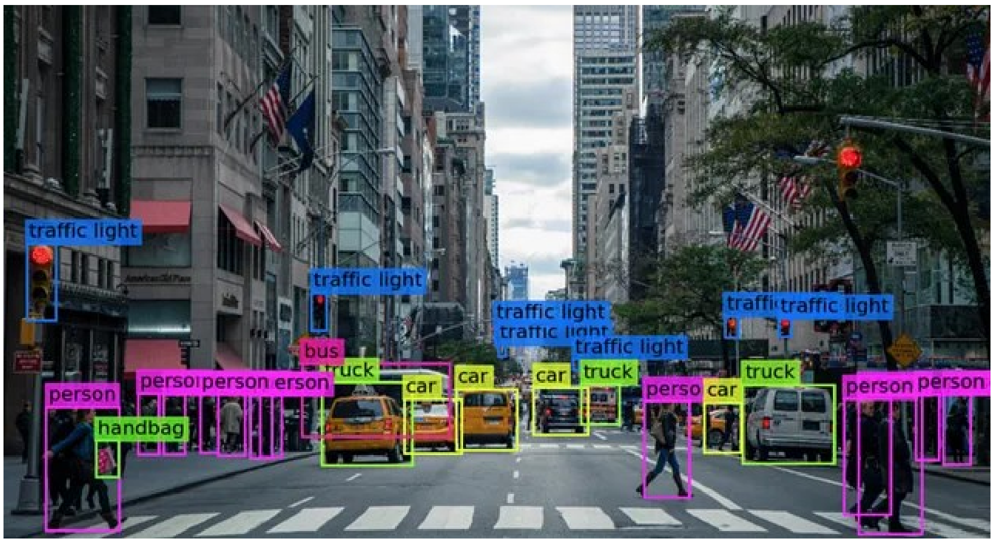


Figure 1.1: YOLO Multi-Object Detection and Classification.

categorize objects in images, Face Detection and Recognition whereby the system can detect faces or individuals in images or videos, Object Detection which is the challenge of identifying and localizing objects within an image or video and Image Similarity detection to find out the similarity between a query image and potential candidates (Stanford Institute for Human-Centered Artificial Intelligence, 2023).
The ever-increasing computing power of computers combined with the enormous availability of digital data produced by electronic devices has made it possible to achieve ever higher levels of model accuracy and reliability. As a result, there has been a growing interest on the part of the industry in creating new solutions based on Artificial Intelligence and, above all, computer vision to improve production processes, increase execution speed and enhance the user experience.

The problem I want to address concerns the development of a validation system for advertising campaigns from a set of images of outdoor billboards. In the image set, the advertising campaign should be the same for all images. However, an image can be found that shows a brand that is different from the rest of the dataset and therefore represents our outlier. The aim is then to detect this outlier.
The best way to tackle this task is to divide it into two sub-tasks and exploit computer vision techniques to analyse the images. The first step should consist of detecting the panel in the image, using Object Detection, to focus attention only where it is needed, leaving out the background context. The second step, and the most important one, concerns the detection of similarities between billboards. By quantitatively extracting how much the images resemble each other, we would be able to identify the outlier i.e., the image that least resembles the others.
What is just mentioned above is a simplified version of the case study that will be addressed in later chapters. The main focus of this research is therefore to be able to express the concept of similarity between images from a computational point of view so that the model can distinguish an image that has been labelled with the correct brand from one that has not. Specifically, we will say that two images are similar if the advertising panels within them depict the same campaign. The similarity between two or more images can be something complicated to express computationally. This requires efficient feature extraction and accurate statistical analysis.

The final objective of this thesis is to develop a semi-automatic validation system for advertising images for an Italian outdoor advertising certification company. Every two weeks, a huge collection (thousands) of images taken from the bill posters is sent to the cloud platform of Technology Reply to prove the correct position of the billboards. Today, a user manually verifies the validation through a dedicated GUI. In the following chapters, we will see how an AI engine has been developed to automate this certification process by detecting the posters in the image

and validating the poster's subject.
The work was divided into two main tasks: the object detection task and the similarity detection task. The following chapters are therefore organised according to this subdivision.

Chapter 2

First task: Billboard detection

In this chapter, an overview of the general concepts of Object Detection is given, followed by a technical introduction to the dataset and the framework used for the development of the model, and finally the first task to achieve the project's goal, i.e. the detection of panels from images, is dealt with.

2.1 Object Detection Frameworks review

Object detection entails finding the position and borders of items in a picture and categorizing the objects. It is a field of Computer Vision that has been extensively studied and analyzed in past years and in fact, many algorithms have been designed to tackle this task in all its facets. For this reason, in this section, we will limit ourselves to presenting the state-of-the-art frameworks appearing since Deep Learning entered the field. These can be organized into two main categories:

- Two-stage methods: higher detection accuracy but typically slower, characterized by a preprocessing step for generating object proposals
- One-stage methods: faster but with a lower detection accuracy, based on a single-step process of detection

2.1.1 Two-stage Frameworks

In a region-based framework, category-independent region proposals are generated from an image, Convolutional Neural Network (CNN) features are extracted from these regions, and then category-specific classifiers are used to determine the category labels of the proposals (Liu, 2019).

Regions with CNN features (R-CNN), proposed by *Girshick et al.* (Girshick, 2014), integrate AlexNet with a process called Selective Search for the region proposals.

The main stage pipelines of R-CNN can be summarized as follows:

1. *Computation of bounding boxes (region proposals)* Agnostic region proposals, which are candidate regions that might contain objects, are obtained via a selective search (Liu, 2019).

2. *Selective search* Looks at the image through windows of various sizes, and for each size tries to group together adjacent pixels (by texture, color, intensity) to identify objects.
3. *CNN model finetuning* Once the proposals are created, they are warped into a standard square size and passed it through to a CNN.
4. *SVM Classifier* On the final layer of the CNN, R-CNN includes a Support Vector Machine (SVM) that classifies whether this is an object, and if so what object.

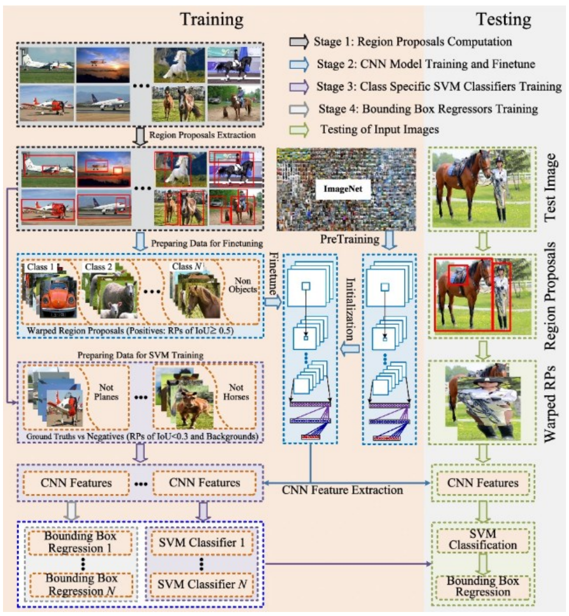

Training
Testing
Stage 1: Region Proposals Computation
Stage 2: CNN Model Training and Finetune
Stage 3: Class Specific SVM Classifiers Training
Stage 4: Bounding Box Regressors Training
Testing of Input Images
Region Proposals Extraction
ImageNet
Test Image
Preparing Data for Finetuning
PreTraining
Class 1
Class 2
Class N
Non Objects
Finetune
Initialization
Region Proposals
Warped Region Proposals (Positives: RPs of IoU≥ 0.5)
Preparing Data for SVM Training
Not Planes
Not Horses
Warped RPs
Ground Truths vs Negatives (RPs of IoU<0.3 and Backgrounds)
CNN Features
CNN Features
CNN Feature Extraction
CNN Features
Bounding Box Regression 1
SVM Classifier 1
SVM Classification
Bounding Box Regression N
SVM Classifier N
Bounding Box Regression

Figure 2.1: Illustration of the R-CNN detection framework (Liu, 2019)

R-CNN can reach remarkably high accuracy but has a notable slow computational time because the CNN feature needs to be extracted from each object proposal (Liu, 2019).

Fast R-CNN was proposed to address some of the drawbacks of R-CNN. It uses a technique called RoIPool (Region of Interest Pooling) that shares the forward CNN pass for an image across its sub-regions. RoI pooling uses warping at the feature level to approximate warping at the image level. In this way, the CNN features for each region are obtained by selecting a corresponding region from the CNN's feature map requiring only one CNN pass of the original image. Another insight of Fast R-CNN is that it enables end-to-end detector training by supplying a simplified training procedure that concurrently learns a SoftMax classifier and bounding box regressor, as opposed to R-CNN, which individually trains a SoftMax classifier, SVMs, and

Bounding Box Regressors (Liu, 2019).
Fast R-CNN has considerably speeded up R-CNN but still relies on region proposals, the calculation of which soon became its bottleneck.

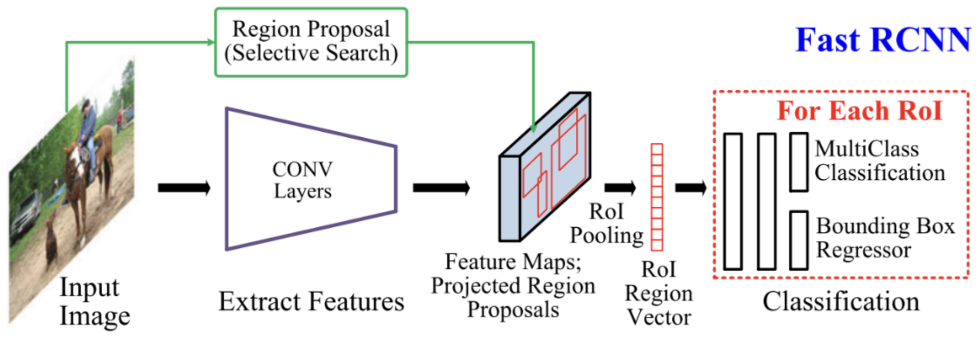


Figure 2.2: Fast R-CNN training Framework (Liu, 2019)

The Faster R-CNN framework proposed by *Ren et al.* offered a further improvement by replacing the slow Selective Search algorithm with a neural network, the Region Proposal Network (RPN). Through RPN the model learns itself the interesting region where to look for objects.

2.1.2 One-stage Frameworks

One-stage pipelines refer to architectures that directly predict the object class and bounding boxes from the full image with a single feed-forward CNN in a large single block setting, with no need of involving region proposals (Liu, 2019), so they are typically faster.

YOLO (You Only Look Once) was introduced by (Joseph Redmon, 2016). It removes the region proposals step and models the detection as a simple regression problem taking the image in input and directly predicting detections using a small set of candidate regions. The image is divided into a grid of S x S and for each location, YOLO predicts C class probabilities, N bounding boxes, and confidence scores. The confidence reflects the accuracy of the bounding box and whether it actually contains an object. Eventually, S x S x N boxes are predicted but many of them are discarded by applying a threshold on the confidence score. Due to the division of the image into a grid, YOLO makes many localization errors and has difficulty locating small objects, also because it only predicts one type of object per grid location.
Since its introduction, YOLO has evolved through several versions, with each version improving upon the previous one in speed, accuracy, and functionality. YOLOv8 is

the latest version of the algorithm provided by Ultralytics. It is the fastest and most accurate YOLO model to date, achieving the state-of-the-art (SOTA) results in many benchmarks. YOLOv8 builds on the success of previous versions, introducing new features and improvements such as:

- Improved accuracy and speed
- A new backbone network based on EfficientNet, which improves the model's ability to capture high-level features
- The possibility to include features from multiple scales
- Enhanced data augmentation techniques
- Relatively low number of parameters, making it efficient and scalable (YOLOv8, s.d.)

SSD (Single Shot Detector) was proposed by *Liu et al.* (Wei Liu, 2016), to be faster than YOLO and with an accuracy comparable to region-based detectors like Faster R-CNN. Like in YOLO, SSD still predicts a fixed number of bounding boxes and scores but performs detections over multiple scales by operating on multiple convolutional layers, progressively decreasing in size, to predict scores and bounding boxes of appropriate size.

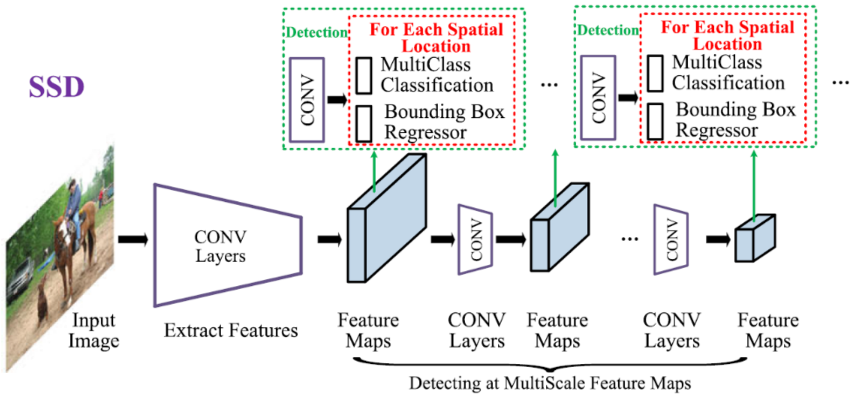


Figure 2.3: SSD training Framework (Liu, 2019)

2.2 Dataset Overview

Thanks to the large availability of images captured by the bill stickers and sent to the company's cloud, it was possible to skip the dataset acquisition phase. While this saved resources and time, the overall quality of the images was not optimal. In fact, until now the images acquired were manually validated by an operator and consequently, the photos didn't need to be of good quality both from the point of

view of resolution, the position of the panel about the lens and the presence of obstacles or reflections on it. When designing an automatic validation system for advertising panels, one would ideally want to have images that clearly show the billboard in optimal lighting conditions and without obstacles, to have a reliable and accurate model. Reality is different, and the data we have available from the real world is never perfect, so we have to come up with solutions to deal with non-idealities.
An example of the pictures available from the dataset is the following:

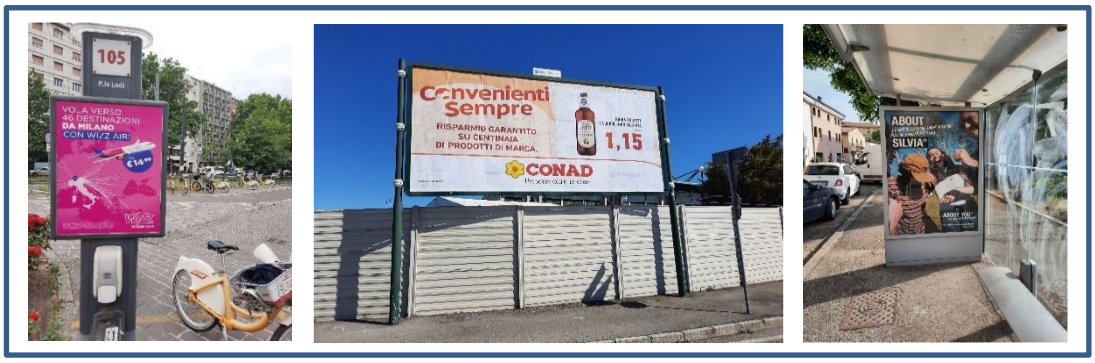

105
VOLA VERSO 46 DESTINAZIONI DA MILANO CON WIZZ AIR!
Convenienti Sempre
RISPARMIO GARANTITO SU CENTINAIA DI PRODOTTI DI MARCA
1,15
CONAD
ABOUT
SILVIA
ABOUT YOU

Figure 2.3: Example of pictures of the dataset

but we can also find images like these:

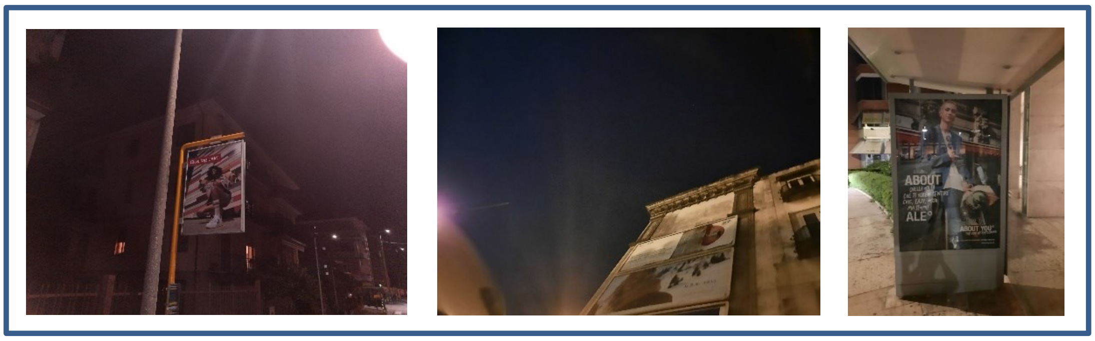


Figure 2.4: Example of low-quality pictures of the dataset

characterized by bad lighting conditions, reflections, and spatial orientations concerning the lens.

As we will see in the following chapters, we do not have much scope to counteract the presence of these artefacts. Considering the quality of picture acquisition, the dataset is very heterogeneous. However, while we have images of poor quality (noise, occlusions, artefacts), it is also true that most of them are of acceptable quality for our model. We will see how the model loses overall accuracy by performing some image pre-processing tests. This is probably because going to handle a particular artefact will result in a loss of information on better quality images, which is crucial for discrimination from outliers.

When acquired the pictures are stored in the cloud and organized according to the following attributes:

- *COD_STATUS*: it can be *"OK"* or *"KO"* and refers to the label hand-assigned to the image by the operator
    - *"OK"* if the declared advertising campaign corresponds to the real one
    - *"KO"* if the declared advertising campaign does not correspond to the real one.
- *TIPOLOGIA*: the panel format type
    - 39 possible panel format types, like *poster, pensiline mono, indicatori stradali, fermate bus mono etc.;*
- *COD_TIPOLOGIA*: the integer ID of the panel format type
- *FORMATO*: the size of the panel format type
    - it goes from a minimum size of *50x70* to a maximum of *600x900*
- *COD_FORMATO:* the integer ID of the size of the panel format
- *SOCI*: the integer ID of the advertising panel operator. Three different advertising operators are considered
- *CIAF*: the integer ID of the images collection cycle
    - every two weeks a new image collection cycle begins, including approximately 10.000 images each
- *NOTP_CD_NOTP*: the integer ID of the image
- *PIAT_DE_PIAT*: the integer ID of the advertising campaign
- *PIIT_DE_PIIT*: the name of the advertising campaign
- *URL_PATH*: the URL of the image to the cloud storage

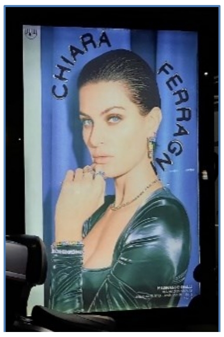


Figure 2.5: Example of one-instance advertising brand Chiara Ferragni (cropped image)

For the Billboard Detection part, we will refer to two collection cycles for the training

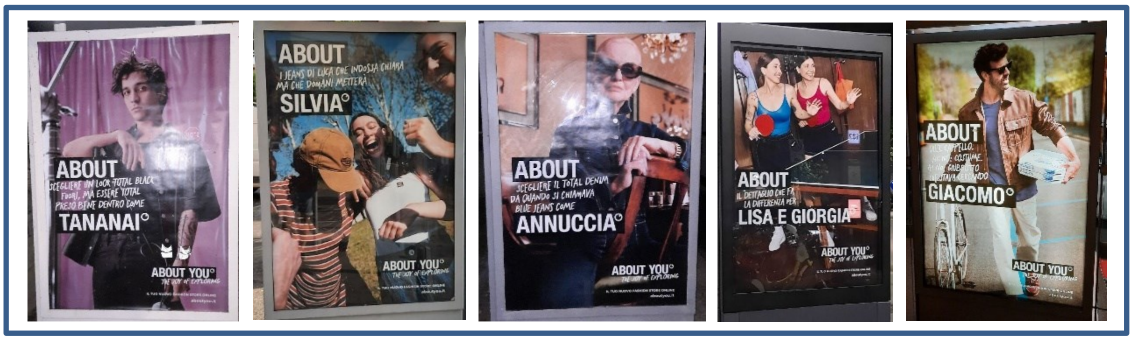


Figure 2.6: Example of multiple-instance advertising brand About You (cropped images)

phase of the model for a total amount of 20.000 images, and then a new cycle will be considered for the test phase. To make it clearer, from one cycle to the next, new advertising brands may be included or a brand present in one cycle is no longer present in the next cycle or the same brand can release a new advertising campaign.
Another aspect to be aware of is that there may be advertising campaigns characterized by a single instance or multiple ones in the same collection cycle. Just to make an example, the *Chiara Ferragni* brand, shown in *Figure 2.5*, proposed billboards that were all the same while the brand *About You*, in *Figure 2.6*, offered different types of them.

2.3 Tools and Frameworks

The Oracle Cloud Infrastructure (OCI) has been used for this task. It is provided by Oracle of which Technology Reply is an official partner. It is a platform of cloud services that enables building and running a wide range of applications. We used three types of services for our goal: OCI Object Storage, OCI Data Labeling, and OCI Vision.

2.3.1 OCI Object Storage

The Oracle Cloud Infrastructure Object Storage service is an internet-scale, high-performance storage platform that offers reliable and cost-efficient data durability. The Object Storage service can store an unlimited amount of unstructured data of any content type, including images and videos (Oracle, s.d.). To store and manage the data we need to know the following resources: Buckets, Objects, Namespace, and Compartment.
Buckets are logical containers for storing data. A bucket is associated with a single compartment that has specific policies that determine what a user can or cannot do

on all the objects in the bucket. *Objects* are any type of data stored in the bucket. *Namespace* is a top-level container for all buckets and objects. When an account is created, each tenant (isolated partition of OCI to administer the user cloud resources) is assigned one unique and immutable namespace name. A *Compartment* is the primary building block used to organize the user cloud resources. A user controls access by creating policies that specify what actions groups of users can take on the resources in those compartments. An Object Storage bucket can only exist in one compartment (Oracle, s.d.).
Object storage has been used to store all the images of our dataset for training the model and for testing it.

2.3.2 OCI Vision

Oracle Cloud Infrastructure Vision is an AI service that allows us to upload images and detect and classify objects on them. Vision supports both pre-trained models and custom models. The available pre-trained models for image analysis are Object detection, Image classification, and Optical Character Recognition (OCR). It is possible to further train the pre-trained model to obtain custom models that fit specific tasks. This service has been used to perform Object detection on the pictures to detect the billboards.

2.3.3 OCI Data Labeling

Data labeling is the process of identifying raw data like images, text files, and videos and adding one or more labels to provide context so that a machine learning or deep learning model can learn from it. Data Labeling lets you create and browse datasets, view data records, and apply labels. In our case, Data Labeling was used to label images of our training set with bounding boxes that highlighted the billboards on the pictures to allow the pre-trained model of OCI Vision to learn from it.

2.4 Billboard detection

As a first step in developing our automatic brand recognition system, object detection aimed at locating the advertising panels in the photos became necessary to be able to exclude the entire surrounding city context, which would otherwise represent useless information for the image similarity step we shall see later. A couple of training tests were conducted to train the object detection model and, finally, a test was carried out to prove the accuracy of the model on new image data. Let us focus first on the training phase.

2.4.1 Training phase

We chose two collection cycles to train our model with a total amount of 20.000 images available. Using a such large dataset would mean having to label the pictures one after the other, which would be very time-consuming. Therefore, extracting 500 images from this dataset was initially decided to compose our first training set. However, the images were not randomly extracted from the dataset. Since we know that there are several types of panel formats, the idea was to keep the same proportion of the number of panels per format type also in the training set. In this way, we optimised the training of the model by favouring the detection of the most common panel format types. Thus, by selecting the images by attribute "*TIPOLOGIA*", we obtained a final training set capacity of 493 images.

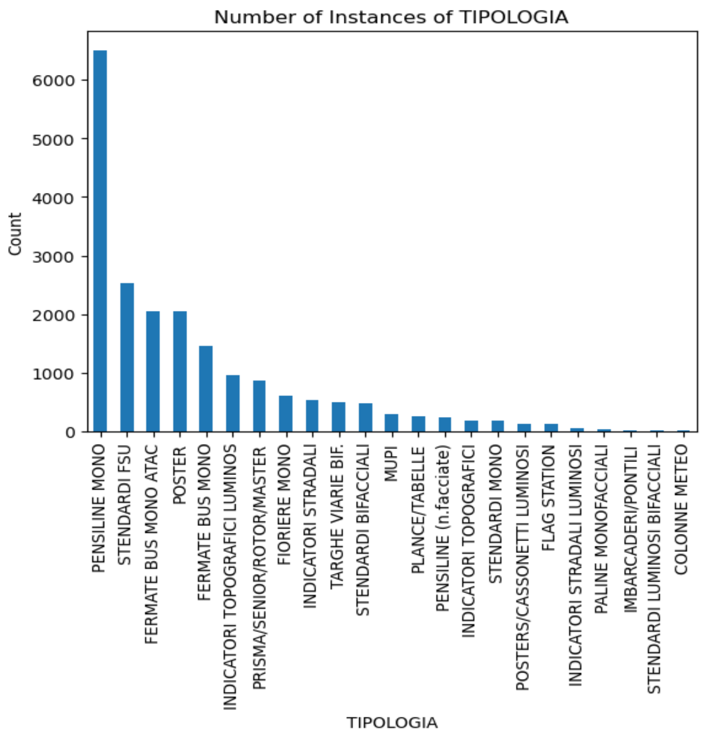


Figure 2.7: Number of instances per format type in the dataset of 20.000 images

A second model was then trained on a larger dataset obtained from the previous training set by adding another amount of 500 images. Again, the additional pictures were selected according to the proportion of format types in the dataset, resulting in a final training set capacity of 984 images.
Before analysing the results obtained, it should be noted that Oracle does not allow us to know the architecture, parameters, and technical operation of the models

provided by OCI Vision service. As we do not have access to its documentation, the model is a 'closed box'. Nevertheless, given the excellent performance obtained, it was decided to rely on it.
Since the object detection module represents the first step in the entire pipeline and the rest of the other processing is made on top of its output, it is crucial to note that for our purpose we will give more relevance to recall rather than precision because it is more important that the model detects all the panels in the photo rather than having false negatives that once discarded cannot be recovered. On the other hand, if all items are classified as false positives, it is possible to reject them in subsequent processing steps that we'll see in later chapters.

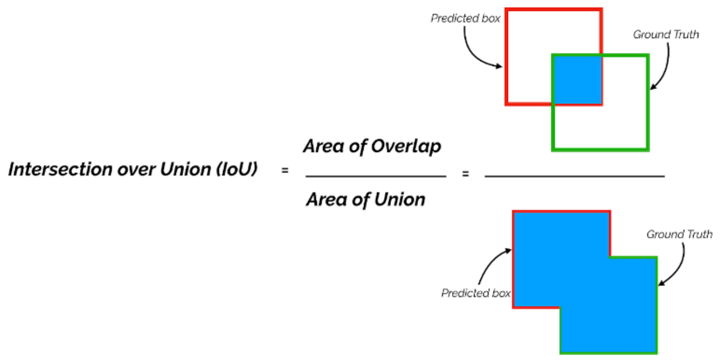


Figure 2.8: IoU is the ratio between the area of intersection and the total area of union of the two bounding boxes

For each image in the dataset, the location of the panels has been manually annotated, with a single class of labels ("billboard"), to provide their ground truth. Then, to evaluate each test done the main important metrics are calculated. In particular, the metrics used are FP (false positive), FN (false negative), TP (true positive), Precision, Recall, F1-score, IoU (intersection over union), and mAP (mean average precision).
The metrics are defined as follows:

- *FP*: is the total number of false detections in the image, that is, objects that are not advertising panels but have been highlighted as such
- *FN:* is the total number of advertising panels that are not detected by the model
- *TP*: is the total number of true detections in the image
- *Precision*: measures how accurate the predictions are, indicating the fraction of true positives out of the total number of detections made by the model

$$Precision = \frac{TP}{TP + FP}$$

- *Recall*: measures how sensible the model is to find all the positives, indicating the fraction of true positives out of the total number of positives

$$Recall = \frac{TP}{TP + FN}$$

- *F1-score:* it is the harmonic mean of the precision and recall.

$$F1 = 2 \times \frac{precision \times recall}{precsion + recall}$$

- *IoU:* measures the overlap of the predicted bounding box to the ground truth box
- *mAP*: metric used to evaluate the object detection model whose calculation is based on the metrics above

Each image composing our dataset displays at least one advertising panel, however, some might show two or three billboards of different brands. Since we do not know a *priori* which brand each ad panel in the image belongs to, we have to check them one by one to find the brand of interest for similarity detection. For such reason comes the importance of detecting all the billboards shown in the images. This fact also explains why we decided not to apply a confidence threshold on detections output by the model.

The metrics just described above are automatically computed by OCI Vision at the end of the training of the model. In particular, the mAP is computed considering an IoU of 0.5 and a confidence threshold of 0.5. These settings are defined by default and cannot be changed. Moreover, the Oracle cloud platform provides us with statistics only in the training phase, meaning that we had to derive them manually when the final model, once trained, was tested for inference on unseen images.

The first training test was performed with a smaller dataset to have an idea of how the OCI Vision model would have worked.
The table below shows the results:

Training test 1	
Total images	493
Test images	49
Training duration	30 min
mAP@0.5	0.8662
Precision	1
Recall	0.7937
Confidence Thr.	0.5

Table 1: Result of Training Test 1

The model had a training time of 30 minutes, which although not long, we can see that it has already achieved quite a high performance. The precision is maximum while the recall is around 0.8, which means that all the model's detections proved to be accurate but there were cases where not all the panels on an image were detected. This could be because the training set used, being relatively small, was not comprehensive enough to include all those special cases where the presence of occlusions, noise or poor illumination in the image is a source of interference in correctly visualising the object.

The second training test differs only in a larger dataset. A new model was trained, and the results are shown below:

Training test 2	
Total images	984
Test images	98
Training duration	30 min
mAP@0.5	0.9896
Precision	0.9903
Recall	0.9533
Confidence Thr.	0.5

Table 2: Results of Training Test 2

Mean average precision and recall increased while precision slightly decreased.

Such high performance could suggest a possible overfitting of the model, however, given the very nature of the dataset with images whose content is quite similar to each other (usually the advertising panel is placed in the centre of the image, furthermore each item to be detected is quite similar to the others, with a rather simple shape, and varying only in size) we expected similar behaviour. However, it is worth considering the possibility that the model is also based on the content of the billboards as well as their shape. This could be a problem because when we are going to apply the model to new datasets, this could favour campaigns that have already been seen in the training or it could not work if these advertising campaigns are not present.

Therefore, testing the final trained model on a new dataset of images belonging to another collection cycle was necessary. The idea is that by feeding the object detection model with photos with new ad campaigns we would have had low performance if it had fixated on the content of the panels during training.

2.4.2 Test phase

As already said, a new image collection cycle was selected for this phase, with a total amount of 13.000 images available. In this case, a dataset of 989 images was obtained from a base of 1.000 images by extracting them proportionally to the number of instances per format type and then labelling the images.
All the statistics needed to evaluate the model were manually derived, in particular precision and recall. To do this, it was necessary to find a criterion for classifying the predictions according to the three categories we have already seen, i.e. TP, FP, and FN. The IoU measures how much our predicted boundary overlaps with the ground truth. Higher IoU indicates the predicted bounding box coordinates closely resemble the ground truth. By setting a threshold for the IoU it is possible to label the predictions as follows:

- *TP*: predicted bounding box overlaps ground truth with $IoU \geq Threshold$
- *FP:* predicted bounding box overlaps ground truth with $IoU < Threshold$ or the predicted bounding box does not match any ground truth
- *FN*: ground truth which does not match with any predicted bounding box

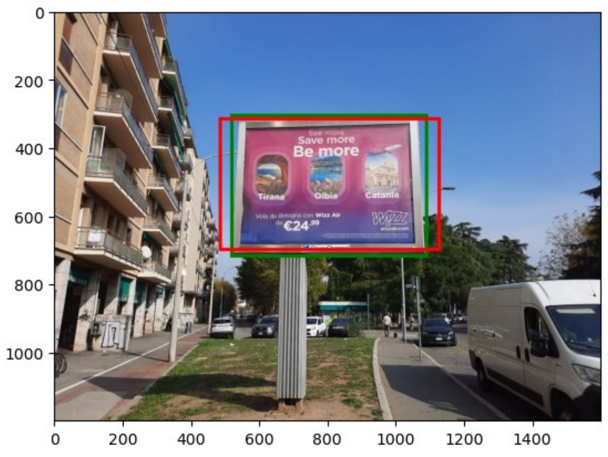


Figure 2.9: Example of ground truth (green) and prediction (red) with IoU = 0.83

For label assignment, the need arose to pair each prediction with its corresponding ground truth. The idea is that if a ground truth is not paired with any prediction instance, then it is a FN, conversely, if a prediction is not found to be paired with any ground truth, then it is a FP. On the other hand, if it turns out that the ground truth and the prediction are paired, then we look at their IoU to decide whether it is a TP or a FP. Just to give an example, *Figure 2.10* shows a possible situation where bounding boxes need to be paired before their IoU can be measured. In this image, three ad panels are highlighted by their corresponding ground truth (green). Only two of these were predicted by the model, *panel 2* and *panel 3*, where the predicted bounding boxes (red) were located. This means that in this image there are indicatively (without looking at their IoU) two TPs, prediction on *panel 2* and prediction on *panel 3*, and one FN, no prediction on *panel 1*. If the ground truth boxes were not paired with the corresponding predictions before the IoU is computed,

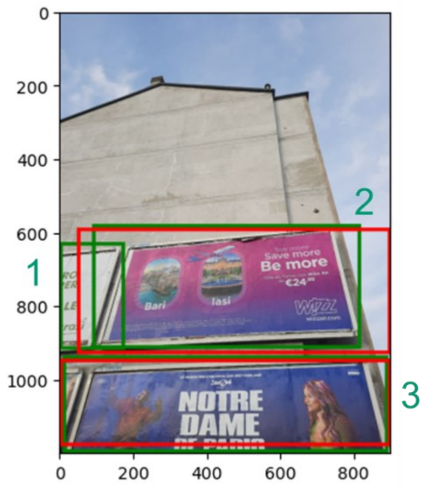


Figure 2.10: Example of picture with three ad panels: two TP and one FN

there could be a misclassification of *panel 1* because its ground truth overlaps slightly with the adjacent prediction bounding box, which actually refers to *panel 2*. In such case the ground truth of *panel 1* would have been classified as FP or TP (depending on their IoU value), when instead it is a FN.
For this reason a method, we will call the *mapping method*, was devised to first allow the correct mapping of ground truths and predictions. The method is based on two steps: first, the centres of the predictions and ground truths are calculated, and then each prediction is mapped to the ground truth with a minimum distance between the centres.
The mapping method is part of a more general algorithm that has been implemented for the classification of predictions.
By iterating over each image, the following algorithm is applied:

Algorithm 1: Classification of Ground Truths and Predictions

```
if the image reports no prediction
   set the ground truth as FN
else if the number of ground truths ≤ number of predictions
   for each ground truth
      mapping between ground truth and prediction
```

```
        for each ground truth-prediction pair
           if IoU < Threshold_IoU
              set prediction as FP
           else set prediction as TP
     if there are unpaired predictions after mapping
        set them as FP
  else (the number of ground truths > number of predictions)
     for each prediction
        mapping between ground truth and prediction
        for each ground truth-prediction pair
           if IoU < Threshold_IoU
              set prediction as FP
           else set them as TP
     if there are unpaired ground truths after mapping
        set them as FN
```

It is possible to identify three cases based on the number of ground truths concerning the number of predictions. *Figure 2.11* shows an example of each case. The total number of FP, TP, and FN was collected in a data frame and then
counted to compute the total Precision, Recall and F1-score.

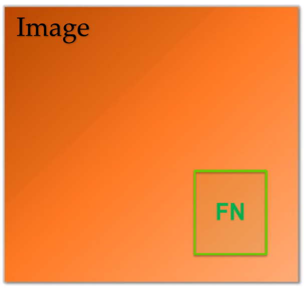


Figure 2.11.a : Case 1, no predictions

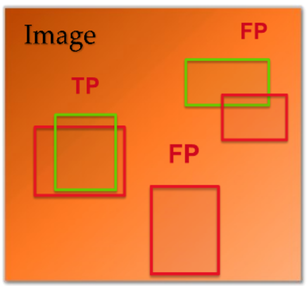


Figure 2.11.b : Case 2, number of Ground Truth instances lower or equal than number of predictions

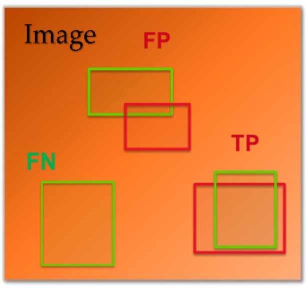


Figure 2.11.c : Case 3, number of Ground Truth instances greater than the number of predictions

In *Figure 2.12* it can be seen that the F1-score remains above 0.8 for an IoU of 0.8 and then drops dramatically as the IoU increases. This is a very positive result because it shows that the model is very good at locating the panels. In fact, it means that most of the predictions obtained have an IoU of around 0.8 with the corresponding ground

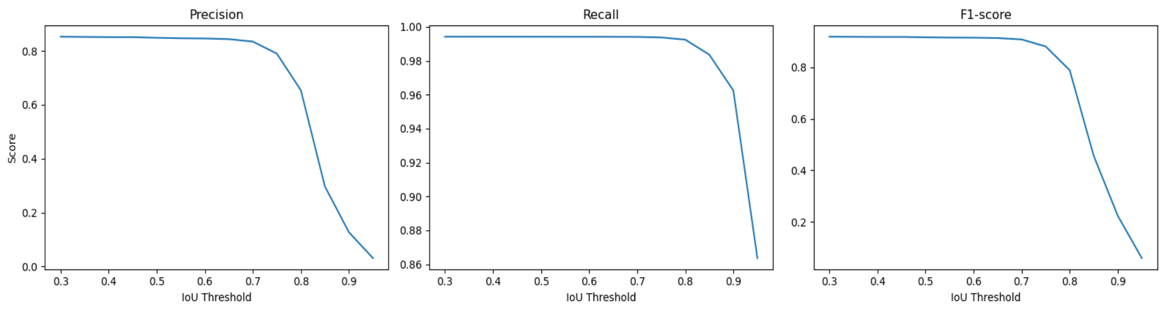


Figure 2.12: Precision, Recall and F1-score measured by varying IoU

truth. Precision remains high for IoU values below 0.8. It then decreases for higher IoU values as the number of FPs increases enormously.

These results show that the model performs very well even on unseen advertising campaigns. This means that during the training phase did not focus on the content of the advertising panels.

Chapter 3

Learned Feature extraction

The keys to understanding whether two images are similar in terms of advertising content are the extraction of features from the images, how these features are compared to each other, and the subsequent statistical analysis of the recorded similarity scores. Different approaches were tested and compared to find the best way to extract features from images. The first exploits the concept of transfer learning by using a pre-trained model for image classification, *Inception v3*, to extract features from images. The model allows the global context of the images to be captured and compared with the possibility to use different similarity distance measures. The second tested approach consisted in extracting text from images, through an OCR system, to compare text fragments based on text similarity, assuming that ad panels for the same brand have most of the keywords in common. The OCR system was implemented using *CRAFT* for text detection and *PARSeq* as a scene text recognition (STR) model. Finally, a more traditional approach was tested, namely *SIFT*. It detects key points in images and obtains image descriptors that can be compared with other images to estimate how similar they are.
An overview of the general concepts inherent in the models just described follows.

3.1 Inception v3

Inception v3 is a convolutional neural network architecture designed for image classification and introduced by Google researchers (Christian Szegedy, 2015). The Inception network is the result of a constant evolution that has led to the creation of different versions of it. InceptionV3 is the third version of the Inception family and is based on an improvement of the previous two versions, especially in terms of accuracy and reduction of computational complexity. Before looking at how InceptionV3 works, it is good to look at what has been done in the previous two versions.
Inception v1, also known as GoogLeNet, was introduced in 2014 to address three main problems faced by deep Neural Networks at the time: the need to capture information on multiple scales, the trade-off between computational efficiency and accuracy, and the vanishing gradient problem.

While it was recognised that increasing the number of hidden layers in the network would increase the performance of the object detection model, it also led to a significant increase in the number of parameters, and therefore a greater need for computational resources and increasingly powerful systems. For that reason, the authors designed an architecture that, more than going deeper with layers in a conventional way, introduces a new module named “inception" that goes wider. The inception module performs convolution on an input, with three different kernel sizes (1x1, 3x3, 5x5). Additionally, Inception layer uses also 1x1 convolution. The outputs are concatenated and sent to the next inception module.

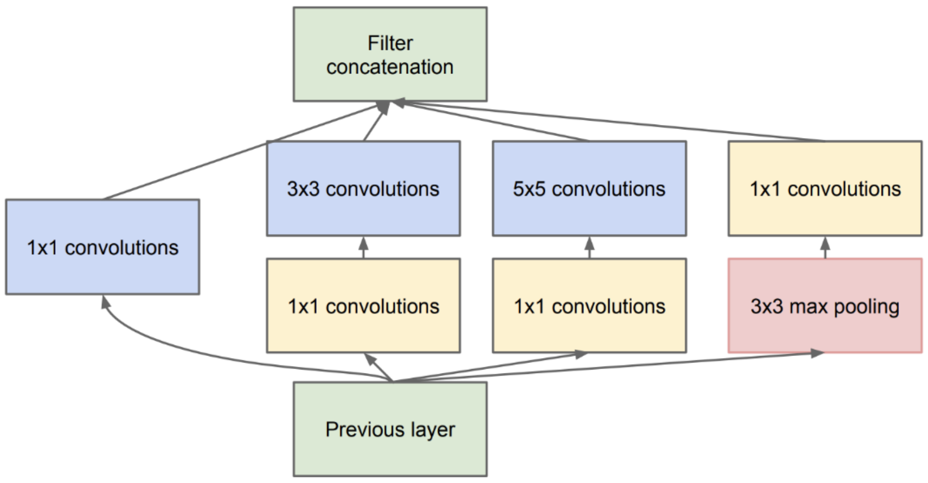


Figure 3.1: Inception module with dimension reductions

GoogLeNet is composed of 9 inception modules with a total amount of 27 layers. To mitigate the vanishing gradient problem the authors introduced two auxiliary classifiers at intermediate stages of the network. They compute the auxiliary loss over the same labels on two inception modules. The total loss function is a weighted sum of the auxiliary loss and real loss (Raj, 2018).
Inception v2 and Inception v3 were presented in the same paper. The aim was to improve accuracy and computational efficiency. Inception v2 introduced the factorization of filters into smaller convolutions to improve computational speed and the Batch Normalization which helps to normalize the activation functions of the hidden layers. This technique accelerates training convergence, improves gradient flow, and enhances the overall performance of the network. Inception v3 incorporates all of the upgrades stated for Inception v2. Moreover, it includes the following :

- RMSProp Optimizer.
- Factorization of 7x7 convolutions.

- Label Smoothing which is a component that acts like a regularizer and is added to the loss formula to prevent the network from becoming too confident about a class and to avoid overfitting. (Raj, 2018).

As mentioned at the beginning of this chapter, it was possible to harness the power of pre-trained Inception v3 through Transfer Learning. The idea behind transfer learning is to leverage the knowledge learned from a pre-trained model to solve a new, but related problem.
Specifically, in deep learning, transfer learning consists of reusing a neural network previously trained on a problem similar to the one being solved. This technique is very convenient because it saves time and resources. Inception v3 was trained on the ImageNet dataset, which contains over 14 million images, and through the Keras library, it is possible to download its weights to reuse the network. Once you have access to the pre-trained model, you can use it in a variety of ways. In our case, it was used as an image feature extraction module by removing the last fully connected layer and reducing the image to a vector of dimension 2048.

3.2 OCR: optical character recognition module

The second approach relies on text similarity between images. The idea is that panels depicting the same campaign should share the same text (if it is a single-instance campaign) or most of it (if it is a multi-instance campaign). Text extraction is performed by an OCR system consisting of two modules: a text detection module and a text recognition module.
CRAFT (Character-Region Awareness For Text Detection) was used for scene text detection. It was presented by (Youngmin Baek, 2019) as an innovative model that guarantees high flexibility in detecting complicated scene text images, such as arbitrarily-oriented, curved, or deformed texts. CRAFT is based on the idea of character-level localization. Instead of detecting text as a whole, CRAFT detects each character individually and then groups them into words or text lines. This way, CRAFT can handle complex text layouts and fine-grained details.
The model is based on a fully convolutional neural network (FCN) that takes an image as input and produces two output maps: a character region map and an affinity map. The character region map indicates the probability of each pixel belonging to a character, while the affinity map indicates the probability of each pixel belonging to the same text instance as its neighbouring pixels. By combining these two maps, CRAFT can generate text bounding boxes that are aligned with the character regions and respect the affinity between characters.

The FCN model of CRAFT is built upon the VGG16 backbone, which is pre-trained on ImageNet for image classification. The model has four stages: feature extraction, feature fusion, region score prediction, and affinity score prediction. The feature extraction stage uses the first 10 convolutional layers of VGG16 to extract low-level features from the input image. The feature fusion stage uses four additional

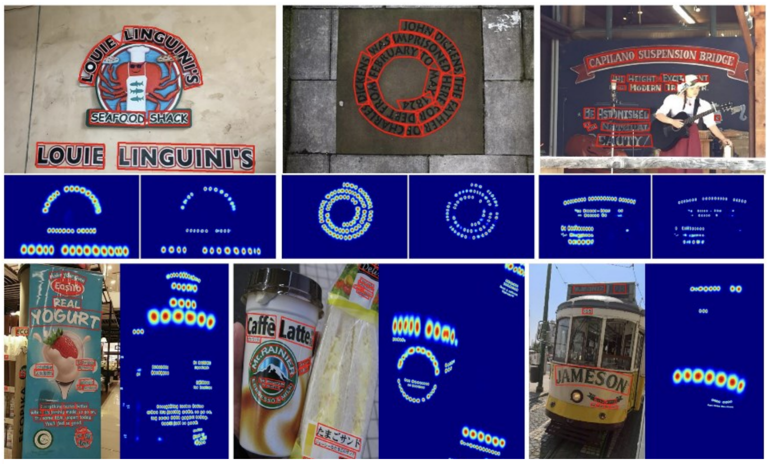


Figure 3.2: Experimental results on the TotalText dataset. First row: each column shows the input image (top) with its respective region score map (bottom left) and affinity map (bottom right). Second row: each column only shows the input image (left) and its region score map

convolutional layers to fuse the low-level features with high-level features from the last convolutional layer of VGG16. The region score prediction stage uses a 1x1 convolutional layer to predict the character region map from the fused features. The affinity score prediction stage uses another 1x1 convolutional layer to predict the affinity map from the fused features.
One of the strengths points of CRAFT is that it can detect text of various shapes and orientations without relying on predefined anchors or sliding windows. Moreover, it can handle text with different fonts, sizes, colors, and backgrounds. It can deal with occluded or overlapping text by using affinity scores to link characters. Finally, it can be easily extended to other languages or scripts by using a language-agnostic character region detector. These characteristics make it perfect for our use case. Advertising campaigns often have text that varies in colour, size, orientation and sometimes even language.
CRAFT is a state-of-the-art method for text detection in natural scenes. It has achieved impressive results on several benchmark datasets, such as ICDAR 2015, ICDAR 2017 MLT, and CTW1500. It is also fast and efficient, running at about 8.6 frames per second on a single GPU (Youngmin Baek, 2019).

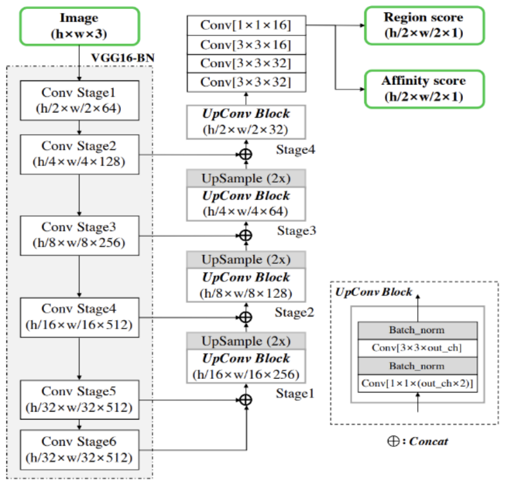


Image (h×w×3)
VGG16-BN
Conv Stage1 (h/2×w/2×64)
Conv Stage2 (h/4×w/4×128)
Conv Stage3 (h/8×w/8×256)
Conv Stage4 (h/16×w/16×512)
Conv Stage5 (h/32×w/32×512)
Conv Stage6 (h/32×w/32×512)
Conv[1×1×16]
Conv[3×3×16]
Conv[3×3×32]
Conv[3×3×32]
UpConv Block (h/2×w/2×32)
Stage4
UpSample (2x)
UpConv Block (h/4×w/4×64)
Stage3
UpSample (2x)
UpConv Block (h/8×w/8×128)
Stage2
UpSample (2x)
UpConv Block (h/16×w/16×256)
Stage1
Region score (h/2×w/2×1)
Affinity score (h/2×w/2×1)
UpConv Block
Batch_norm
Conv[3×3×out_ch]
Batch_norm
Conv[1×1×(out_ch×2)]
⊕ : Concat

Figure 3.3: Schematic illustration of CRAFT network architecture

Once an image is passed as input to CRAFT it will produce two main outputs: the character region map and the text bounding box map. By cropping the image at the predicted bounding boxes, it is possible to pass the cropping as input to the STR model. Scene Text Recognition is a challenging task that involves recognizing text in natural images. Unlike document text recognition, STR has to deal with various factors such as low resolution, blur, distortion, occlusion, and complex backgrounds. To overcome these difficulties, STR models use language context to be more robust against noisy or corrupted images.

PARSeq (Scene Text Recognition with Permuted Autoregressive Sequence Models) is a novel method for scene text recognition (STR) recently presented by (Darwin Bautista, 2022) which achieves state-of-the-art results in STR benchmarks.

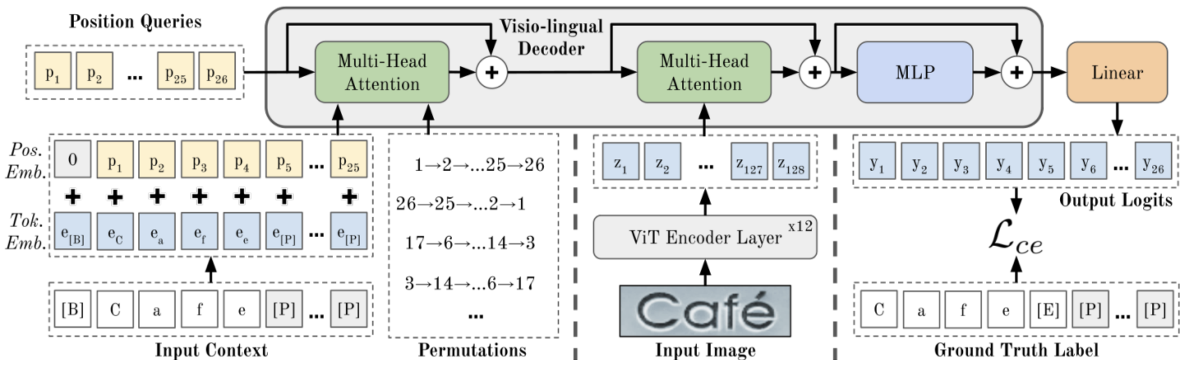


Figure 3.4: PARSeq architecture

On a conceptual level, the model can be separated into two main components: the encoder (*ViT Encoder* Layers) and the Decoder (*Visio-lingual Decoder*).

The Vision Transformer (ViT) adapts the standard transformer approach for text to visual data. The input image is broken down into sub-images and flattened to obtain a vector (z_i). This vector is then combined with the positional embedding and fed into the Visio-lingual decoder, more specifically to the Multi-Head Attention (MHA) layer (Darwin Bautista, 2022) (Ashish Vaswani, 2017).
The main novelty behind PARSeq is based on the new concept of Permutation Language Modeling (PLM), which was first introduced by XLNet for natural language understanding, and it consists of feeding, during the training, different permutations (orders) of words, rather than a fixed order. PLM randomly permutes the order of tokens in a sequence and trains the model to predict each token based on its visible neighbours according to the attention mask. By using different attention masks for different permutations, PARSeq learns an ensemble of autoregressive (AR) models with shared weights that can capture bidirectional context.
As shown in *Figure 3.4*, the Visual Encoder consists of two MHA modules. The first MHA module is used for *context–position* attention. The Positional embeddings/Position queries represent the encoding of the character position in the input text, while the Token Embedding is the encoding of the character in the input text. First, the token embeddings and the positional embeddings are added to form the *context embedding*. Then, this context embedding, which serves as the *key* and *value* vectors, is fed to the MHA module along with the separate positional embedding which serves as a *query* to the MHA. In addition, a list of permutations is also passed to the MHA. Next, the results of this module are combined with the output of the encoder using a second MHA which is used for *image-position* attention. The last decoder hidden state is an MLP whose result is sent to a linear layer where the final output is given. Due to its extensive use of attention, it is robust on arbitrarily-oriented text which is common in real-world images. This aspect proved to be fundamental to the correct recognition of text on images representing advertising campaigns. For this reason, it was decided to use this model, that is also very accurate even on images with noise or blur.

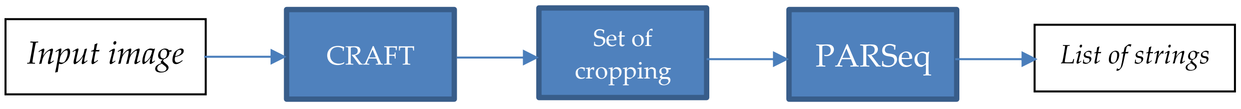


Figure 3.5: The pipeline of the OCR module

3.3 SIFT: Scale-Invariant Feature Transform

In computer vision, deep learning solutions are often more accurate than traditional algorithms. However, they do not always guarantee good results because it depends on the type of dataset available and the task at hand. For this reason, we wanted to test at least one classic image-matching algorithm, namely SIFT.
The *SIFT (Scale-Invariant Feature Transform)* algorithm was presented in 2004 by David G. Lowe (Lowe, 2004) and represents a milestone in computer vision. It is still one of the most reliable feature extractor and descriptor, even if not the fastest. It works by identifying keypoints based on their local intensity extrema and computing descriptors that capture the local image information around those keypoints. Once these descriptors are computed they can then be used for multiple tasks. The most important aspect of this algorithm is that the image content is mapped into local feature coordinates that are invariant to translation, rotation, scale, and other imaging transforms.
The algorithm can be divided into four steps: Scale-space extrema detection, Keypoint localization, Orientation assignment, and keypoint descriptor. The scale-space extrema detection step ensures that the features are scale-independent. To achieve this goal, a scale space, defined as a collection of images with different scales, is generated from a single image. The scale space is divided into octaves with half of

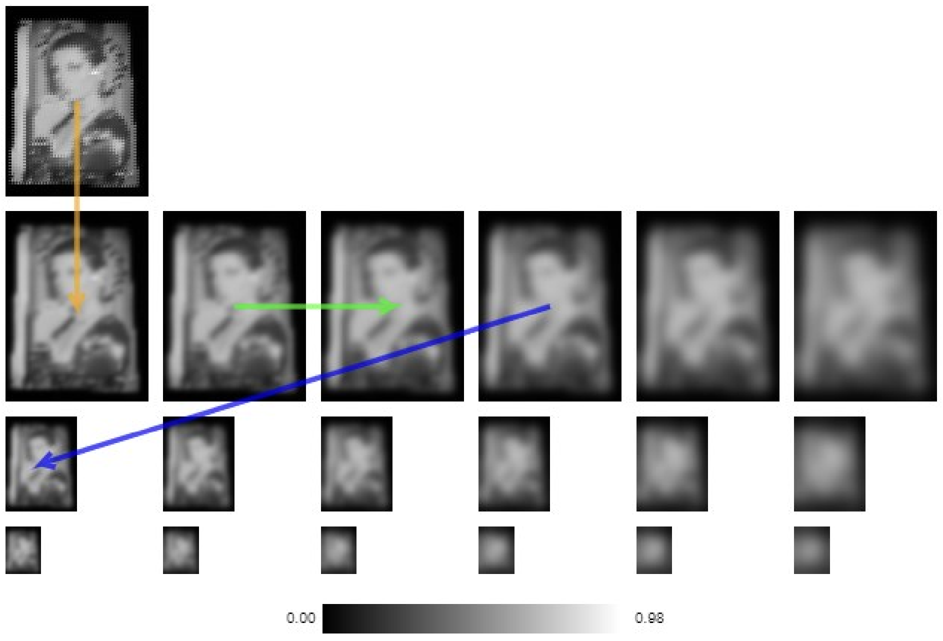


Figure 3.6: Example of scale space: the algorithm first doubles the width and height of its input using bilinear interpolation (the first picture above), the picture is subsequently blurred using a Gaussian filter with different standard deviations (from left to right), finally the antepenultimate picture is down sampled (see the blue arrow) starting another row of Gaussian filtering.

the resolution of the previous octave and a Gaussian blur is applied to each octave by doubling its standard deviation σ. Within each octave, the picture is gradually blurred by generating a Gaussian scale space with *S* intervals. The more *S* the more feature points, but on the other side the matching accuracy decreases.
By blurring the images, we eliminate noise and make the most important features stand out. To make them even more evident, we use a technique called Difference of Gaussian (DoG). DoG is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original (Singh, 2019).
For each octave, DoG creates another set of images by subtracting every image from the previous image in the same scale.

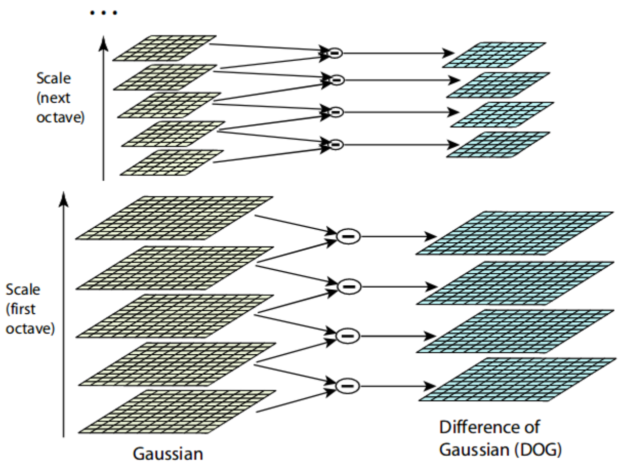


Figure 3.7: Adjacent Gaussian images, within the same octave, are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated.

The next step consists of keypoint localization. The idea is to find local maxima and minima of the DoG. To locate local maxima and minima we compare each point with the eight neighbours on its scale and nine on the previous and next scale levels, for a total of 26 points to be compared.
Once we have selected the possible points of interest, we need to carry out the final check to ensure that we are selecting the most accurate keypoints. Indeed, keypoints localization in the maxima points is not precise due to the quantization on the pixel grid. For such reason, the Second order Taylor series approximation of the DoG is applied in the neighbourhood of the point to accurately locate the maxima. In particular, the zero-crossing of the derivative is considered to keep only the more stable keypoints (discard points with gradient < 0.03). Another aspect to consider is that DoG has a strong edge response, but typically edges are not good features, so a

Hessian matrix is used to identify such keypoints.

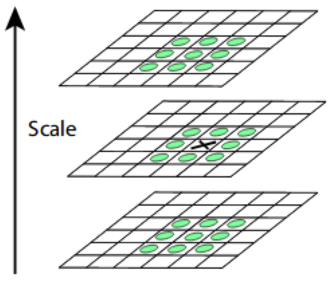


Figure 3.8: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3 × 3 regions at the current and adjacent scales (marked with circles).

Now that we have refined the selection of keypoints, all that remains is to assign an orientation value to make them invariant to rotation. First, for each keypoint we compute the gradient and then we can find the magnitude M and the orientation θ which represent respectively the intensity of the pixel and its direction. At this point, all the subsequent operations will be made on an image that is rotated with respect to the found direction and scaled according to the keypoint scale. Then, a histogram of the gradient directions at the chosen scale in a neighbourhood of the keypoint is built. The histogram is divided into 36 bins of 10° each. Each sample is weighted according to its gradient module M and the Gaussian smoothing term σ with $\sigma = \sigma_{scale}$, to give more weight to the samples closer to the keypoint location. The final orientation assigned to the keypoint is given by the direction of the peak value of the histogram. Additionally, if there are more similar peaks, then multiple keypoints with different orientations are generated.
The final step for SIFT consists in generating a unique vector for every keypoint, by considering the orientation and the magnitude of the neighbouring pixels. We start by considering a 16x16 window around the keypoint. This window is further divided

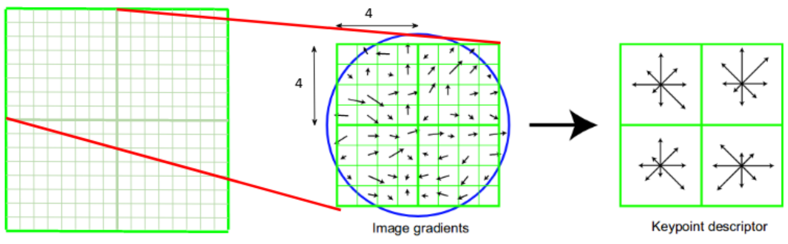


Figure 3.9: keypoint descriptor generation (Vidhya, 2023).

into 4x4 sub-windows, and for each of them, we generate the histogram using magnitude and orientation, with 8 values corresponding to 8 intervals of 45°. For each interval, the sum of the gradient modules of the samples and the direction falling in that interval is computed. The final descriptor is a vector of $\boldsymbol{d}$ dimension where $\boldsymbol{d}$ = (16/4)x(16/4)x8 = 128 elements.
In our use case, SIFT was used to extract feature descriptor vectors that were then used to compare images and calculate a similarity score to detect outliers, if any. A more detailed description of its use follows in the next chapter.

Chapter 4

Second task: outlier detection

This chapter describes the pipeline defined to detect outliers within a dataset of images depicting the same advertising campaign. We will first assume the Inception v3 model, described in the previous chapter, as the feature extraction module. Then we will introduce the necessary modifications to be able to apply the OCR module and the SIFT algorithm independently, although most of the necessary processing steps will remain unchanged. Finally, we will present the final version of the algorithm workflow given by the combination of two of the approaches seen for extracting features from images.

4.1 Methods

4.1.1 Image preprocessing

The pipeline is divided into three main blocks: the image preprocessing block, the feature extraction and similarity estimation block, and the outlier detection block. The first includes what we have seen in the early chapters of this thesis, the second block includes all the operations needed to extract features and compute the similarity score between images. The latter aims at identifying the outliers in the dataset.
As described in the first task, the first step is to detect the panels using the object detection model. Before that, each image is given a name, which is made up of some of the attributes in the dataset presented in Chapter 2. From the name we want to extract the most important information, i.e. the ID of the image, which campaign it belongs to, which cycle it comes from and the type of format.
Let us take the following name as an example: *"10143285_97535_202211_76_1.jpeg";* the first field is the *NOTP_CD_NOTP* which identifies the image, the second is the *PIAT* which indicates that the campaign in question is "About you", the next is the cycle and finally we have the panel type and format codes.
After applying the model, we obtain the coordinates of the identified bounding boxes for each image, if they are successfully detected. While such instances are quite infrequent, there may be rare cases where the model fails to detect any bounding

box. In such scenarios, we have decided to treat these images as outliers. Several factors can contribute to this, including excessively blurred images, billboards that are positioned too far from the lens, or the presence of obstructing objects. Based on the obtained coordinates, we will proceed to crop the image, generating a number of crops equal to the total panels detected within that specific image. It is important to note that from now on, we will reason at the level of the clippings, treating them as independent images. To always be able to trace back to the image to which they belong, each cropping has been given a name made up of the union of the image name and the string "*cropping_n*", where *n* is a sequential integer identifying the cropping.
For example, consider the following image, where the name is reported, which shows two advertising panels with their corresponding bounding boxes:

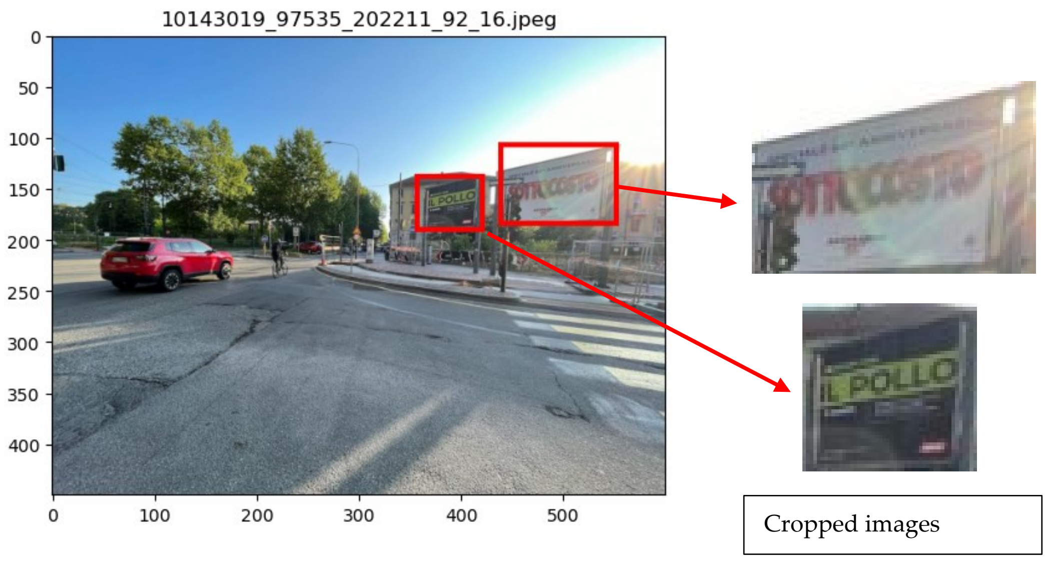


Figure 4.1: Example of an image with two ad panels and its corresponding cropped images

In this case, it will be given two crops named:

- *10143019_97535_202211_92_16cropping0.jpeg*
- *10143019_97535_202211_92_16cropping1.jpeg*

4.1.2 Feature Extraction and similarity estimation

The next step involves feature extractions from the crops just obtained. This is the most important part of the entire data flow. In fact, the ability to discriminate between outliers and non-outliers mainly depends on the quality of features extracted. At this stage, every crop is first resized to a standard input shape of (224 x 224 x 3) and then sent to the pre-trained Inception v3 model, implemented through the *Keras* library. The output consists of image embeddings, which represent the

images in a vector space. This provides a valuable approach to effectively compress the high-dimensional pixel space of the images (224 x 224 x 3) into a significantly lower-dimensional representation (2048). Once the images have been reduced to vectors, it is possible to treat them as a set of points in a feature space where the distance between each point represents how similar they are. The distance can be computed in many ways depending on the task at hand.
The most used distance measures are:

- *Cosine similarity*: it measures the cosine of the angle between two vectors. This kind of similarity measure does not depend on the magnitudes of the vectors, but only on their angle.

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

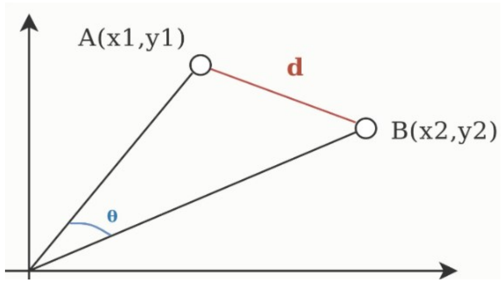


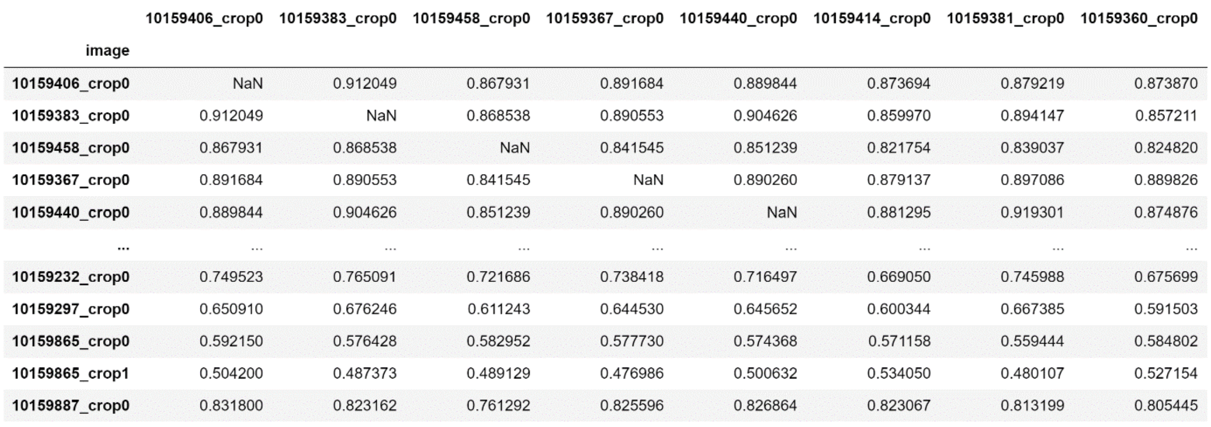
Figure 4.2: Visual representation of cosine distance

- *Euclidean distance:* represents the length of a line segment between the two points. It can be calculated using the Pythagorean theorem (Wikipedia, s.d.). For points given by Cartesian coordinates in n-dimensional Euclidean space, the distance is:

$$d(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_n-q_n)^2}.$$

For our case study, the use of the cosine similarity distance was considered appropriate. There are several reasons for choosing this distance measure, in particular, we know that this measure takes into account the direction of the feature vectors rather than their magnitudes. In our case, two panels representing the same advertising campaign should have similar feature vectors in direction, although the vector magnitudes may vary due to factors such as lighting conditions, panel orientation, panel size or the presence of noise. A similarity measure that only considers the absolute differences between the feature vectors could instead be affected by these variations. Another reason has to do with computational efficiency: the calculation of cosine similarity is computationally very efficient compared to e.g., Euclidean distance. In our case, since we are dealing with thousands of images, this can help to reduce the computation time and make the process more scalable.

However, another classic alternative to using cosine similarity is to apply the Euclidean distance to the normalised vector whose length has been scaled to have a unit norm. This is helpful when the direction of the vector is meaningful, but the magnitude is not.
Having chosen the similarity distance measure, we can proceed to compare the images. Each image is now compared with all the others and the score, which is bounded between 0 and 1 where 1 is maximum similarity, is recorded in an $n \times n$ *comparisons table*, where n is the number of crops compared. For each crop, the average score is then calculated to give a general estimate of the degree of similarity between that crop and all the others.



	10159406_crop0	10159383_crop0	10159458_crop0	10159367_crop0	10159440_crop0	10159414_crop0	10159381_crop0	10159360_crop0
image								
10159406_crop0	NaN	0.912049	0.867931	0.891684	0.889844	0.873694	0.879219	0.873870
10159383_crop0	0.912049	NaN	0.868538	0.890553	0.904626	0.859970	0.894147	0.857211
10159458_crop0	0.867931	0.868538	NaN	0.841545	0.851239	0.821754	0.839037	0.824820
10159367_crop0	0.891684	0.890553	0.841545	NaN	0.890260	0.879137	0.897086	0.889826
10159440_crop0	0.889844	0.904626	0.851239	0.890260	NaN	0.881295	0.919301	0.874876
...	...	...	...	...	...	...	...	...
10159232_crop0	0.749523	0.765091	0.721686	0.738418	0.716497	0.669050	0.745988	0.675699
10159297_crop0	0.650910	0.676246	0.611243	0.644530	0.645652	0.600344	0.667385	0.591503
10159865_crop0	0.592150	0.576428	0.582952	0.577730	0.574368	0.571158	0.559444	0.584802
10159865_crop1	0.504200	0.487373	0.489129	0.476986	0.500632	0.534050	0.480107	0.527154
10159887_crop0	0.831800	0.823162	0.761292	0.825596	0.826864	0.823067	0.813199	0.805445

Figure 4.4: Example of comparison table: the names of the images are given in both columns and rows, and each cell reports the score between the corresponding croppings. The 'NaN' value was assigned to the comparisons between the same image.

4.1.3 Score analysis

So far, our focus has been on the cropping aspect, but what we are truly interested in is the classification of the entire image. Each image may contain multiple panels, but only one panel is of interest to us. This panel should be the one that showcases the same advertising campaign as the dataset to which the image belongs. However, we lack prior information that allows us to determine which of the different panels present in an image is the target panel. Given this situation, was decided to select the panel with the highest average score. This decision comes from the assumption that the crop with a higher average score closely resembles the rest of the dataset and is therefore more likely to belong to the advertising campaign in question. Consequently, the next step involves pruning the comparison table. Specifically, we select the target cropping for each image by choosing the one with the highest score among those associated with that image. This process results in a new table, known as the *pruning table*, which assigns the target cropping and its average score to each

image.
Before proceeding to the final step of outlier detection, it is necessary to pre-process the scores through normalization. The goal of normalization is to transform the features to be on a similar scale and to make them comparable. Many normalization methods exist such as:

1. *Min-max normalization:* it is the simplest method and consists in rescaling the range of features to scale the range in [0, 1]. The formula is the following:
$$x' = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$
It preserves relationship among original data values.
2. *Z-score normalization:* it uses the standard deviation to normalize. It is calculated as:
$$x' = \frac{x_i - \mu}{\sigma}$$
Where x_i is the data value, μ is the mean of the dataset, and σ is the standard deviation.
This kind of normalization is useful when the actual minimum or maximum of attribute x_i is unknown and can help when outliers might influence the min-max normalization.
3. *Decimal scaling normalization:* normalize by moving the decimal point of the values of the attribute. The number of decimal points moved depends on the maximum absolute of x_i.
$$x' = \frac{x_i}{10^j}$$

In our case, it was considered appropriate to use z-score normalisation, which is not affected by the presence of outliers, such as min-max normalization. Furthermore, as we will see later, z-score normalization allows us to give data a well-defined meaning, knowing that if a score is, for example, three variances away from the mean, it is most likely an outlier. At this point, the pruning table will look like the one in *Figure 4.5.*

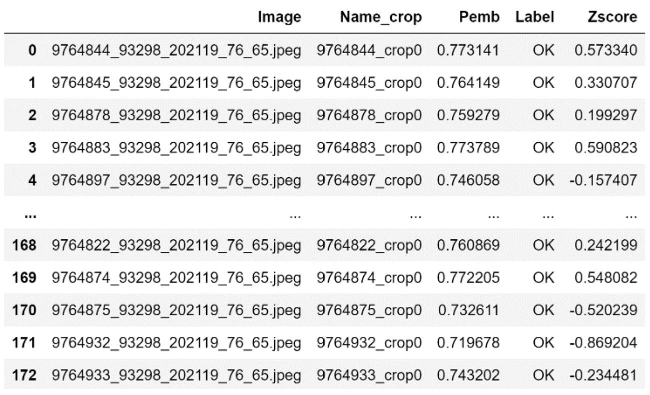


	Image	Name_crop	Pemb	Label	Zscore
0	9764844_93298_202119_76_65.jpeg	9764844_crop0	0.773141	OK	0.573340
1	9764845_93298_202119_76_65.jpeg	9764845_crop0	0.764149	OK	0.330707
2	9764878_93298_202119_76_65.jpeg	9764878_crop0	0.759279	OK	0.199297
3	9764883_93298_202119_76_65.jpeg	9764883_crop0	0.773789	OK	0.590823
4	9764897_93298_202119_76_65.jpeg	9764897_crop0	0.746058	OK	-0.157407
...	...	...	...	...	...
168	9764822_93298_202119_76_65.jpeg	9764822_crop0	0.760869	OK	0.242199
169	9764874_93298_202119_76_65.jpeg	9764874_crop0	0.772205	OK	0.548082
170	9764875_93298_202119_76_65.jpeg	9764875_crop0	0.732611	OK	-0.520239
171	9764932_93298_202119_76_65.jpeg	9764932_crop0	0.719678	OK	-0.869204
172	9764933_93298_202119_76_65.jpeg	9764933_crop0	0.743202	OK	-0.234481

Figure 4.5: Example of the Pruning table with the name of the image (Image), the cropping with the highest score (Name_crop), its score in normalized (Zscore) and not normalized form (Pemb), and the label taken from the dataset that says us if the image is labeled as outlier (KO) or not outlier (OK)

4.1.4 Outlier detection

The final step consists in analysing the normalized scores. To do this, we first calculate the histogram to visualise the distribution. Most of the time the distribution pattern resembles a left-skewed Gaussian curve, like in *Figure 4.6*, however, sometimes it may slightly change due to the type of feature extractor used or the overall quality of the images.

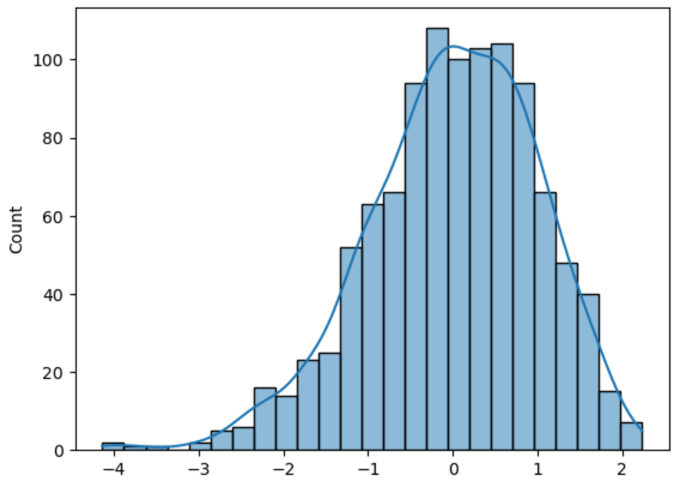


Figure 4.6: Example of the distribution of normalized scores. Note how the distribution spans a range from -4 variances from the mean to a maximum of +2 variances above the mean.

From the distribution, it was then decided to set a threshold to classify whether the images are outliers or not. Based on some tests, -3 was chosen as the threshold. All values below the threshold are then classified as outliers. Note that in this case the threshold was set only for negative values of the variance, in contrast to the usual way of searching for outliers with z-score normalization. The reason for this is clearly related to the fact that all images with a z-score above 0 (i.e. the mean) are very unlikely to be outliers. An important aspect to note is that by mapping scores onto a scale using z-scores and fixing a threshold, we create a dynamic threshold that accounts for the specific characteristics of each distribution. Moreover, this threshold carries a meaningful mathematical interpretation, indicating the degree of deviation from the mean and helping us identify potentially anomalous values.

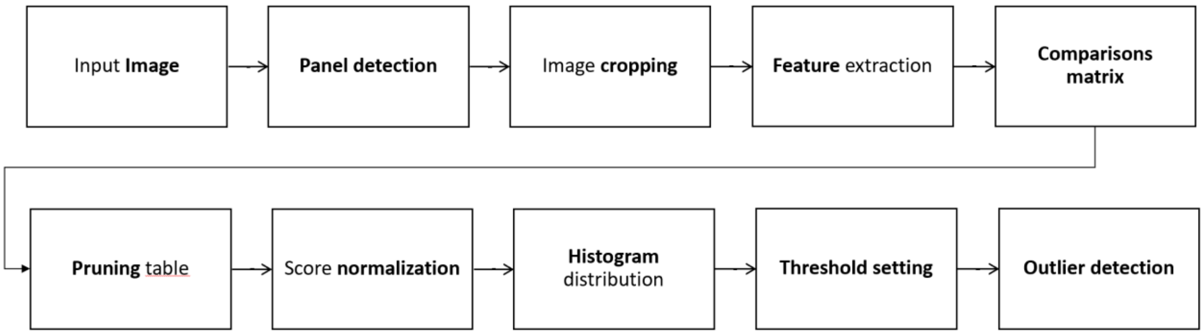


Figure 4.7: General pipeline of the outlier detection algorithm

4.2 OCR adaptation

When applying OCR to identify outliers, certain adjustments need to be made to the data flow. In this scenario, the feature extraction block undergoes modifications to incorporate text extraction. Instead of reducing the image to a vector of numbers, we use the text on the image as a means of comparison with other images. The images are fed as input to the OCR system, which utilizes CRAFT for text detection and subsequently employs PARSeq for text recognition. As a result, the output provides the extracted text for each cropped section. At this stage, text pre-processing becomes crucial, entailing cleaning and transforming unstructured text data to render it suitable for analysis.

The pre-processing part is performed in four steps:

1. *Tokenization* of the text items: in this step the text is split into smaller units called tokens. Here, tokens can be either words, characters, or subwords.
   Ex: “Never give up” → [Never], [give], [up]
2. *Punctuation Removal*: all the punctuation from the text is removed. There is the

string library of Python that contains some predefined list of punctuations such as *'!"#$%&'()*+,-./:;?@[\]^_`{|}~'*

3. *Stop Word Removal:* the stop words are the most common used words that carry less or no meaning. For many languages, the *NLTK* Python library contains a list of words that are considered stop words for a particular language. In our case, we considered the stop words for the Italian language since the detected text was in Italian. An example of Italian stop words are: "a", "abbastanza", "abbia", "abbiamo", "abbiano", "abbiate", "accidenti", "ad", "adesso", "affinché", "agli", "ahimè", "ai", "al", etc.
4. *Conversion of the text to uppercase:* converting text to capital letters.

At this stage, it is possible to encounter empty strings, which can occur either due to the absence of detected text during pre-processing or because the string became empty after undergoing pre-processing, as there were no meaningful words present. The presence of empty strings could disrupt the distribution of scores by affecting the correct detection of outliers. In fact, since they are empty, their comparison with all the others would give zero scores. For this reason, it was decided to separate the crops with empty strings from those with non-empty strings.
At this point, the comparison table is generated considering only the crops with text. In this case, comparisons between text strings are again made using a similarity distance measure. The *Levenshtein distance* was chosen for this purpose, which ranges from 0 to 100. This is also known as an edit distance-based algorithm because it calculates the number of edits required to transform one string into another (Analytics Vidhya, s.d.). The edits counted as one operation are:

- Insertion of a character
- Deletion of a character
- Replacing a character

The higher the number of operations, the lower the similarity between the two

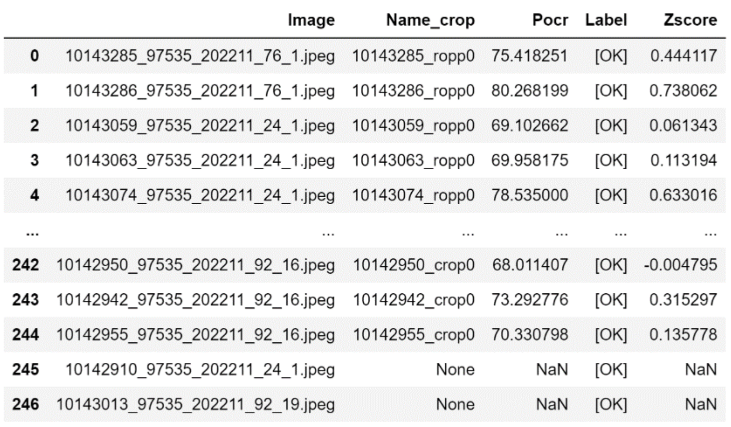
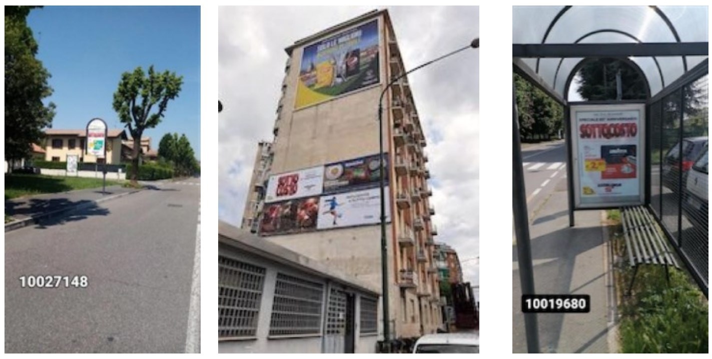


	Image	Name_crop	Pocr	Label	Zscore
0	10143285_97535_202211_76_1.jpeg	10143285_ropp0	75.418251	[OK]	0.444117
1	10143286_97535_202211_76_1.jpeg	10143286_ropp0	80.268199	[OK]	0.738062
2	10143059_97535_202211_24_1.jpeg	10143059_ropp0	69.102662	[OK]	0.061343
3	10143063_97535_202211_24_1.jpeg	10143063_ropp0	69.958175	[OK]	0.113194
4	10143074_97535_202211_24_1.jpeg	10143074_ropp0	78.535000	[OK]	0.633016
...	...	...	...	...	...
242	10142950_97535_202211_92_16.jpeg	10142950_crop0	68.011407	[OK]	-0.004795
243	10142942_97535_202211_92_16.jpeg	10142942_crop0	73.292776	[OK]	0.315297
244	10142955_97535_202211_92_16.jpeg	10142955_crop0	70.330798	[OK]	0.135778
245	10142910_97535_202211_24_1.jpeg	None	NaN	[OK]	NaN
246	10143013_97535_202211_92_19.jpeg	None	NaN	[OK]	NaN

Figure 4.8: Pruning table with approach including OCR. Ad campaign "ESSELUNGA".

strings. Again, once the comparison matrix is defined, we proceed to prune it by generating the *pruning table*, which contains only images with at least one crop with text. For each image, we associate its crop with the highest average similarity score and then normalise these scores using z-score normalisation. Only at this point are the images without text added to the pruning table and the 'NaN' score associated with them.
With this approach, the pruning table will look like the one in *Figure 4.8*, where at the bottom it is possible to see two images where no text was detected. These will be classified as outliers along with images whose normalized average score is below the threshold set at -3.



a) *b)* *c)*

Figure 4.9: Figures a) and b) show the images where no text was detected. In both images the panels appear to be too far away from the lens, making it impossible for OCR to detect the text. Figure c) shows the image reported in line 1 on the table of Figure 4.8, which recorded an average score of 80.26 out of 100 when compared with all other images.

4.3 SIFT adaptation

SIFT operates exclusively on greyscale images, from which it can extract keypoints and descriptors. For this reason, once cropped, the images must pass through a pre-processing stage where they are converted to greyscale before being passed as input to the feature extraction block. The latter, based on SIFT's algorithm, identifies the key points within each crop and generates descriptors for each of these key points. The comparison of images is then performed by comparing their respective descriptors. In this particular case, similarity estimation is based on the number of common keypoints shared between the images. Each keypoint is matched against all the keypoints in the other image using a matching algorithm, which can be either a *Brute-Force Matcher* (BFMatcher) or a *Flann-Based Matcher*. BFMatcher is the simplest option, as it takes the descriptor of one feature in the first set and is matched with all other features in second set using some distance calculation. Classical feature descriptors like SIFT or SURF are usually compared and matched using the Euclidean distance (or L2-morm). On the other hand, FLANN, which stands for Fast Library for Approximate Nearest Neighbours, comprises a collection of algorithms optimized for efficient nearest neighbour searches in large datasets and high-dimensional features. It operates faster than the BFMatcher when dealing with large datasets, making it more suitable for our case, as we have thousands of images to compare. The knnMatch() function is used to perform a k-nearest neighbour search to find the two nearest neighbours. The resulting matches are then filtered using the distance ratio test proposed by Lowe in its paper. The distance ratio between the two closest matches of a considered keypoint is calculated and it is a *good match* if this value is below a threshold. This ratio helps to distinguish between ambiguous matches (distance ratio between the two nearest neighbours is close to one) and well-discriminated matches (OpenCV , s.d.). In our case, it was decided to set 0.8 as the threshold. Therefore, the similarity score that we assign to each pair of cropped images is exactly the number of good matches found that will populate the comparison matrix.

4.4 Integration of Inception v3 and OCR for enhanced image analysis

The previous sections presented the workflow that each image must go through to be correctly analyzed and that characterizes the outlier detection system. Once the specific steps required for each of the three feature extraction algorithms had been defined, it was decided to combine two of these approaches, namely Inception v3 and OCR. This choice was made because we wanted to prioritize feature extraction algorithms that provide a score based on a distance measure between images, thus ensuring a degree of objectivity in the estimation of similarity. However, this objectivity was lacking when using SIFT, where the comparison was based on the average number of good matches obtained from each image, which itself depended on the number of key points detected. Using both of these approaches in our system allows us to increase its robustness by overcoming the limitations that would arise if we relied solely on one of these approaches. For instance, if we were to exclusively employ OCR, we would encounter issues when the text in the images is not clearly visible, leading to incorrect classifications. Similarly, if we were to solely rely on Inception v3, the system would struggle when presented with distorted billboards or poorly illuminated images. By leveraging the strengths of both algorithms, we can compensate for these limitations and improve the overall performance and accuracy of the model.

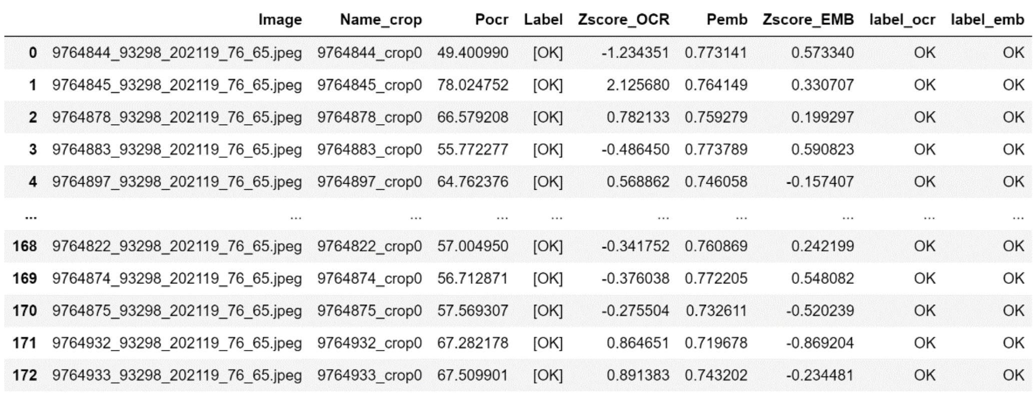


	Image	Name_crop	Pocr	Label	Zscore_OCR	Pemb	Zscore_EMB	label_ocr	label_emb
0	9764844_93298_202119_76_65.jpeg	9764844_crop0	49.400990	[OK]	-1.234351	0.773141	0.573340	OK	OK
1	9764845_93298_202119_76_65.jpeg	9764845_crop0	78.024752	[OK]	2.125680	0.764149	0.330707	OK	OK
2	9764878_93298_202119_76_65.jpeg	9764878_crop0	66.579208	[OK]	0.782133	0.759279	0.199297	OK	OK
3	9764883_93298_202119_76_65.jpeg	9764883_crop0	55.772277	[OK]	-0.486450	0.773789	0.590823	OK	OK
4	9764897_93298_202119_76_65.jpeg	9764897_crop0	64.762376	[OK]	0.568862	0.746058	-0.157407	OK	OK
...	...	...	...	...	...	...	...	...	...
168	9764822_93298_202119_76_65.jpeg	9764822_crop0	57.004950	[OK]	-0.341752	0.760869	0.242199	OK	OK
169	9764874_93298_202119_76_65.jpeg	9764874_crop0	56.712871	[OK]	-0.376038	0.772205	0.548082	OK	OK
170	9764875_93298_202119_76_65.jpeg	9764875_crop0	57.569307	[OK]	-0.275504	0.732611	-0.520239	OK	OK
171	9764932_93298_202119_76_65.jpeg	9764932_crop0	67.282178	[OK]	0.864651	0.719678	-0.869204	OK	OK
172	9764933_93298_202119_76_65.jpeg	9764933_crop0	67.509901	[OK]	0.891383	0.743202	-0.234481	OK	OK

Figure 4.10: Example of combined pruning table, where for each image the score of Inception v3 (Pemb) and the score of OCR (Pocr) were associated. In addition associated to each image there are the label assigned based on the OCR approach (label_ocr) and on the Inception v3 approach (label_emb).

Therefore, a further processing step was added before the final one. This step consists in combining the two pruning tables obtained by each of the two feature extraction algorithms. Then, the final classification of each image is done by considering two scores, the one obtained from the textual features extracted by OCR and the one obtained from the global features extracted by Inception v3. Once the threshold is set, an image is classified as a non-outlier only if both scores are greater than the threshold, in all the other cases the image is classified as an outlier.

Chapter 5

Results

In this Chapter are reported the results obtained from the tests carried out using the system described in Chapter 4 in its final form that includes the combination of Inception v3 and OCR approaches. All tests were performed by setting the threshold for normalized scores at x′ = -3, where x′ is computed, according to the definition of z-score normalization, as :

$$x' = \frac{x_i - \mu}{\sigma}$$

Where μ is the mean value, x_i is the data value and σ is the standard deviation.
First, we describe the datasets used for the tests conducted, followed by a discussion on the performance of our validation system, highlighting any significant findings or observations.

5.1 Dataset Description

The training datasets used for the evaluation consist of images already labelled by a human operator, coming from different advertising campaigns. Each campaign, namely *Esselunga*, *Visa*, and *About You*, has its own dataset, which may or may not include outliers. It is important to note that the datasets differ in terms of size, reflecting the varying scale and scope of each advertising campaign. Additionally, it is important to acknowledge that images of the same advertising campaign may

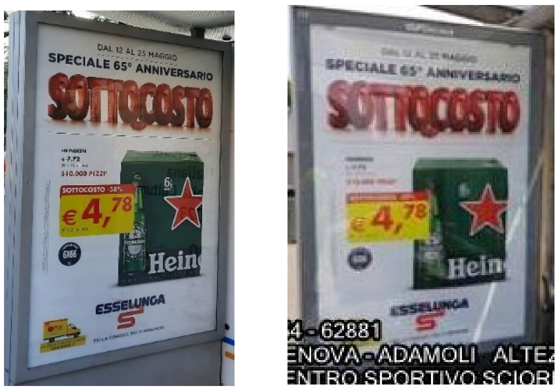


Figure 5.1: Comparison between the image supplied by Provider 1 (left) and one supplied by Provider 2 (right) from the same advertising campaign, ESSELUNGA.

come from different campaign providers, which may have different image quality. We will focus on two providers, which are called *Provider 1* and *Provider 2* for simplicity. Provider 1 typically supplies clear, high-resolution images, whereas Provider 2 provides low-resolution and therefore less clear images.
To guarantee better system performance, it is advisable to treat the images of each campaign separately for each provider. This approach is necessary because if we were to consider images acquired from both providers for a given campaign together, the lower-resolution images from Provider 2 would generally yield lower average similarity scores. Consequently, this would distort the score distribution, leading to a significant number of false outliers being erroneously positioned at lower score positions.

Ad Campaign	Provider	Total Images	Negative examples (OK)	Positive examples (KO)
VISA	Provider 1	175	171	4
VISA	Provider 2	267	266	1
ESSELUNGA	Provider 1	357	357	0
ESSELUNGA	Provider 2	889	889	0
ABOUT YOU	Provider 1	1057	1051	6

Table 3: Datasets composition with the total amount of pictures per dataset and the number of pictures per class.

Table 3 presents a breakdown of the datasets employed during the testing phase. It is important to note that not all datasets include outliers (KOs), allowing us to assess the model's performance in the absence of such instances. To explore variations and potential challenges, we specifically examined two providers for the initial two campaigns. This involved a single-instance campaign, namely VISA, and a multi-instance campaign, ESSELUNGA. Subsequently, a more extensive dataset was utilized for the ABOUT YOU campaign, which, as previously seen in Chapter 2, is multi-instance campaign. For this campaign, only Provider 1 was considered in the evaluation process.

5.2 Evaluation Metrics

Before looking at the specific results, it is essential to establish the evaluation metrics employed to assess the system's performance. For our analysis, we classify panels as either positive examples (outliers) or negative examples (non-outliers).
To measure the effectiveness of the campaign validation system, we utilized three key metrics: *Recall*, *Specificity* (or True Negative Rate - TNR), and *F2 score*. Recall has

already been introduced in the first task. The F2 score is an accuracy measure that combines Precision and Recall, with an emphasis on Recall over precision:

$$F2 = \frac{5 \times precision \times recall}{4 \times precision + recall}$$

We preferred this metric to the *F1 score* precisely because F2 score gives more weight to Recall than to Precision, whereas F1 score considers them to be of equal importance. We wanted to give more weight to Recall because we are more interested in the model correctly recognising all positive examples, i.e. having as high Recall as possible.

Specificity, on the other hand, represents the ability of the model to correctly classify negative examples in the data set. It is calculated as:

$$TNR = \frac{TN}{TN + FP}$$

The three metrics used provide, in our opinion, a comprehensive expression of the performance of the model in terms of the ability to correctly classify outliers (TPs) through Recall, in terms of the ability to correctly classify even negative instances through Specificity, and in providing an overall view of the relationship between Precision and Recall through F2 Score. In this particular application, *Precision* alone holds less significance. In general, outliers are present in extremely small numbers, on the order of a few dozen or less, in datasets with hundreds or thousands of images. Consequently, it is easy to find true positives in smaller numbers than the number of false positives. This generally results in very low Precision values. *Accuracy* is not suitable for unbalanced datasets such as ours.

5.3 Results analysis

In this section, we present the results obtained from the evaluation of each advertising campaign dataset, discussing the performance of our system.

5.3.1 VISA Advertising Campaign

The first advertising campaign we analyse is that of VISA. Examples of billboards of this single-instance ad campaign are shown in *Figure 5.2.*

In this case, two tests were conducted with two separate datasets of different dimensions, one for each provider.

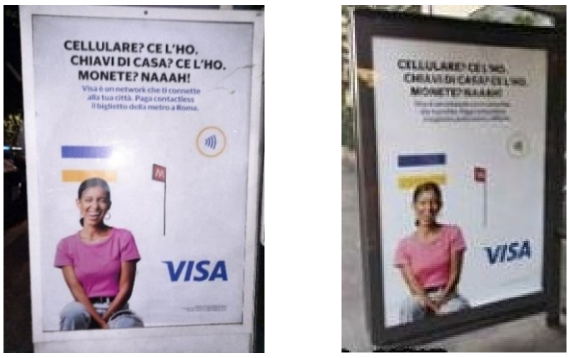


Figure 5.2: Example of billboards of VISA ad campaign. The left picture was provided by Provider1 and the right one by Provider 2.

Provider 1 test

PROVIDER 1			
OCR label	InceptionV3 label	Assigned label	Total images
OK	OK	OK	169
KO	KO	KO	2
OK	KO	KO	2
KO	OK	KO	0
Not processed images		KO	2

Table 4: Results for VISA, Provider 1

Table 4 shows the results for Provider 1. As can be seen, 169 images were classified as OK and 6 images as KO including two images where no advertising panel was detected ("Not processed images").

PROVIDER1					
TP	FP	TN	FN	Recall	SPECIFICITY
4	2	169	0	100%	98%

Table 5: Statistics of the results for VISA, Provider 1

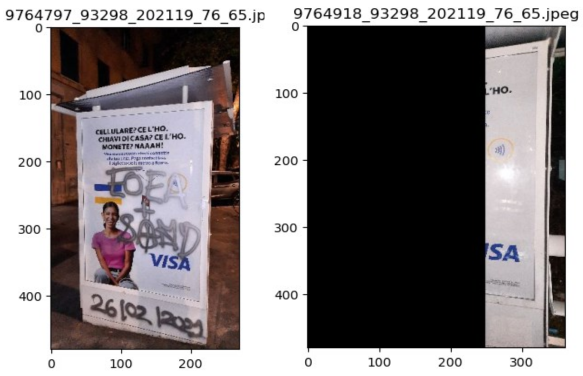

9764797_93298_202119_76_65.jp
9764918_93298_202119_76_65.jpeg
0
100
200
300
400
0
100
200
300

Figure 5.3: the FPs that were classified as OK from the OCR approach and KO from the Inception V3 approach.

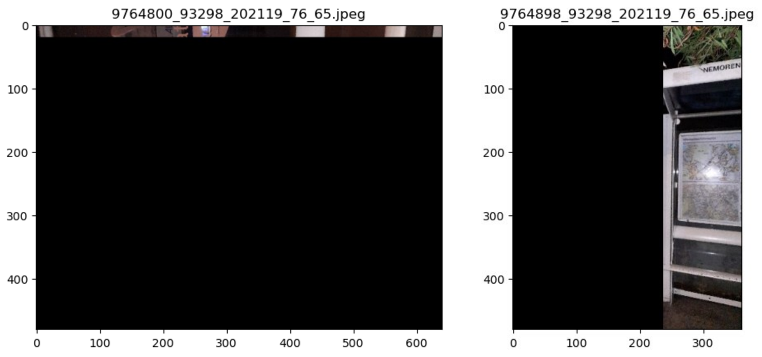

9764800_93298_202119_76_65.jpeg
9764898_93298_202119_76_65.jpeg

Figure 5.4 the images where no panel was detected

Among the different images, it is worth showing those for which no panel was detected and those that were misclassified as positive examples, i.e., the two FPs.
The images shown in *Figure 5.3* are the two false positives found. In this case, Inception V3 struggled to extract the features correctly because, as we can see in one image, the billboard is marred by graffiti, while the other image is damaged. While in In *Figure 5.4* we can clearly see the reason why the object detection model was unable to detect any panels, indeed, the two images are corrupted. In all cases, the corrupted images had already been made available in this way, so there must have been a problem uploading the photo to the cloud that compromised its content, which is not our fault.
In conclusion, the model was able to detect all the outliers in this test. It also detected two false positives which, as we saw, are justified by the reasons just mentioned above.

Provider 2 test

PROVIDER 2			
OCR label	**InceptionV3 label**	**Assigned label**	**Total images**
OK	OK	OK	262
KO	KO	KO	1
OK	KO	KO	3
KO	OK	KO	1
Not processed images		KO	0

Table 6: Results for VISA, Provider 2

Table 6 shows the results for Provider 2. It is important to say that the positive example we have in the test dataset was originally mislabeled as a non-outlier. However, the campaign validation model was able to find and classify it correctly. The image in question, shown in *Figure 5.5,* actually refers to another advertising campaign, so it can be concluded that an error was made in the human operator's control phase. Moreover, the image was labelled as KO by both approaches.

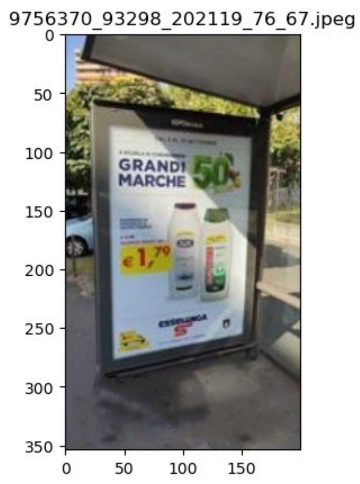


Figure 5.5: This image was labelled as not outlier but actually it is because it belongs to another ad campaign.

For this test, therefore, statistics were recorded as shown in *Table 7*.

PROVIDER 2					
TP	FP	TN	FN	Recall	Specificity
1	4	262	0	100%	98%

Table 7: Statistics of the results for VISA, Provider 2

The results suggest that the model has a maximum Recall rate, and very high Specificity, indicating the ability correctly classify most of negative instances. Below we can see which examples are classified as FP.

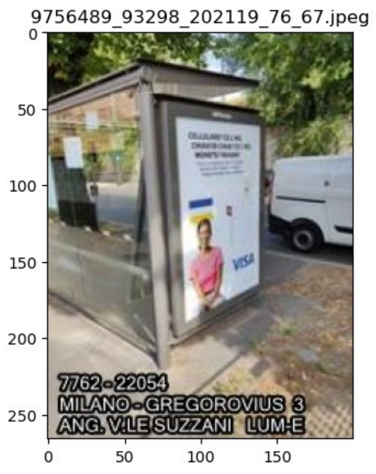


Figure 5.6: Example that belongs to the case where OCR assigned the label KO and Inception V3 the label OK.

In *Figure 5.6* we observe the case where the image was classified as KO by OCR and as OK by Inception V3. In fact, due to the low quality of the image and the distance from the lens, the OCR was not able to correctly detect the text, while Inception V3 managed to correctly classify the image. In *Figure 5.7*, on the other hand, we observe examples of the opposite case, where OCR was able to extract and recognise the text correctly, while Inception V3 had difficulties due to the poor lighting conditions and the presence of shadows.

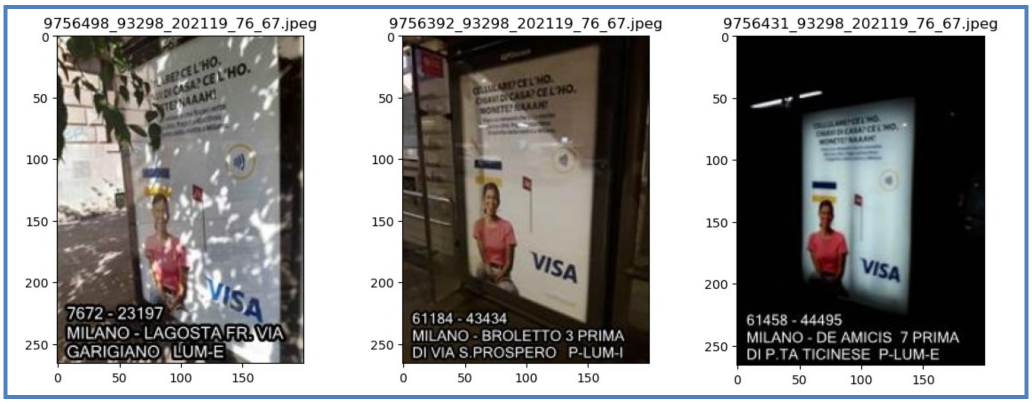

9756498_93298_202119_76_67.jpeg
7672 - 23197
MILANO - LAGOSTA FR. VIA
GARIGIANO LUM-E
9756392_93298_202119_76_67.jpeg
61184 - 43434
MILANO - BROLETTO 3 PRIMA
DI VIA S.PROSPERO P-LUM-I
9756431_93298_202119_76_67.jpeg
61458 - 44495
MILANO - DE AMICIS 7 PRIMA
DI P.TA TICINESE P-LUM-E
VISA

Figure 5.7: Examples that belong to the case where OCR assigned the label OK and Inception V3 the label KO

Image pre-processing techniques did not prove effective in improving model accuracy. In the latter case, for example, histogram equalization applied to the image dataset led to a deterioration in performance, as the outlier was not detected (received a score slightly above the -3 in both approaches) and the same number of FPs were detected as in the case without equalization, although with different images. This is probably because the available datasets are characterised by a very heterogeneous set of images, varying in resolution, brightness conditions, presence of obstacles and occlusions, visual clarity of the billboard, etc. Consequently, the lack of a standard image acquisition procedure means that the application of pre-processing techniques improves the quality of some images, but at the same time degrades the quality of others.

5.3.2 ESSELUNGA advertising Campaign

We move now on to a multi-instance ad campaign, which is ESSELUNGA. Again, we considered two datasets, one for each provider. In this case, neither contains negative examples. The test on a dataset without outliers was carried out to assess how the model behaves in 'normal' or 'no outlier' situations (not always provided datasets include outliers). This type of test helps to understand whether the model can generate false positives (FPs) in cases like this. However, it is important to emphasise that evaluating performance in the absence of outliers is not the main objective of the model, but rather the ability to correctly detect outliers in realistic situations.

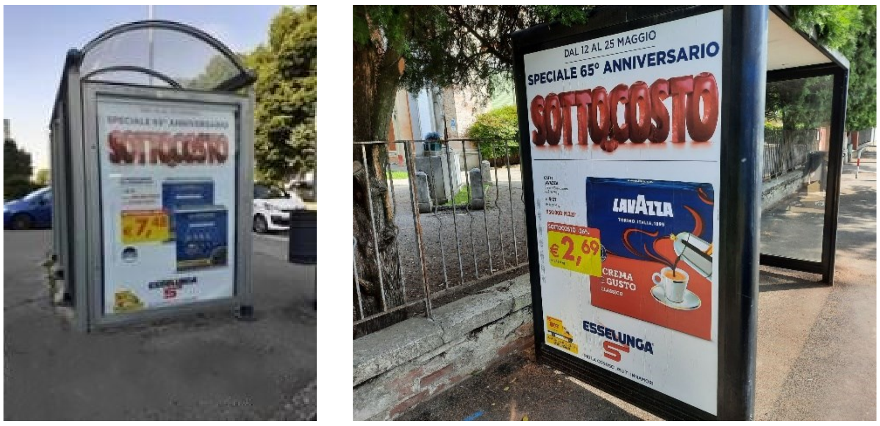


Figure 5.8: Examples of instances of ESSELUNGA ad campaign. On the left is an instance of Provider 2 and on the right is an instance of Provider 1.

Provider 1 test

PROVIDER 1			
OCR label	**InceptionV3 label**	**Assigned label**	**Total images**
OK	OK	OK	352
KO	KO	KO	0
OK	KO	KO	5
KO	OK	KO	0
Not processed images		KO	0

Table 8: Results for ESSELUNGA, Provider 1

With a larger dataset and no outliers, there were only 5 false positives, which is a very good result. Since Recall would be uninformative in this situation, due to the absence of outliers, we will consider Specificity as the main evaluation metric.

PROVIDER 1					
TP	**FP**	**TN**	**FN**	**Recall**	**Specificity**
0	5	352	0	/	98%

Table 9: Statistics of the results for ESSELUNGA, Provider 1

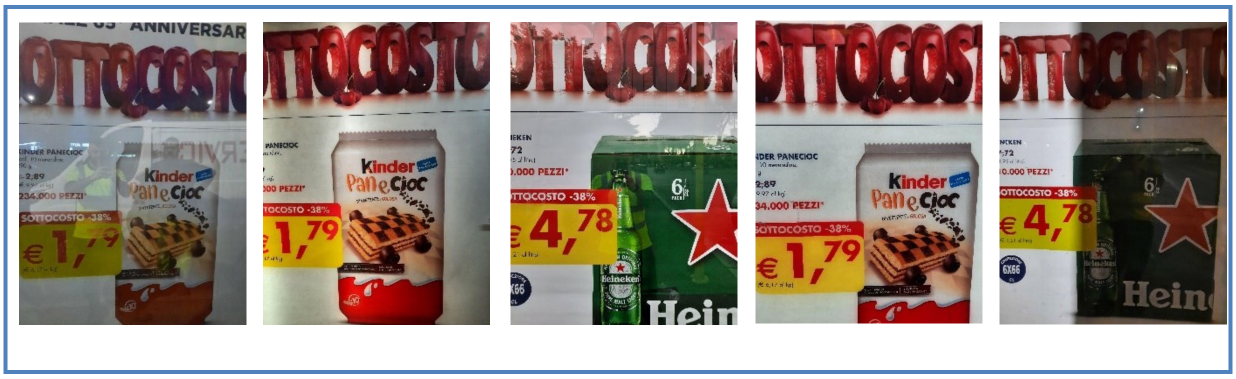


Figure 5.9: KOs classified by the model

The misclassified images, shown in *Figure 5.9*, were only labelled KO when approached with Inception V3. This is probably because the pictures were taken too close, cutting off part of the billboard.

Provider 2 test

PROVIDER 2			
OCR label	**InceptionV3 label**	**Assigned label**	**Total images**
OK	OK	OK	866
KO	KO	KO	4
OK	KO	KO	9
KO	OK	KO	7
NaN	OK	KO	3
Not processed images		KO	0

Table 10: Results for ESSELUNGA, Provider 2

We can see the presence of a new case, NaN-OK, which is eventually classified as KO by the model. Indeed, the OCR did not detect any text string on the three images shown in *Figure 5.10*.

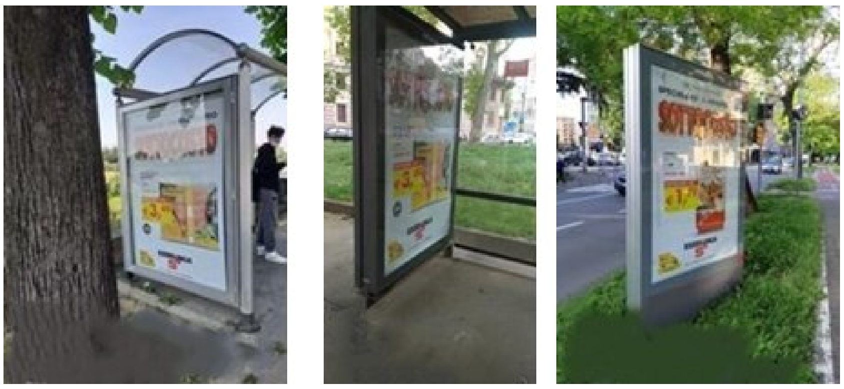


Figure 5.10: Examples where the OCR did not detect any text string

This is probably due to the presence of reflections on the billboards and the poor image quality.
Another case to investigate is where images were labelled OK from the OCR and KO from the Inception v3. Some examples are shown in *Figure 5.11*. What these images have in common is that the advertisement is facing sideways from the lens. This shows how Inception v3 struggles to embed the image correctly in these cases. It is also interesting to note that, although barely visible, the OCR has managed to extract some of the text correctly, demonstrating great robustness to borderline cases such as these.

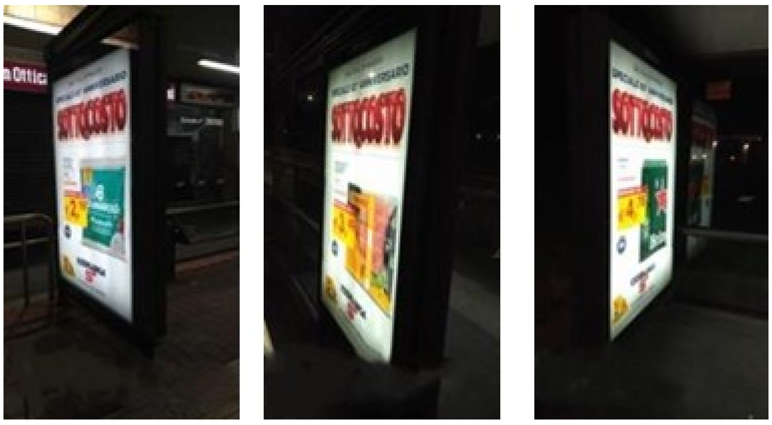


Figure 5.11: Example of the case where the OCR label is OK and the Inception V3 label is KO

PROVIDER 2					
TP	**FP**	**TN**	**FN**	**Recall**	**Specificity**
0	23	866	0	/	97%

Table 11: Statistics of the results for ESSELUNGA, Provider 2

Also in this case we gave more relevance to the Specificity highlighting once again the high performance of the system.

5.3.3 ABOUT YOU advertising campaign

The final test was run on the largest dataset, related to the About You campaign we saw in Chapter 2, where six outliers are present. The aim is to see if the model performs similarly to the previous tests where it was subjected to smaller datasets or similarly sized datasets but without outliers. A multi-instance campaign was also considered in this case.

OCR label	InceptionV3 label	Assigned label	Total images
OK	OK	OK	1038
KO	KO	KO	0
OK	KO	KO	3
KO	OK	KO	9
NaN	OK	KO	2
NaN	KO	KO	3
Not processed images		KO	2

Table 12: Results for ABOUT YOU ad campaign, Provider 1

PROVIDER 1					
TP	FP	TN	FN	Recall	Specificity
6	13	1038	0	100%	98%

Table 13: Statistics of the results for ABOUT YOU, Provider 1

The model confirmed the performance recorded in previous tests. In this case, we have five images where no text was detected and two where no billboards were found.
One interesting aspect of this test is worth mentioning. During the test, we observed two interesting cases. Firstly, we identified an additional outlier that had not been previously labelled as such in the test dataset even if it did not actually belong to the "About You" campaign (*Figure 5.12*). Additionally, we found that one of the images, actually belonged to the "About You" campaign, had been mistakenly labelled as an outlier in the test dataset (*Figure 5.13*). It is noteworthy that our model was able to

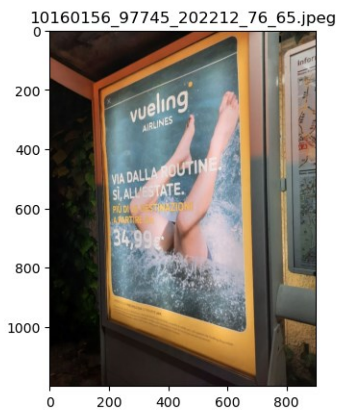


Figure 5.12: The image that was erroneously labelled as not outlier.

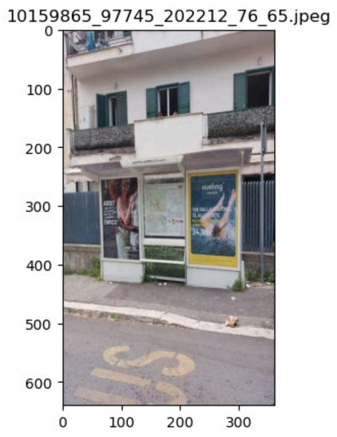
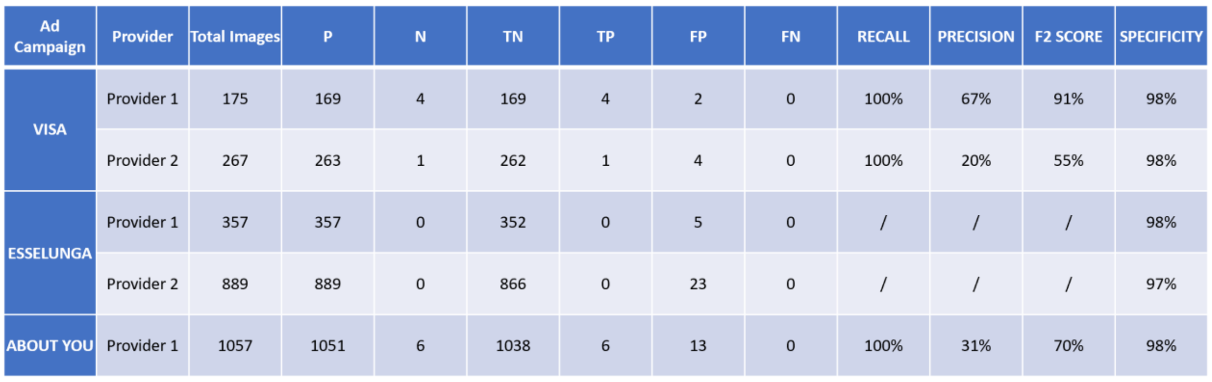


Figure 5.13: The image that was erroneously labelled as an outlier. The left ad panel clearly belongs to the 'About you' ad campaign.

correct both errors, successfully detecting the unlabelled outlier image and correctly classifying the image previously mislabeled as an outlier. Therefore, for performance analysis, there remain six actual outliers in the dataset, all correctly classified as TP by our model. These results highlight the ability of the model to identify and rectify labelling errors, thereby improving the reliability of the advertising campaign validation system.

5.4 Overall Performance and Discussion



Ad Campaign	Provider	Total Images	P	N	TN	TP	FP	FN	RECALL	PRECISION	F2 SCORE	SPECIFICITY
VISA	Provider 1	175	169	4	169	4	2	0	100%	67%	91%	98%
	Provider 2	267	263	1	262	1	4	0	100%	20%	55%	98%
ESSELUNGA	Provider 1	357	357	0	352	0	5	0	/	/	/	98%
	Provider 2	889	889	0	866	0	23	0	/	/	/	97%
ABOUT YOU	Provider 1	1057	1051	6	1038	6	13	0	100%	31%	70%	98%

Figure 5.14: Summary table of the results

The table above provides a summary of the test results conducted, including the addition of the F2 score to provide a comprehensive view of Precision and Recall, giving a greater emphasis on Recall.

First and foremost, it is worth noting that in all cases, we achieved maximum Recall by successfully detecting all outliers present in the dataset. In scenarios where there were no positive examples in the dataset, we evaluated the model's performance in correctly classifying negative examples. Once again, the model performed very well, exhibiting high Specificity values. The F2 score yielded results above 70% when the dataset was provided by Provider 1, indicating a robust performance. However, it showed a slightly lower value of 55% when the dataset was obtained from Provider 2, where the image quality was poorer. In this case, the prevalence of false positives in the model's predictions significantly affected the Precision, subsequently resulting in a decline in the F2 score. It is important to note the increase in false positives as the dataset size grows. This response was expected, but the important thing is that the false positive rate (FPR) remained below 3% of the total number of negative examples. Consequently, precision also drops as the size of the dataset increases.

It must be considered that the number of outliers to be found is very low compared to the size of the analysed dataset, so it is very easy for the precision to drop dramatically even with a few FPs detected.

Chapter 6

Conclusions

In this work we provided a possible pipeline for a control and validation system for advertising campaigns. The aim is to automate or semi-automate a process that would otherwise take up a lot of time and resources that could be used elsewhere. This process consists of checking a large number of images, captured every two weeks and stored in the cloud, to find images that have been tagged with the wrong advertising campaign. The system was divided into two macro modules according to the task at hand: the object detection module and the similarity estimation and outlier detection module.

The object detection module was used to detect the advertising panels in the images and isolate them from the surrounding context, thus focusing attention on the most important part of the image, i.e. the billboard. This part was implemented using a proprietary Oracle model via the Oracle Cloud Infrastructure (OCI) platform. The similarity estimation and outlier detection module was the most difficult to implement. It consists of a series of blocks assigned to different functions, in which the image is reduced to a feature descriptors vector, compared with all others to measure its similarity, analysed and then classified as an outlier or non-outlier. The most challenging part was analysing the scores obtained from the comparisons to identify those that were outliers. This part required the application of statistical knowledge and the use of mathematical tools to extract the right information we needed. The use of two approaches based on different image features made the system more robust and allowed high overall performance.

However, there are still several issues that may be addressed in future improvements.

In some cases, Provider 2 supplied an advertising campaign dataset consisting of low-resolution images that made text recognition impossible. As a result, the OCR approach produced unreliable and consistently low scores for image comparisons. To address this issue, a potential improvement would be to implement a reliability check by calculating the median of the score distribution. By setting a threshold, the system could determine the reliability of the OCR scores. If the median is above the threshold, the scores are considered reliable, but if it is below the threshold, the scores are considered too low. In the latter scenario, the classification process would

be based on the Inception v3 approach alone. This solution would also solve the problem of handling advertising campaigns with no text, which would give us null scores from OCR. Another issue with OCR is the edge case where two different campaigns might be characterised by similar text. Words such as 'SOTTOCOSTO' and 'SCONTO' could register similar scores using Levenshtein distance. In this case, further research into more accurate text comparison techniques might resolve this issue. Finally, the last problem that arises is when the dataset for a particular campaign contains only a small number of images, say a few dozen. In this scenario, the reliability of the model cannot be guaranteed, as it was designed for larger datasets. For this reason, it was decided to formally set a threshold of at least 50 images per dataset. Future analysis may suggest a more precise threshold that can handle the trade-off between the number of images per dataset and the value of reliability of the model.
The work carried out in this thesis could provide a solid support for the development of image analysis algorithms in areas other than advertising. Indeed, the pipeline developed could be used in general to search for out-of-context images within context-specific datasets.

References

[1] Analytics Vidhya. (n.d.). *A Simple Guide to Metrics for Calculating String Similarity*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/02/a-simple-guide-to-metrics-for-calculating-string-similarity

[2] Ashish Vaswani, N. S. (2017). Attention is All You Need. *Arxiv*. Retrieved from https://arxiv.org/pdf/1706.03762.pdf

[3] Christian Szegedy, V. V. (2015, December). *Arxiv*. Retrieved from Rethinking the Inception Architecture for Computer Vision: https://arxiv.org/abs/1512.00567

[4] Darwin Bautista, R. A. (2022, july). *Scene Text Recognition with Permuted Autoregressive Sequence Models*. Retrieved from arXv: https://arxiv.org/abs/2207.06966

[5] Girshick, R. a. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[6] Gupta, S. (n.d.). *Top 5 Distance Similarity Measures implementation in Machine Learning*. Retrieved from Medium: https://medium.com/@gshriya195/top-5-distance-similarity-measures-implementation-in-machine-learning-1f68b9ecb0a3

[7] IBM Corporation. (2023). *What is computer vision?* Retrieved from IBM Corporation Web Site: https://www.ibm.com/topics/computer-vision

[8] Joseph Redmon, S. D. (2016). You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[9] Liu, L. O. (2019, October 31). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*. Retrieved from Springer Link: https://doi.org/10.1007/s11263-019-01247-4

[10] Lowe, D. G. (2004). *Distinctive Image Features from Scale-Invariant Keypoints*. Retrieved from SpringerLink: https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

[11] OpenCV . (n.d.). *OpenCV - Open Souce Computer Vision*. Retrieved from Feature Matching with FLANN: https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html

[12] Oracle. (n.d.). *Overview of Object Storage*. Retrieved from Oracle Cloud Infrastructure Documentation: https://docs.oracle.com/en-us/iaas/Content/Object/Concepts/objectstorageoverview.htm

[13] Raj, B. (2018, may). *A Simple Guide to the Versions of the Inception Network*. Retrieved from towardsdatascience: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[14] Singh, A. (2019). *SIFT Algorithm | How to Use SIFT for Image Matching in Python*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python

[15] Stanford Institute for Human-Centered Artificial Intelligence. (2023). *THE AI INDEX REPORT*. Retrieved from Measuring trends in Artificial Intelligence: https://aiindex.stanford.edu/report/

[16] Vidhya, A. (2023). *SIFT Algorithm | How to Use SIFT for Image Matching in Python* . Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

[17] Wei Liu, D. A.-Y. (2016). SSD: Single shot multibox detector. *ECCV.*

[18] Wikipedia. (n.d.). *Cosine similarity*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Cosine_similarity

[19] *YOLOv8*. (n.d.). Retrieved from Ultralytics: https://docs.ultralytics.com/models/yolov8/

[20] Youngmin Baek, B. L. (2019, April). *Character Region Awareness for Text Detection*. Retrieved from arXiv: https://arxiv.org/abs/1904.01941