Master in Informatics Engineering
Internship
Final Report

SABADO - SmArt BrAnd DetectiOn

Gonçalo Filipe Palaio Oliveira
gpalaio@student.dei.uc.pt

Advisors:
Bernardete Ribeiro
André Pimentel
Date: 2nd September 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Master in Informatics Engineering
Internship
Final Report

SABADO - SmArt BrAnd DetectiOn

Gonçalo Filipe Palaio Oliveira
gpalaio@student.dei.uc.pt

Jury:
César Teixeira
David Palma

Advisors:
Bernardete Ribeiro
André Pimentel
Date: 2nd September 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

The need for an automatic system that detects brands in digital content grows simultaneously with its pervasiveness and all the marketing business surrounding it. Most of the current used solutions use implicit placement of advertisements without regard for content or context, which has been shown to be an effective way of raising brand awareness. Moreover, knowing where a brand is used (and how), could serve as support for business decisions.

This work aims to explore new ways of assessing the presence of branded products or services in images using machine learning and computer vision techniques. A brand is much more than a graphic logo, it can defined by several other subtle attributes, for example, how it is advertised and by its reputation. In this work, we tackle the problem of brand detection through the graphic logo, since it is the most tangible element of a brand.

Given all the possible ways a graphic logo can appear in an image, building a highly accurate system is a challenging task. We propose a brand detection system that takes advantage of modern *deep learning* computer vision systems. We built several models, tuned and tested them using the FlickrLogos-32 dataset. Using the Fast Region-based Convolutional Networks (FRCNs) model our system yields a mean average precision value of 73.47%, outperforming previous methods while also having reasonably fast detection time and outputting multiple logo detections per image.

From an application standpoint, many challenges still remain, but with advances recently done in computer vision and machine learning, we will come to reach a system capable to even capture subtle characteristics of a brand and help companies and brands captivate their customers and to make better decisions.

Keywords: Brand detection; Computer vision; Machine learning

Resumo

Com o aumento de conteúdo digital disponível e todo o negócio de publicidade que o envolve, companhias e marcas começam a ter necessidade de sistemas capazes de analisar e extrair informação útil de forma automática desse conteúdo. Grande parte das soluções de publicidade integrada usadas correntemente, não têm em conta o tipo de conteúdo e o contexto presente, informação que tem dado provas de ser uma forma efectiva de aumentar o conhecimento e reconhecimento de marcas por parte dos clientes. Esta informação também útil para suportar decisões, tirando partido de que forma e onde uma marca é utilizada.

Este trabalho tem como objetivo explorar novas formas de avaliar a presença de produtos ou serviços de uma marca em imagens tirando partido de técnicas de visão e aprendizagem por computador. Uma marca é definida não só pelo logotipo, mas também através de outros atributos, alguns mais subtis, como por exemplo, como é publicitada e a sua reputação. Neste trabalho, lidamos com o desafio de deteção marcas através de logotipos, dado que é a representação mais tangível de uma marca.

As várias formas em que um logotipo poderá aparecer numa imagem, fazem desta tarefa um desafio. Propomos um sistema de deteção de marcas que tira partido de técnicas visão computacional usando *deep learning*. Construímos vários modelos que optimizamos e testamos usando o dataset FlickrLogos-32. Utilizando o modelo *Fast Region-based Convolutional Networks* (FRCNs) o nosso sistema atinge um valor de *mean average precision* de 73.47% superando métodos anteriores com tempos de deteção razoáveis e fornecendo múltiplas deteções por imagem.

Ainda assim, de forma a ser aplicado num ambiente real, continuam a existir vários problemas. Com os avanços correntes na área de aprendizagem e visão computacional, eventualmente, será possível chegar a um sistema que é capaz de detetar até características mais subtis de uma marca de forma a suportar empresas e marcas e as relacionar com os seus clientes.

Keywords: Deteção de marcas; Visão por computador; Aprendizagem por computador

Acknowledgements

This work would not have been possible without the support of multiple people.

I would like to express my gratitude to Professor Bernardete Ribeiro, for her support, guidance and encouragement.

To EyeSee, particularly Eng. André Pimentel, for all the support, advice and trust. In addition, i also would like to thank Eng. Xavier Frazão and Eng. João Redol for the insightful discussions and the invitation to visit the EyeSee facilities.

Gratitude is also due for all the other people that directly or indirectly supported during this time, specially my colleagues and friends.

I am also indebted to my family for their unwavering love, encouragement and support throughout my life.

Contents

List of Figures

List of Tables

Abbreviations

API Application Programming Interface

AP Average Precision

BoW Bag of Words

BRIEF Binary Robust Independent Elementary Features

CCTV Closed-circuit television

CNN Convolutional Neural Network

DNN Deep Neural Network

DoG Difference of Gaussians

DPM Deformable Part Model

FAST Features from Accelerated Segment Test

FC Fully Connected

fps Frames Per Second

FRCN Fast Region-based Convolutional Network

FV Fisher Vector

GPU Graphics Processing Unit

HOG Histogram of oriented gradients

ILSVRC ImageNet Large Scale Visual Recognition Competition

IoU Intersection Over Union

LoG Laplacian of Gaussian

mAP Mean Average Precision

NNS Nearest Neighbor Search

ORB Oriented FAST and Rotated BRIEF

PASCAL Pattern Analysis, Statistical Modelling and Computational Learning

R-CNN Regions with CNN features

RANSAC Random Sample Consensus

RBF Radial basis function

RGB Red, Green and Blue

RoI Region of Interest

RPN Region Proposal Network

SGD Stochastic gradient descent

SIFT Scale-Invariant Feature Transform

SURF Speeded Up Robust Features

SVM Support Vector Machine

USPTO United States Patent and Trademark Office

VOC Visual Object Classes

Chapter 1

Introduction

As the Internet grows and becomes more ubiquitous, companies and brands have more and more ways to reach customers and are now taking advantage of this fact to build new opportunities and better understand the needs of their customers. The possibility to connect real life situations to virtual information grows as the amount of media available and consumed in the Internet rises. As the field becomes crowded, there is a necessity of optimizing the message passed to the customers and to provide the message when its needed. Kenny and Marshall [38] envisioned the future of internet business as being focused more on context than content, while foreseeing that companies should build systems specifically to perform data analysis and thus better directed to the needs of the costumers. Presently there is a trend towards that direction.

The SABADO (SmArt BrAnd DetectiOn) project attempts to answer some of those needs, connecting brands to customers through its content and analysis. In this work we explore brand detection in images for that purpose, but in the long run, the goal is video analysis, ideally in real-time.

1.1 Motivation

Advertisement is the primary way to get revenue or gathering interest for a product or service. Although television advertisement is one of the most pervasive advertisement formats, the shift towards digital and internet media brings forward the need for new ways of gathering interest for products and services.

Digital media advertisement brings different ways of controlling the message transmitted to the costumers, enabling Advertisers to create a message that is relevant and focused on the target audience. There are multiple methods of advertisement delivery

that Advertisers use to maximize their reach and to improve on how they target their customers. Contextual targeting is one of these methods. It is based on the concept that if one shows advertisements that are contextually relevant to the content, the user is most likely to take part of it. Google's Adsense and Adwords are prolific in web content and are an example of such type of targeting. The delivery of digital content, like video, often relies on showing advertisements during or in-between content. This generates awareness for the respective product or service, even if someone is not aware of the need for it. In order to well target the user, there is a need to analyse the content and relevant advertisements that exist nowadays.

Most of the current solutions use implicit placement of advertisements without regard of context or content, which has been shown to be an effective way of raising brand awareness.

1.2 Goals and Contributions

The general purpose of this project is the research and development of an image brand detection system prototype. In this context, brands pertain to graphic logos or object features that are characteristic to a brand or company. This project was created in collaboration with EyeSee, a technological company that focus on novel digital advertising solutions.

This problem has many challenges from two perspectives: computer vision and pattern recognition. Regarding the former, earlier methodologies have been used to answer these challenges. With respect to the latter, we take on the challenge to apply Convolutional Neural Networks (CNNs) for brand detection.

We extend and build upon general concepts and models of object recognition in images. Thus, we propose a solution that takes advantage of specific characteristics of brands based on current cutting edge-research.

Our proposed system is based on CNNs which presents high accuracy in a benchmark using the FlickrLogos-32 dataset evaluated through adequate metrics for object recognition. We focus on company graphic logos since they give a distinctive way of assessing a particular advertising campaign or brand.

The system we propose uses transfer learning to leverage image representations learned with CNNs on large-scale annotated datasets. The transferred representation leads to significantly improved results for brand detection. This is a very important innovation

in this study because we have shown empirically that it performs well and can be as well applied in other contexts and problems.

More specifically, since content can appear in several forms, the proposed system should be prepared to handle real world images that are subject to varying conditions, background noise and not specifically created to display a certain brand in an evident way. Knowing where the brand or graphic logo is located in the image is not a requirement, but should be kept in mind for contextual advertisement insertion in the image.

Our contributions in this Dissertation attempt to answer partially to the challenging problem of brand detection. In summary, we show that the methodology employed by using CNNs and transfer learning outperforms the empirical study with traditional computer vision techniques on the same problem.

1.3 Thesis framework

This study describes the work developed in the area of brand recognition at EyeSee, for the SABADO - SmArt BrAnd DetectiOn project, as part of the Thesis/Project course of the Informatics Engineering Masters at the University of Coimbra.

EyeSee is a technological startup, funded by Olisipo, focusing on intelligent systems. As of today, EyeSee developed a novel digital advertising solution with unique insertion and interaction features, owning 5 patents in the United States Patent and Trademark Office (USPTO).

1.4 Thesis structure

This document is organized as follows:

- Chapter 1, Introduction – the current chapter aims to introduce the motivation of the dissertation, what are the goals of the project aims to achieve and what is the context.
- Chapter 2, Brand Detection: earlier work – in this chapter we describe related work performed with brand and graphic logo detection in mind, then we describe commercial solutions that perform brand detection in several kinds of media.
- Chapter 3, Feature Extraction in brand images – this chapter gives an outline of image features that have been used to perform brand detection and other object

detection tasks. We first describe techniques that use mostly texture information followed by features that use color characteristics present in images.

- Chapter 4, Model approaches for brand detection – this part of the dissertation describes models that have been successful used in object recognition. We start by describing classical models that have been successful in such specific tasks. What follows is a description and several details of recent methods using Deep Neural Network Models, considered today, the primary candidate for computer vision tasks.

- Chapter 5, Computational Experiments – we describe the experiments performed with some models from the previous chapter. The data used in the experiments is first described, followed by experiments with classical and recent Deep Neural Network models, finally we analyze and compare results obtained in the experiments.

- Chapter 6, Final Conclusions and Future Work – in this final chapter, we review the work and reflect on the accomplished achievements. To conclude, we present some ideas that could improve the current work.

- Appendix A, Datasets – in this appendix, several datasets closely related to our task are described.

- Appendix B, Fast R-CNN additional results – we present additional results using our final proposed model, in particular per class precision and recall values.

- Appendix C, Minutes of the meetings – we give a summary of each formal meeting held over the year.

Chapter 2

Brand Detection: earlier work

This chapter reviews the literature and current business landscape regarding brand detection in videos or images. The main goal is to identify possible research directions.

Brand logos help to assess the identity of something or someone. Most solutions use brands as the main target of detection since they often present distinct shapes and appear in high contrast regions. They can also appear placed into common objects, depending on what product the brand is promoting. In this case they are often subject to multiple angles and sizes, varying lighting conditions and noise. Therefore, although they give perceptual distinctiveness, it is a challenge due to all the conditions they can appear in.

Information retrieved from brand information with these systems has a multitude of applications among them. Several examples have been reported in the literature, such as automatic identification of products, improved product search and recommendation in online stores, targeted advertising and business intelligence.

We go through several works that have addressed brand detection in images and video, most using features like Scale-Invariant Feature Transform (SIFT), followed by recent approaches using Deep Neural Networks (DNNs). We then describe commercial software products that use or perform some form of brand detection.

2.1 Graphic logo recognition

Keypoint extraction is one of the most widely used feature extraction methods. Local feature representations or descriptors are extracted from the picture using algorithms that detect regions of interest. These provide representations that are invariant (unaltered by) to affine transformations and robust when facing lighting conditions and

clutter. The Scale-Invariant Feature Transform (SIFT) feature descriptor from Lowe [42] and the, Speeded Up Robust Features (SURF) from [14], partially inspired by SIFT are broadly used algorithms which provide these types of representations. Features extracted from these algorithms also inherit characteristics that help to avoid common problems of clutter, visibility and even partial occlusion in images [42].

Liao et al. [41] proposed a system that associates advertisements to viewed videos, optimizing the revenue based on advertisers bids and budget. This system matches feature points to the image object that the advertiser chooses as a key characteristic or object to advertise against. They coin the term *adImage* to describe the query graphic logo. The system starts by extracting SIFT feature descriptors around detected keypoints. Then these descriptors are matched against the adImage using an approximate nearest neighboring indexing method. After outlier feature descriptors are removed using spatial constraints.

Spatial and geometric context is built around extracted SIFT keypoints in [52]. SIFT keypoints and their respective context are used to find point correspondences between graphic logos in two images. Similarity is checked using built adjacency matrices that model interactions between interest points and their respective context at different orientations and locations. A parameter τ is used to control the fraction of matched interest points that are sufficient to assess the presence of a logo in the image. This solution shows lower recognition error rates than a generic SIFT keypoint one, while allowing to control cases where partial occlusion occurs.

The work by Wang et al. [59] focuses on recognition under troublesome conditions, that is, situations where the size of logo might be small, texture is simple, and where they might be blended into the background. Feature extraction is performed over a set of possible locations where a logo might appear. The detection of meaningful locations (where a logo might appear) is done using the JSeg segmentation algorithm [23]. It produces a region tree with images of different scales (ensures the logo appears in at least one of the tree levels). The produced tree is pruned according to leafs image perimeter and area in order to reduce the search space size problems. The segmentation step aims to reduce the background noise and keeps the contour integrity.

A clear type distinction is done between graphic logos using features from which a good logo model can be obtained.

- Point type - Logos where a good number of SIFT can be extracted. SIFT keypoints are extracted for each segment.
- Shape type - Flat logos where their shape context is the most distinctive feature. Around all keypoints, edges are detected using the Canny Operator from [17] ,

the nearest contour is then segmented into bins and a log-polar histogram is built. This histogram is used to find correspondences between shapes.

- Patch type - Their color distribution is the most distinctive feature. A color histogram is built using gray scale values from the segmented section.

In order to improve the performance of logo detection, features related to these three types are combined for matching.

In [60] feature trees are built for each logo category, the trees are internally sorted by discriminability and spatial distribution of every feature, based on the training samples. Feature trees are built by first selecting discriminative regions of a logo, splitting each region into several clusters, finally the tree will contain the features from each region. Detection is performed by finding similar sets of features in the image using the previously built feature trees.

For logo matching, SIFT features are first extracted from the query image, and using the same clustering parameters, the interesting feature regions are setup for better recognition. Similarity is first measured using the root nodes of the tree set. Euclidean distance to the feature points is used to build a confidence value that the current logo features matches the region. Using a threshold value the algorithm avoids checking all the tree nodes, and if the confidence value is low, lower levels will not be checked. Bounding boxes proposals can then be built using the highest matched regions.

Applied to video, Hall et al. [34] propose an architecture for logo detection and tracking in videos using scale-normalized receptive fields over limited regions for recognition and a Kalman filter for tracking. Detection and tracking is done in separate modules. After a logo is detected in a certain region, the tracking module then maintains information about the region, following the movements of the logo thus avoiding detection of the current logo in subsequent frames of the video. The scale-normalized receptive field approach is compared to detection using only chrominance histograms, showing lower values of false positives than the former method, as expected, relying only on color is not optimal, since not all color distributions on reference logos are really discriminative, subject to varying lighting conditions and sometimes blend into the background, although this is approach is faster in terms of computation.

As a final remark, some improvements to the model are suggested. Since at all times the most likely logo is computed for all regions of the video frame, the system will perform badly when there is no logo. It is suggested an additional classifier for a non-logo class using misclassified samples. Shape features are also suggested to reduce the number of false positives due to limited discriminability of the measured features in the set.

A popular method for logo retrieval has been the Bag of Words (BoW) model. A typical BoW representation uses localized descriptors, such as SIFT and SURF, clustered into a vocabulary of visual words. This representation encodes images as an histogram by taking into account image patches that are present in the image and are common to the built vocabulary. The lack spatial information in this representation, is a problem that has been tackled by several authors. Spatial information could help solve cases where two similar visual words are ambiguous without taking into account where the are in the image.

Wan et al. [58] propose a shape descriptor built using extracted keypoints. Using each extracted of the keypoints, tree representations of the logo are created by taking into account the neighborhood of keypoints around the root keypoint, in the training phase tree representations are mined using spatial constraints and indexed using the root keypoint. The retrieval phase uses the built index and takes into account spatial constraints of the tree representation.

FIGURE 2.1: Example of a Tree-based Shape Descriptor in four logo images [58].

Some brand logos rely strongly on color components to be distinctive. In [43] color information is used, taking advantage of inherent characteristics of localized descriptors. Color-localized spatial masks are used using scale and orientation information that is inherent to keypoint based extractors. Usually ignored when using these descriptors, color layout information of the brand logo is captured as a complement.

The baseline method used follows the bag of words model with the description of image patches given the SIFT algorithm, color layout information is then added to the model. Around each detected keypoint in the image, a square mask is applied to the image

patch. The orientation and scale of this mask is determined by the current keypoint scale and orientation information, rotation invariance information is then, taken into account. Having this square mask, local color information is extracted from each of the four triangles formed by the mask using RGB color histograms. Spatial information is captured by ordering the color histograms clockwise from the orientation of the keypoint. Using a similar encoding method to the BoW model, color information is retrieved and indexed into fewer color information vectors using a clustering algorithm. Visual and color words are finally indexed for each image in the dataset.

FIGURE 2.2: Examples of color-localized keypoints with the respective spatial masks drawn [43].

The authors of [48] propose a image retrieval system using bundles of local features. By aggregating features of the spatial neighboorhood, sets of discriminative features that hold more information than a single one are built. By querying a database of reference images mapped by those bundles, logo recognition is performed. The mapping of bundles of features is performed using min-hashing. This hashing technique allows a compact representation of the features, while it does not imply a strict match to perform search. This allows to efficiently search for similar bundles of features present in the reference images. This approach currently holds the best known results for the Flickr32 dataset [49].

Although keypoint approaches were widely used in several tasks, such as brand detection, more recent works have been outperforming these by a wide margin, Girshick et al. [32] argues that keypoint based features for general object recognition are now bottlenecks in terms of improvement of recognition performance when compared with recent approaches.

The work from [15] explores Deep Neural Networks (DNNs) for the task of logo detection, in particular, Convolutional Neural Network (CNN) models. By reusing pre-trained networks with a large dataset, the networks are fine-tuned to recognize graphic logos, taking advantage of generic high level features previously learnt by the network. The authors use the full image (including background) to train and test the different network structures created by other authors. The results achieved are below the state-of-the-art results of [48], but this is one of the first works on graphic logo recognition using *deep learning* models.

2.2 Commercial solutions

There is a wide scope of commercial solutions that apply some kind of image recognition or matching related to brand recognition. Since technical details are usually scarce, their description will be based on available information.

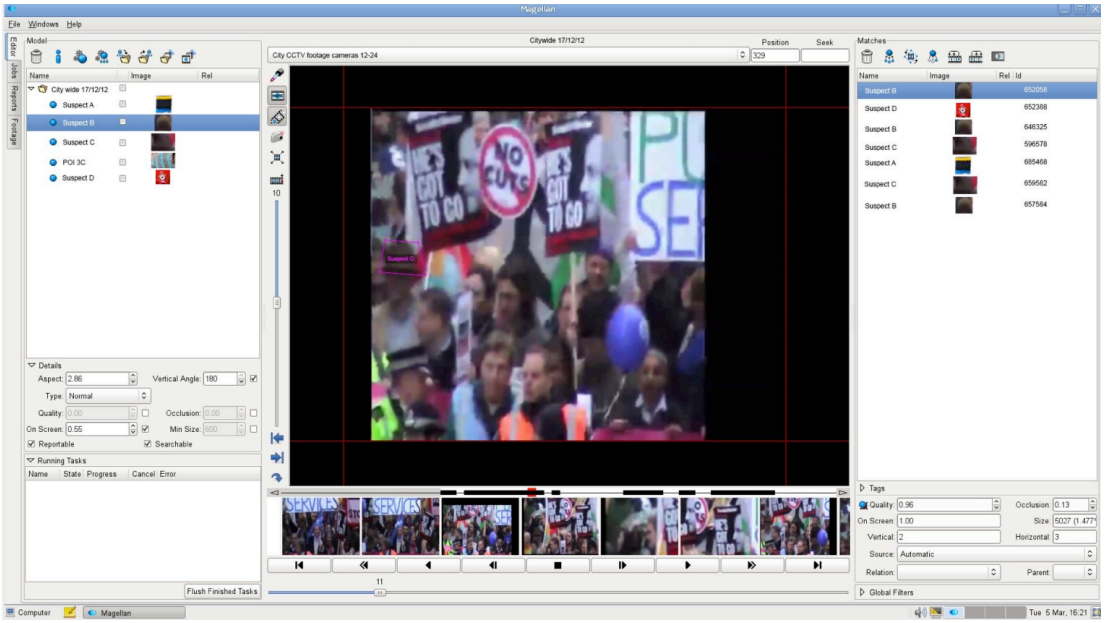


FIGURE 2.3: Omniperception Magellan from Digital Barriers

- Omniperception Magellan from Digital Barriers - Magellan is a in-video brand and logo recognition software, it focuses in the sports and forensic markets. It

offers object and brand detection in video and creates reporting the airtime of the recognized object, brand image or logo. In case of the sports market it allows to report the time a brand is displayed in a sports video (its purpose is similar to [13]). For the forensic market, analysis of Closed-circuit television (CCTV) footage to report on object or key suspect appearances, is listed as a possible use.

- Orpix Logo Recognition - This solution offers similar funcionality to Omniperception Magellan. It produces a report of detections of brand images or logos, duration of appearance and location, the listed purposes are television broadcast monitoring for brand images, product placement and brand protection and copyright infringement.

- Logograb - Logograb offers a product that allows new ways of marketing through logo recognition in a consumer mobile application. It allows brands to connect and offer marketing campains through their logo (i.e. augmented reality), the user uses their mobile phone camera to capture a brand image and its given access to further information or special campaigns. Brands then can receive analytics about specific campaign effectiveness. They also give access to their recognition system through a paid Application Programming Interface (API).

- ObjectVideo - This suite offers general recognition and tracking of objects, rule based analytics, event counting for business intelligence, forensic and video surveillance.

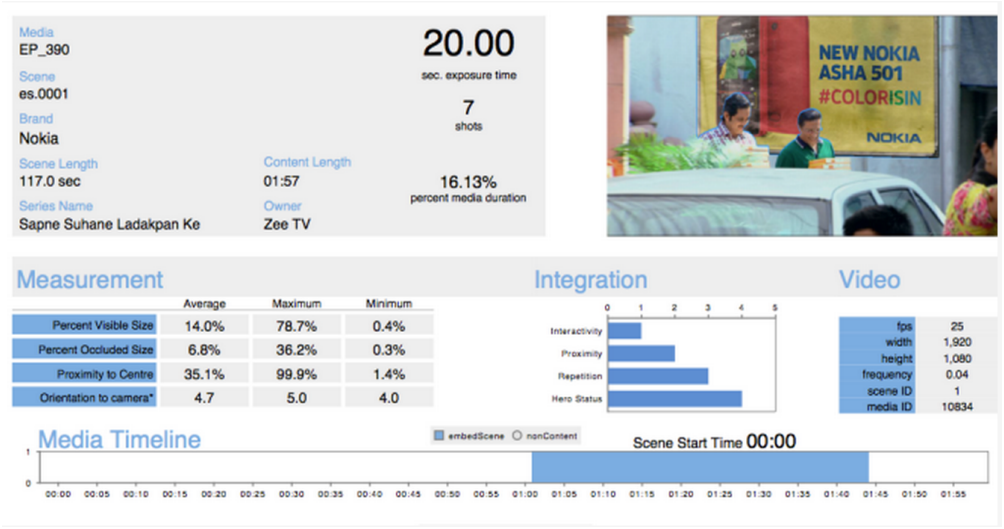


FIGURE 2.4: Mirriad web interface

- Mirriad - Mirriad offers a service of ad integration in video scenes using computer vision technology. It allows content creators and marketers to perform dynamic and seamless ad placement into videos and delivery of the modified content. It first

analyses seamless ad placement opportunities into the video and it allows the user to set the campaign parameters such as scheduling and specific demographics. The system does object and surface tracking and scene segmentation to find locations where ads can be inserted seamlessly.

- Amazon applications and family of websites - Amazon relies throughout their mobile applications and website on visual search and suggestions based on visual similarity of items. A9 a subsidiary company of Amazon provides and researches scalable image retrieval, feature compression, object detection and tracking solutions for their parent company.

Chapter 3

Feature Extraction in brand images

This chapter describes some of the most widely used feature extraction techniques used in images. Given the conditions that an object can appear in an image, scale and rotation invariance is one of the most important characteristic that a feature representation should have. We first delineate algorithms that details a part of the image or patch based on texture features. We describe the Scale-Invariant Feature Transform (SIFT) algorithm, and several other algorithms that were directly inspired by it. We also give a description of a simple method to encode shape and texture, called Histogram of oriented gradients (HOG). We finalize with techniques to extract color features from an image.

3.1 Keypoint extractors and descriptors

The keypoint recognition problem is often split into main components. The first is the detection of salient points or regions which characterize the object or image in mind. Based on the location and neighborhood of the found salient region in the previous component, the second component tries to extract a unique and discriminative signature of the region, often called a feature descriptor. These algorithms are widely used given their properties. Since objects in images can present themselves with a multitude of deformations and can be subject to varying lighting conditions, these algorithms try to make their feature descriptors highly distinctive, robust to lighting changes, noise and invariant to changes in position, scale and rotation.

3.1.1 Scale-Invariant Feature Transform

The Scale-Invariant Feature Transform (SIFT) algorithm is one of the most widely used local feature detector and descriptor. The final descriptor representation is said to be rotation and scale invariant and robust to some lighting changes.

The algorithm follows four steps in order to achieve scale and rotation invariance. We will now give an overview of these steps.

Scale-space extrema detection

First, a multiple scale analysis of the image is performed. It starts by doing edge detection using the Difference of Gaussians (DoG) method, an approximation of the Laplacian of Gaussian (LoG) filter. Multiple sets of pyramids are built at different scales of the image where, each one of these sets is called an octave. On each octave the image at scale is defined at different blur levels. Images at each octave are then to used to build a Difference of Gaussians, representing the spatial look of an image at different frequencies, using the differences between different levels of the Gaussian pyramid. This whole process allows for the search of keypoints at different scales.

For each level of the DoG pyramid, maxima and minima is searched in neighboring pixels, and then compared at the levels below it and above.

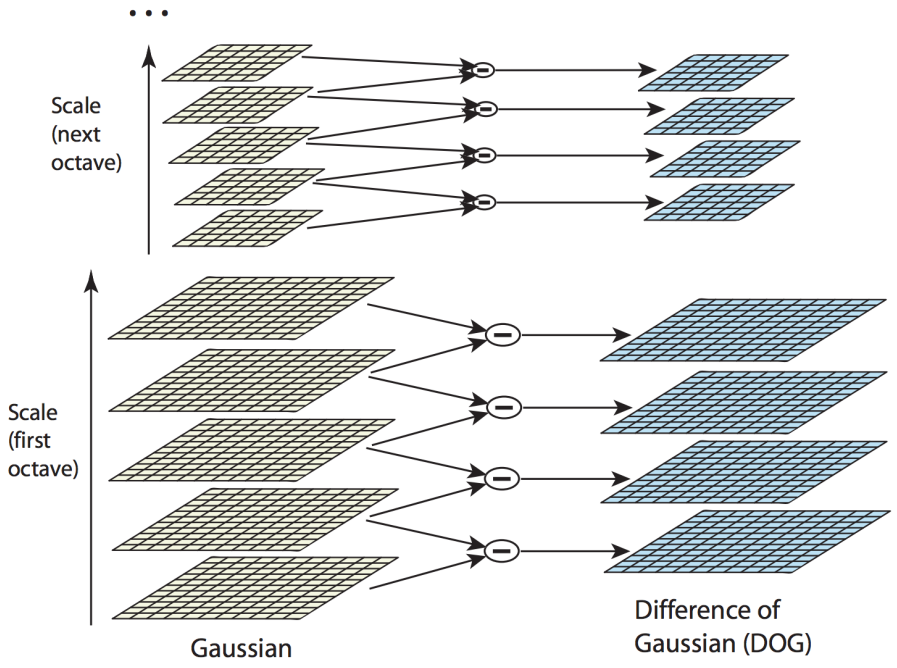


FIGURE 3.1: SIFT scale-space extrema detector schema [42].

Keypoint localization

Having potential keypoint locations, these results are first refined. Using a Taylor series expansion of scale space, a more accurate location of extrema is obtained. In this case low contrast regions (intensity at this extrema) are then rejected using a threshold value. Since the most valuable features are at corners, and DoG has high response on edges, edge responses are removed. What remains is strong interest points.

Orientation assignment

To achieve invariance to image rotation, gradient magnitude and orientation is calculated around each keypoint location at a certain scale. Gradient magnitude and directions are then put into an orientation histogram with 36 bins covering 360 degrees. Each sample added to the histogram is weighted by its gradient magnitude and a gaussian window that is dependent on the scale the keypoint is located at. The highest peak in the histogram is taken and any peak above 80% is also considered for a new keypoint. We now have the final representation of a keypoint with a set location, scale and orientation.

Keypoint descriptor

For the creation of the keypoint descriptor, around the location of the keypoint an 16x16 pixel window is defined. It is then split into 16 sub-blocks of 4x4 pixels. Given the gradient values of the pixels in this window, an 8 bin orientation histogram (45 degrees per bin) is created for each sub-block, gradient orientation is set according to the current keypoint orientation in order to achieve rotation invariance. By now we have an 4x4 array of histograms with 8 orientation bins, that is a 128 element feature vector for each keypoint. Finally the vector is normalized to reduce the effects of varying illumination changes using the vector norm.

3.1.2 Other variants

We now present some of the alternatives to the SIFT algorithm, given its properties some of them were directly inspired by it.

Speeded Up Robust Features

The Speeded Up Robust Features (SURF) algorithm is directly inspired by SIFT. For the feature detection, it uses a similar process to SIFT, instead of a DoG method to build image pyramids, in SURF, an integer approximation of the determinant of Hessian

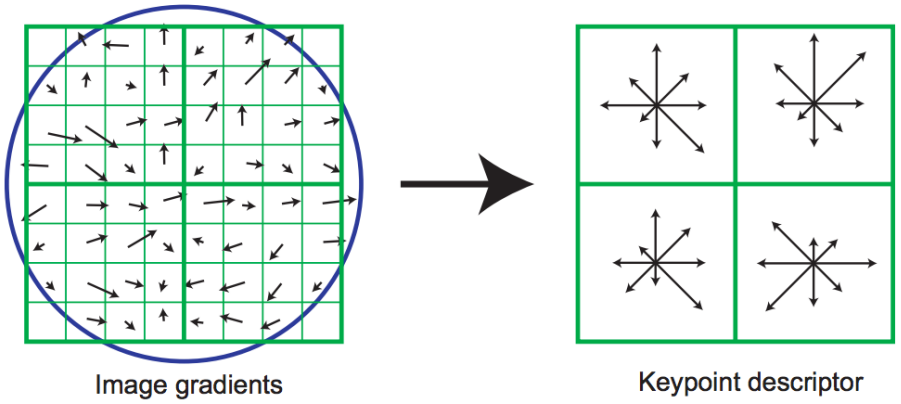


FIGURE 3.2: SIFT orientation histogram is constructed from specific regions determined by the detector [42].

blob detector is used. One of the reasons this algorithm is generally faster than SIFT is that, instead of building pyramids at different scales of the image, the scale of the image is unaltered and different scales of Gaussian masks are used. By not downsampling the image, time is saved.

For the feature descriptor, instead of a orientation histogram around the found keypoint, this algorithm, computes the sum of Haar wavelet response around the point of interest, in a circular neighborhood that is dependent of scale. The actual orientation of the descriptor is found using the area where the largest sum value was found in the previous step. To get the final descriptor feature vector, a square region around the point and rotated according to the orientation is created, the scale of this region is also dependent of scale. This region is then split into 4x4 square sub-regions, and for each sub-region, Haar wavelet responses are computed. Sum values in both x and y orientations are extracted, each sub-region has then a 4 dimension vector, giving the final descriptor representation of a 64 dimension vector.

Binary Robust Independent Elementary Features

Binary Robust Independent Elementary Features (BRIEF) [16] is a feature descriptor, which follows a binary format for its descriptor format, meaning that information in the patch is retrieved and encoded using binary comparisons between intensities differences between pixels. The binary format has major advantages in terms of comparison between descriptors. In fact, the distance between two descriptors can be efficiently computed using Hamming distance, in contrast to floating point descriptors like SIFT and SURF where the distance computation is more costly with Euclidean distance. [16] shows that this algorithm is robust to changes in illumination and also to blur and partial perspective distortions. However, it is sensitive to rotation deformations.

ORB

The ORB algorithm is presented as an "efficient alternative to SIFT or SURF", in [51] . Some of the disadvantages of SIFT and SURF rely on the fact that gradients of each pixel in the patch need to be computed, also, they are patent-protected, this algorithm tackles both problems. The ORB algorithm is built on top of other two, the Features from Accelerated Segment Test (FAST) algorithm [50] and the BRIEF descriptor [16]. After keypoint detection, a sampling step is performed, which uses 256 pairwise intensity comparisons. In ORB, the group of tests were extracted and built using an image dataset as well as machine learning techniques, while BRIEF uses an elaborate sampling pattern. The ORB algorithm overcomes the lack of rotation invariance of BRIEF by estimating local orientation, using the interest point location and the centroid of its neighborhood.

3.2 Histogram of oriented gradients

The Histogram of oriented gradients (HOG) feature descriptor although straightforward, is really popular for object detection. First order image gradients are computed, capturing contour and some texture information as well as reducing the effects of illumination variation, similarly to SIFT.

FIGURE 3.3: HOG visual representation.

The image is first divided into small regions, from all pixels in the region, gradient information is accumulated into an orientation histogram, covering a certain number of bins, and finally normalization is performed across the bins. Figure 3.3 shows the visualization of HOG features with the Ferrari logo.

3.3 Color Features

Color is an essential feature in images. It is often presented using the Red, Green and Blue (RGB) color space, since most computer screens use this model to perform color variations. The three RGB colors, are called primary colors; by varying their combinations in an additive way, other colors are obtained. Images can be represented in multiple other color models through linear or non-linear transformations. A reason to convert from RGB color space is to better track objects under changing color conditions. When an object is subject to varying light, the object color does not change, only the measured object luminance (indicates how bright a surface will appear) changes. Although widely used, the RGB model carries luminance information in all three channels, therefore a format where luminance is on a separate channel has advantages, since it can be easily removed, one example of such format is the HSL (Hue, Saturation, Luminance) format.

Often, color information is not accounted into local feature descriptors like SIFT where the description of color is hindered by the large amount of variations objects that an image can be subject to. Such variations cause the measured color values to vary significantly.

If the RGB components are normalized (Equation 3.1), objects are less subject to changes in surface orientation relative to the light source. Removing intensity variations, the luminance component is removed from the RGB model.

$$\begin{aligned} R' &= \frac{R}{R+G+B} \\ G' &= \frac{G}{R+G+B} \\ B' &= \frac{B}{R+G+B} \end{aligned} \tag{3.1}$$

Other color models present the luminance color separate by default, one such example is the YUV model, the Y component carries alone the luminance information, and the color information is contained in the U and V components.

3.3.1 Color Moments

This is one of the simplest and effective features respecting color. These features consist of mean, standard deviation and skewness. Their equations from (3.2) to (3.4) are:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij} \tag{3.2}$$

$$\sigma_i = \left(\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^2\right)^{\frac{1}{2}} \tag{3.3}$$

$$\gamma_i = \left(\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^3\right)^{\frac{1}{3}} \tag{3.4}$$

where f_{ij} is the j-*th* pixel color value of the i-*th* color component/channel of the image, N corresponds to the total number of pixels in the image. μ_i, σ_i, γ_i denote the mean, standard deviation and skewness of each color channel of the image, respectively.

3.3.2 Color Histogram

This method captures the frequency of occurrence of every color in the image through an histogram. Usually color information is captured across different color channels and then joined to form a single feature vector.

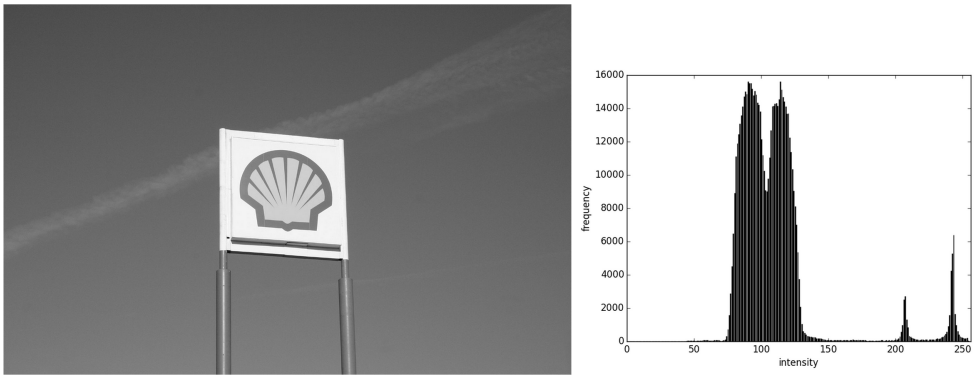


FIGURE 3.4: Color histogram of a grayscale image

Figure 3.4 shows the distribution of color values for a grayscale image that is used to describe the colors present in an image.

Chapter 4

Model approaches for brand detection

To reliably assess what brands are present in an image, not only we have to rely on low level features from the image, but also to build models that permit a higher level analysis. This chapter describes various models used for computer vision tasks that should be able to learn to detect brands. We start by describing various classical models followed by a description of more recent methods based on Deep Neural Networks (DNNs), and their respective building blocks and training process. We also describe transfer learning, a technique that allows the use of large networks, with a limited amount of data that proved to be essential for our work.

4.1 Earlier Models

In general, these models use low level features like SIFT and HOG features to build a higher level representation. Even if most recent methods are now outperforming these methods, some elements of these models remain important, since images still present constraints that these features and models were hand-designed for. We begin by describing a direct image matching method, the Nearest Neighbor Search (NNS), followed by an image classification method, the Bag of visual words and a further refinement of the model, the Fisher Vector (FV). Finally we describe the sliding window detection model, one of the primary ways of localizing an object in an image.

4.1.1 Nearest neighbor search

This method is often used in an information retrieval context. A database of images is built and indexed by their respective feature representation vector. To query the database for similar or duplicate images, first feature extraction is done to the query image and second, a search using the distance between the feature vector of the query and all images in the database is performed.

It is often used with features extracted with algorithms like SIFT. After keypoint detection and feature extraction is performed, the database is searched in order to find possible alignment of descriptors between the query image and a certain image in the database.

4.1.2 Bag of visual words

In [19] a generic visual categorization method is proposed. It is similar to the bag of words concept. Instead of words, categorization is based on a set vocabulary of image patches representations, extracted using interest points and formed using a clustering algorithm. Features vectors, from several images, are reduced to a few, using clustering. The resulting centroids, that distribute the feature space, are called visual words. The final representation formed by this model uses the visual words to quantize the keypoint features into bins. The bin distribution is used to achieve discrimination at category level. The goal of the clustering step is not necessarily to perform perfect clustering, but rather to achieve accurate categorization (see Csurka et al. [19]).

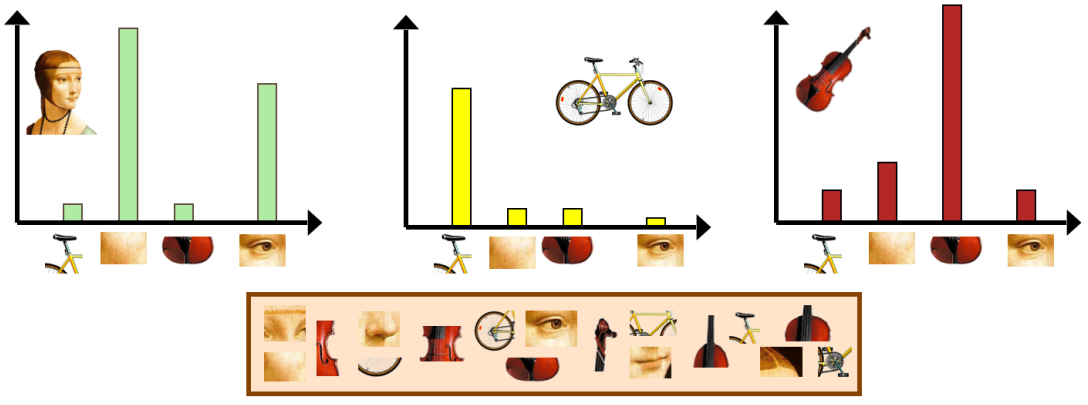


FIGURE 4.1: Bag of Words model high level representation. Taken from [1]

The final representation could be viewed as a histogram of present particular image patterns in the image. Figure 4.2 shows an high level schema of the model, with its dictionary and histogram representation. One major disadvantage of this approach

is that positional information of the keypoints is lost in the final representation of the feature vector, since the final description of the image is a single vector, with no positional information. Additional methods are often used to recover localization information, such as, the use of spatial pyramids, by performing matching at different regions, spatial information is added to this model. This approach aims to give a framework of generic visual categorization for a large number of classes.

4.1.3 Fisher Vector

The Fisher vector [44] representation presents itself as an advancement to the bag-of-words image representation. Both Fisher Vector (FV) and BoW are based on a visual vocabulary, given that patches are equally assigned to specific visual words in a dictionary. They differ on the way the dictionary is produced and on its assignments. FV is based on a Mixture of Gaussians for clustering of the patches representations while BoW generally uses K-Means clustering.

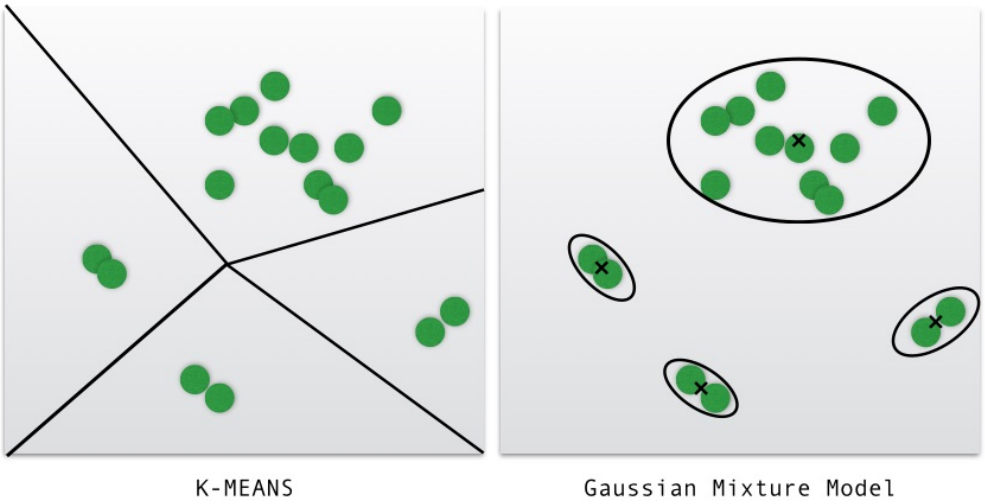


FIGURE 4.2: Representation of K-Means and Mixture of Gaussians feature space division.

The BoW representation does a direct assignment to a visual word while the FV takes into account the actual distribution in space of visual words within the cell using its mean and variance. Although this leads to a larger signature of a certain set of feature vectors, the quantized final vector of that set will be better describe by using its distribution within the pre-computed dictionary. The larger signature also allows this representation to perform well with linear classifiers.

4.1.4 Sliding window detection

This classical technique performs an exhaustive search by sliding a window throughout the image, classifying each window on how likely is an object present in it. HOG features are widely used in this sort of approach due to low computational effort required to compute them.

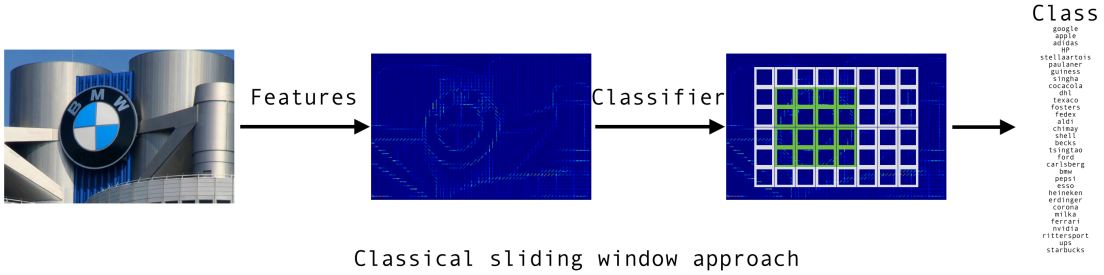


FIGURE 4.3: Sliding model overview.

The number of windows analyzed is dependent on the sampling density, determined by several parameters; therefore, there is a tradeoff between object coverage and speed. There are several parameters to take into account with this approach. The size of the window being analyzed and also its aspect ratio could be configured for a certain type of object. Since an object may appear at several positions and be scaled differently, translation and scale step size between the windows must be also taken into account. The authors of [21] shown this approach to be effective in the human detection task, and due to its simplicity its has been used as a starting point for other works.

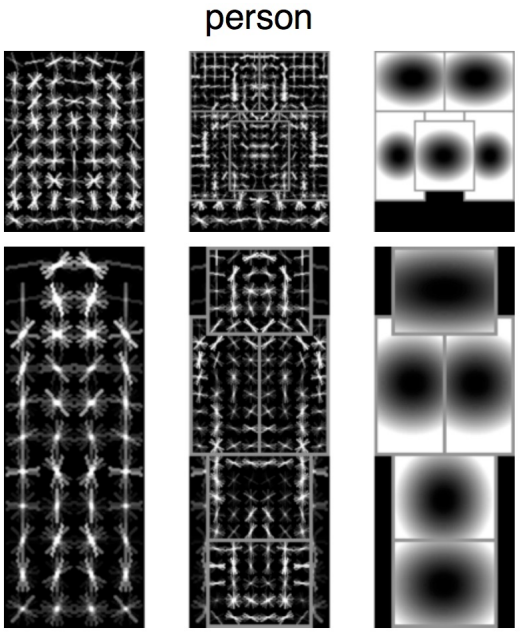


FIGURE 4.4: Deformable Part Model (DPM) person template. Taken from [27].

One example of such work is the Deformable Part Model (DPM) [27]. It is a complex variant of the sliding window method. The main contribution of that work is the concept of defining an object and detecting it by its parts. The model contains a number of part detectors (using HOG features), and learns constraints between each part. This method showed, at the time it was published, state-of-the-art results in the PASCAL VOC challenge and it is still to date used as a baseline, although other recent methods now surpassed its performance.

4.2 Deep Neural Network Models

Most of the improvements in object detection are associated with new object representations and machine learning models. A prominent example is the Deformable Part Model (DPM), using decompositions of objects it builds an higher level model that allows high-precision for a variety of object classes. Manually engineered representations and features have been among the best performing paradigms for computer vision tasks, but in the recent years, Deep Neural Networks (DNNs) have sucessfully emerged. Deep Neural Networks present major differences from traditional methods since they have the ability to learn complex models and robust object representations without the need for hand-designed models and features. The results obtained across thousands of classes in the ImageNet classification task [22] corroborate this fact.

4.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are biologically inspired variants of Multilayer perceptrons. From Hubel and Wiesel's early work on the cat's visual cortex [36], we know that the visual cortex contains a complex disposition of cells, called receptive fields, these are sensitive to small regions of the visual field. These simples cells act as filters that respond to edge-like patterns and form complex hierarchies that are invariant to the position of the pattern. Being the visual cortex the most powerful visual processing system known, its behavior has inspired many models, for instance, the Convolutional Neural Network (CNN).

The work by LeCun et al. [20] has been one of the main pioneering works for the current CNNs that are researched today. Recently CNN have been in the center of object recognition research. Partially due advances in training techniques and the emergence of parallel programming models that take advantage of Graphics Processing Unit (GPU) processing power.

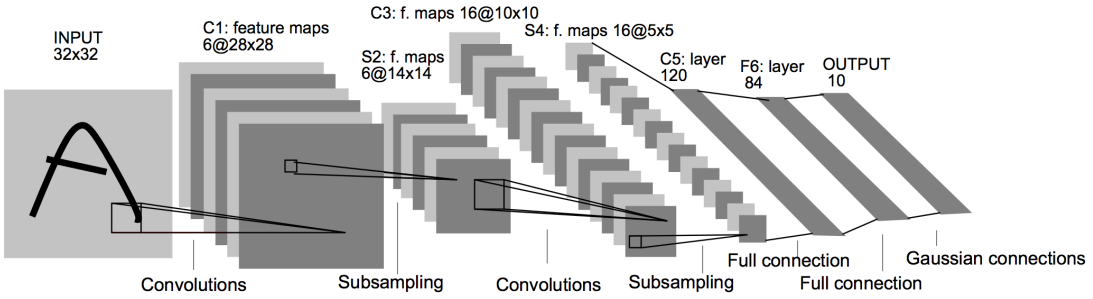


FIGURE 4.5: Architecture of LeNet-5. Taken from [40].

Figure 4.5 depicts the architecture of a CNN proposed by LeCun on the scope of the well-known handwritten character recognition problem for which it became famous.

A single CNN is composed by several types of layers. The main types of layers used to build a Convolutional Neural Network architecture are the convolutional layer, the pooling layer and the fully-connected layer. These networks work similarly to a regular artificial neural network but they differ on the assumption that the input will be an image (or other media signal). This assumption allows a CNN to encode particular properties into its architecture.

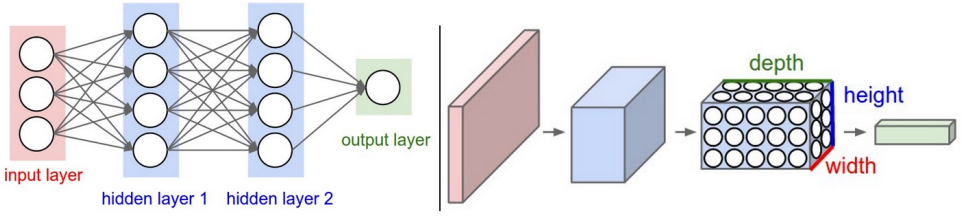


FIGURE 4.6: A regular three layer neural network when compared to a CNN. Taken from [4].

Every layer of a regular CNN architecture transforms an input three dimensional volume, e.g an image, with its red, green, blue components, to a three dimensional volume with a certain differentiable function that may (or may not) have parameters.

AlexNet [39] is usually attributed as the main work that popularized convolutional networks in the recent years, mostly due to its success in the 2012 ImageNet ImageNet Large Scale Visual Recognition Competition (ILSVRC) challenge [22], performing significantly better than its competitors. It presented a deeper but similar architecture to LeNet-5, with improvements to the layers, such as Dropout that would allow it to outperform past versions of these networks. Combining outputs of several models often lead to reduced test error, but due to the size of these networks, it is mostly prohibitive. Dropout is an efficient way of, inside a single model, producing what can be seen as several versions of the model itself. This aspect is achieved randomly dropping out outputs of several

hidden neurons in every back-propagation step. This way a single neuron cannot rely completely on its connection to others, being then forced to learn more robust features.

CNNs allow to apply a raw image to the input and to learn local feature extractors that are invariant to small distortions of the input. By implementing the principle of weight sharing, their generalization capacity is increased while reducing the number of free parameters. To avoid overfitting in extremely large networks, and train bigger models, powerful regularization techniques such as Dropout [54], the regularization methods in [30] and data augmentation [39] have been proposed. Using deep CNN networks, combined with large datasets has allowed to solve large scale computer vision problems in several areas, not only in visual recognition tasks, but also in speech recognition [53].

Donahue et al. [24] show that the high level features provided by the intermediate layers of these networks are useful and capable of outperforming previous methods, even in mutually exclusive datasets, making these models the primary candidate for object recognition and detection tasks [45].

GoogleLeNet [55] was the winner of ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014 and it is a massive network compared to AlexNet and LeNet-5. It uses many optimizations of previous works, such as Dropout. Its main contribution is the concept of "Inception" architecture, reducing significantly the parameters required. Even if these models are characterized by being large, its possible to propagate an image through the network extremely fast using GPU computing. Other methods have been also focusing on improving object detection speed in alternative ways as in [46]. Where a unified pipeline for object detection is defined. This method is capable of reaching real-time (45 Frames Per Second (fps)) detection with moderate accuracy.

While the most straightforward way of improving performance of deep networks is to increase their size and depth, there are still examples of works that take advantage of classical computer vision techniques with deep architectures, one such example is R-CNN, that will be explained further.

We will now describe, in particular, some basic building blocks for CNN networks, starting by the convolutional layer, pooling layer and fully-connecter layer. Finally, we describe Transfer Learning, which is a very recent methodology that allows to train a DNN with limited training data. Several details regarding the process of training of these models will be given.

4.2.1.1 Convolution Layer

The parameters of this layer consist of a set of learnable filters. In the forward pass, these filters are convolved (slided) across the width and height of the input volume. In more broad terms, we are computing the dot product between these filters and the input. The filters are small along the width and height dimensions, but to form the full output volume, the activation maps of all of these filters are stacked along the depth dimension. During training, the network will learn filters that have high response to specific patterns in the input volume.

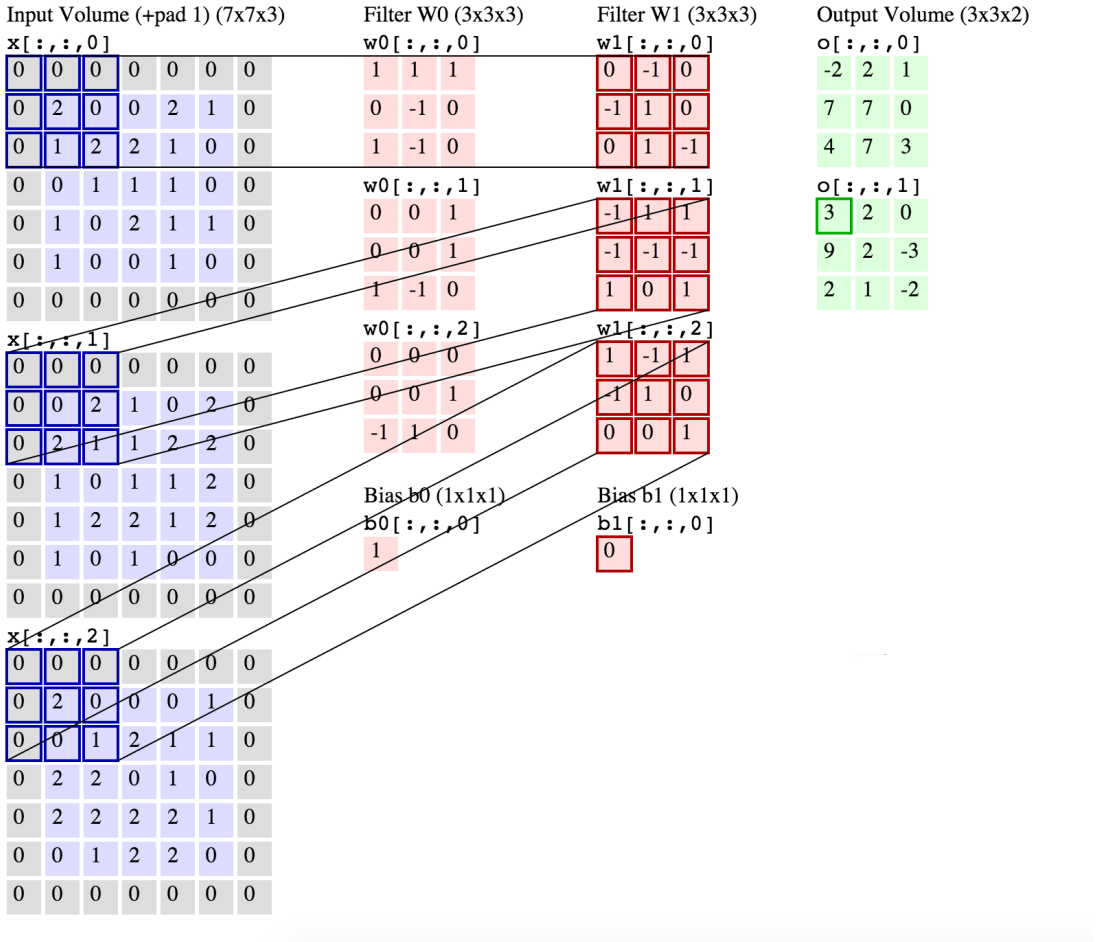


FIGURE 4.7: Convolution demonstration for an input volume of 5x5x3, with two filters of size 3x3 with a stride of 2. Taken from [4].

The number of parameters required for this layer are usually reduced by a parameter sharing scheme. As mentioned, the set of learnable filters are convolved with the input, and at that instant connected with a certain spatial region of the input. Without this scheme, parameters would have to be learned for each region of the input. This technique reduces dramatically the number of parameters required under the assumption that features for a certain region will be also useful for another region in another position.

Since each depth slice of the volume shares the same weights across the input, the number of required parameters to learn is significantly reduced.

4.2.1.2 Pooling Layer

The main purpose of this layer is to progressively down-sample the input, reducing the amount of parameters and computation required in subsequent layers, by doing so it also controls overfitting by the network.

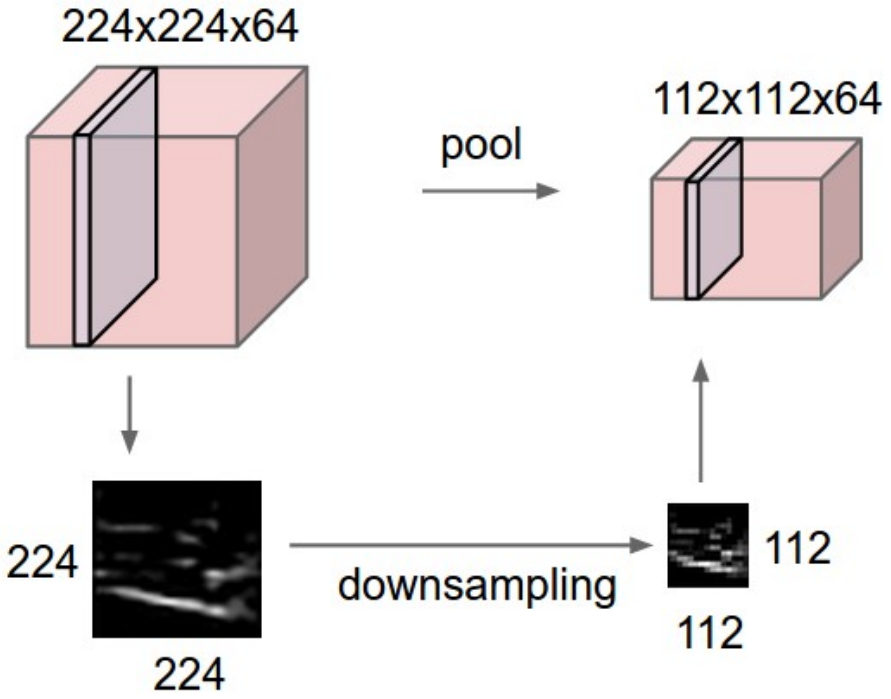


FIGURE 4.8: Pooling demonstration for an input volume of 224x244x64, with filter size 2 and a stride of 2 into output volume of size 112x112x64. Taken from [4].

The most common downsampling operation is max-pooling. It uses the max function to pool and down-sample input volume using the highest responses.

4.2.1.3 Fully-connected Layer

These layers are normally positioned after several convolutional and pooling layers to perform classification. They are fully connected to the previous layer activations in order to extract meaning and classify the input image, activations from the previous layer are weighted from learned parameters. A CNN, layer by layer learns more and more abstract concepts about the image and this layer responsibility is to transform those learned concepts into an interpretable result.

4.2.1.4 Transfer Learning

Training a CNN from scratch is a large endeavor, these networks are able to learn high level concepts but one major downside is that they require large amounts of data for training. AlexNet [39] was one of the first works that revealed to be successful, the dataset used for training was from the ImageNet Large Scale Visual Recognition Challenge, and the training data consisted of 1.2 million images distributed by 1000 classes. Modern CNN with this amount of data can also take weeks to train, so it is also a time consuming task.

It is now common for authors to release their pre-trained models to the community. There are several approaches that can take advantage of pre-trained models. A CNN can be used as a fixed feature extractor. Removing the last fully-connected layer would allow to access the intermediate activations from the network. Giving AlexNet [39] as an example, we would get a 4096 dimension vector, discarding the 1000 class scores that the fully-connected layer would give. With the 4096 dimension vector it would be possible to train a linear classifier for other tasks.

Fine-tuning a pre-trained network is another alternative. Previous layers of these networks learn to extract generic features like edges, and as we move into further layers the learned features get more and more specific towards the original dataset. It is then possible to continue training and repurpose, or transfer the weights so that they adapt to other datasets. The process takes advantage of the more generic learned features, given that, both datasets live in a similar domain, in this case, object detection in images. This method enables training a large network with a small dataset without overfitting. For this reason, during fine-tuning, a low training rate should be used, so the weights do not get distorted too quickly or too much.

In Yosinski et al. [61] this exact property is explored. The authors show that transferring features, even from tasks that are not closely similar, can be advantageous when compared to training from scratch.

4.2.1.5 Training

Training is usually performed using Stochastic gradient descent (SGD). While in Gradient Descent the training process runs through all the training samples, in SGD, only one or a few samples are used per iteration. The back-propagation algorithm, at each iteration of training, in the forward pass, computes the output and loss. In the backward pass, it computes gradients and updates parameters. This whole process, that is, a full pass through the training set, encapsulates what is usually called an epoch. Before

attaining convergence, the algorithm runs for many epochs. Next, we will describe some of the parameters used in these networks.

Learning rate (η)

During gradient descent, weight updates are multiplied with the learning rate denoted by η. Each layer might have a multiplier for this rate. During training this value is usually decreased over iterations t, making the learning rate a decreasing function η_t, such as a step function. This allows to guide the system towards optimal weights. We will denote the initial value of the learning rate as the base learning rate.

Step size and gamma(γ)

These two parameters control how the learning rate is decreased, gamma(γ) defines the degree of the change, while step size determine the interval of iterations to do so. Normally the transition between intermediate values of the learning rate are defined by a policy function, for example the step function.

Maximum iterations

Determines how many training iterations must complete before stopping training.

Momentum (μ)

Momentum helps the converge to a lower error by adding a fraction of the previous weight updates to the current, avoiding local minimum in training.

4.2.2 Region generation

Classification in a typical sliding window paradigm needs to be performed across an exhaustive list of positions and scales. In order to balance computation requirements and detection quality, there is the concept of "detection proposals" (sometimes called "region proposals" or "selective search"). Under the assumption that all objects share common traits that differentiate them from the background, using cues in the image, regions are selected, reducing significantly the number of windows that would be processed using a classical sliding window model. This approach has been proved efficient when paired with current classification methods, and still is a focus of research, since it has the capability of reducing the number of regions analyzed in the image as also augmenting

the effectiveness of classifiers further in the pipeline, directing them to regions that are likely to contain an object, reducing background clutter and false positives.

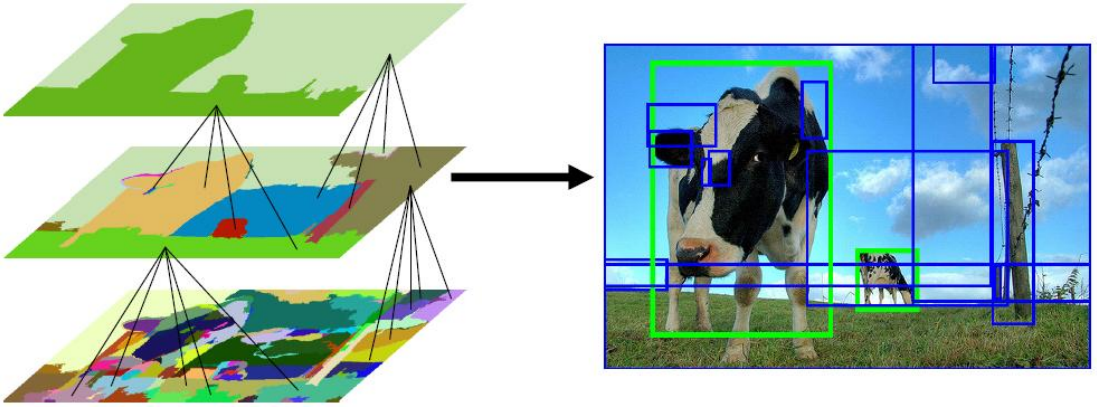


FIGURE 4.9: Selective Search system overview. Taken from [10].

Two of the earliest works, Objectness [12] and Selective Search [56] are the most well known methods. Objectness generates window proposals by choosing salient regions in the image, and then scores them according to several object cues, based on color and texture features, location and super-pixel straddling (measures if the region is completely contained in the window).

Selective Search uses segmentation as an instrument to select good locations for object recognition. It uses the segmentation algorithm proposed by [28], and then greedily merges segments using similarity features. The segmentation algorithm proposed by Felzenszwalb and Huttenlocher [28], captures important groups by first smoothing the image with a parameter σ and then using dissimilarity between neighboring pixels to build segments (called super-pixels). Parameter k influences the produced number of segments and respectively their size, the actual number and size of segments may vary depending on the local contrast of the image.

The generated segments are adapted using the segment features and greedily merged using the similarities between neighboring regions, the process is repeated until it creates a region that spans the whole image. In order to diversify the window proposals that are created by the method, several starting regions can be used. They are diversified by varying the parameter k of [28], or using several color spaces to compute similarity. The actual similarity measure used to merge regions takes into account, color, by using color histograms, texture, using SIFT features, size of the region, and fill, that measures how well a region fits into other (in order to first fill holes).

There are several works in this area, in [35] the authors analyze the pros and cons of several approaches. Selective Search has been broadly used in state-of-the-art models

such as the Region-CNN [33] due to being moderately fast while still producing quality windows.

4.2.3 Regions with CNN featuress

This method combines several approaches already mentioned. It is a state-of-the-art object detection system that combines bottom-up object region proposals and features learned by a CNN.

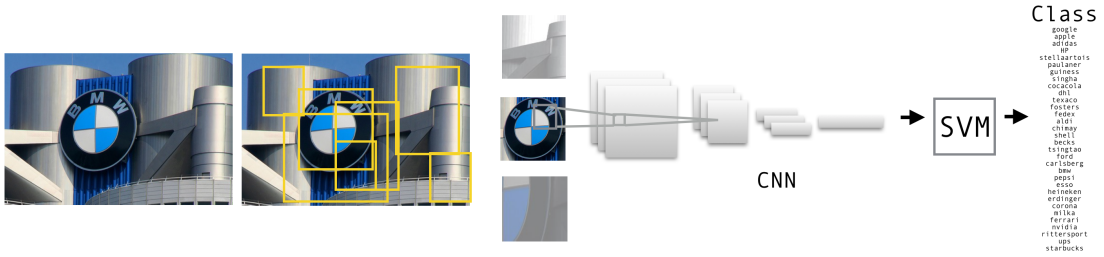


FIGURE 4.10: R-CNN model overview. Adapted from [32].

Taking the PASCAL VOC challenges as an example, previous top performers used methods like DPM and low-level features such as SIFT and HOG, until the resurgence of Deep Neural Networks (DNNs).

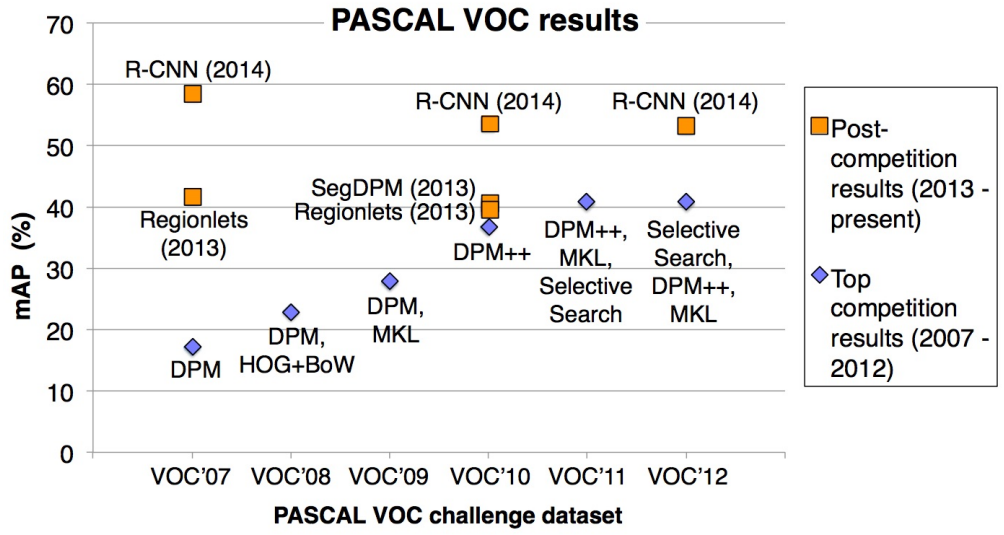


FIGURE 4.11: R-CNN PASCAL VOC results, taken from [8].

The R-CNN, at the time, outperforms all previous methods in several PASCAL VOC challenges (shown in Figure 4.11) and serves as a baseline for the ImageNet challenges, being outperformed by larger models or ensembles of methods on those challenges, such as GoogleLeNet.

At test time, using Selective Search [57] this model starts by generating around 2000 region proposals from an input image, warps the image to fit the input size of the CNN, computes a feature vector for each region using a CNN then classifies the region using per-class SVM classifiers and lastly, localization is improved using class-specific bounding box regressors.

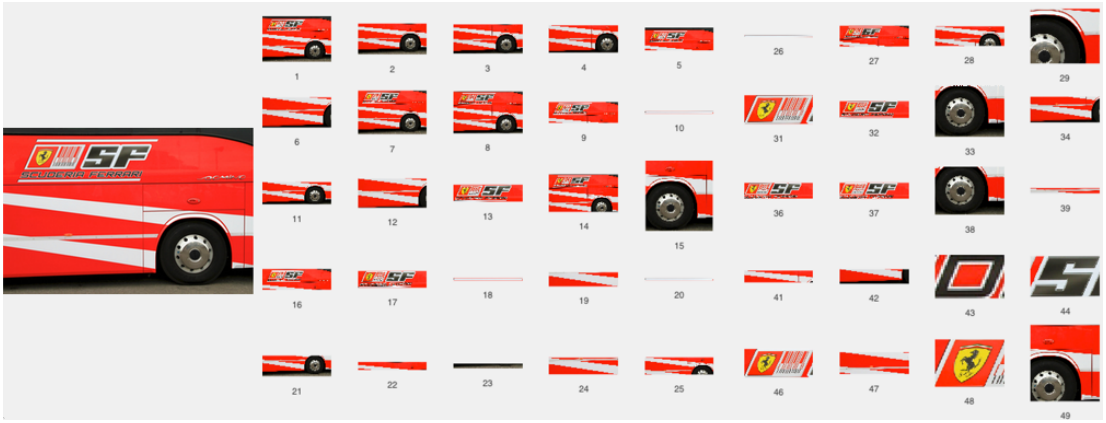


FIGURE 4.12: Examples of windows or regions generated by SelectiveSearch. First 75 of 683 regions.

Training takes a first auxiliary step, the CNN is pre-trained on a data-rich dataset, such as the ILSVRC 2012 dataset, consisting of 1.2 million images, each image objects annotated with bounding boxes. Then fine-tuning of the network to the smaller target dataset (PASCAL VOC) is performed. Using the image region proposals computed from the training data, and the fine-tuned CNN features, an SVM is trained per class, finally several class-specific bounding box regressors are trained in order to improve localization performance.

Present in the R-CNN supplementary materials [9] one can find the justification of more specific design decisions. It is also therein discussed the issue of speed in terms of training and testing. Finally it is argued that it would be possible to join all stages together, removing explicit SVM training while improving detection performance as well as training and test speed.

4.2.4 Fast R-CNN

Fast Region-based Convolutional Network (FRCN) addresses some of the drawbacks of R-CNN. Training in the R-CNN is a multi-stage process; first, the convolutional neural network must be fine-tuned, then SVM must be trained with the CNN features, and, finally bounding-box regressors are learned. Another issue is the storage required. Therefore three separate stages are needed, for the SVM training, each of the window proposals feature vector must be stored to disk, requiring large amounts of storage. At

test-time, a feature vector of each region must be also computed, making the detection time high.

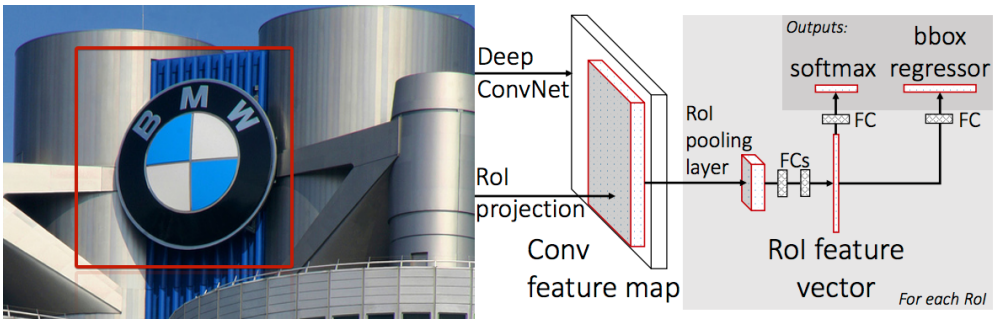


FIGURE 4.13: Fast R-CNN architecture overview. Each of the RoI are input to a CNN. Each RoI is pooled is the mapped from its feature map to a feature vector using fully-connected layers (FCs). The output from each RoI are softmax probabilities and bounding-box regression offsets. Adapted from [31].

FRCN fixes some of these issues by introducing a RoI pooling layer. It first computes feature maps for the whole image, then using that pooling layer builds feature maps specific for that RoI using the full feature maps. Its then possible to perform training in a single-stage, jointly training for classification and bounding-box regression. Each region is classified using a multi-class Softmax linear classifier [25], this classifier differs mostly from the SVM classifier from its loss function by replacing the hinge loss of SVM with cross entropy loss. The major advantage of using a Softmax classifier instead of a SVM classifier is that the results given by Softmax are immediately more interpretable, given that they are normalized confidence values for each class, instead of uncalibrated scores normally given by an SVM.

The authors report a speedup in terms of training of 9 times in comparison to the R-CNN and a test of speed per image of 0.3 seconds (excluding object proposal time).

This approach, as in the R-CNN still takes advantage of region proposal methods like Selective Search, in a recent work by the same authors, they propose a major improvement towards real time detection. In [47], the authors propose the Faster R-CNN, where a special type of network, the Region Proposal Network (RPN), is used to generate window proposals. By sharing convolutional features with the network responsible for detection it achieves a frame rate of 5 fps on a GPU, including all steps while achieving state-of-the-art performance in the PASCAL VOC.

Most current region proposal algorithms are slow, taking most of the detection time of these systems, taking the FRCN as an example, with region proposal time included, an image will take around 3 seconds (the number of proposals and time depends on the image). The authors achieve better results and speed with this system by using a lower

number, but higher quality set of region proposals generated with a RPN, in comparison with FRCN.

Chapter 5

Computational Experiments

With this chapter we aim to give a description of the process we went over to achieve our final model and quantitative results. We first describe the data used in the experiments, followed by several metrics further used to assess the quality of our results. Experiments with classical models are first described, followed by the experiments using Deep Neural Networks (DNNs), in particular the Fast Region-based Convolutional Network (FRCN). The chapter finishes with the discussion of the results, and the particular issues found in each model while experimenting.

5.1 Experimental Procedure

All the current experiments were performed using the Flick32 Logo Dataset[49]. Among the present existing datasets it contains more classes of logos, more images per class, and takes into account images with no visible logos. These characteristics align with our initial objectives, since not only it accounts for discrimination between graphic logos, it also takes into account real world environments, with or without logos.

The dataset contains images with 32 different graphic logos and a separate set of images with none of those 32 graphic logos present in the images.

Partition	Description	Images/class	#Images
Training set	Hand-picked images	10 per class	320 images
Validation set	Images showing at least a single logo under various views	30 per class	960 images
	Non-logo images	3000	3000
Test set	Images showing at least a single logo under various views	30 per class	960 images
	Non-logo images	3000	3000

TABLE 5.1: Flickr32 Logo Dataset[49] partitions/subsets. Taken from [5].

The dataset is annotated with image binary masks. Testing is performed in images, that do not only contain graphic logos, but also some amount of background noise, similar to those Figure 5.1 shows.

FIGURE 5.1: Example images from the *Carlsberg* class.

5.2 Model Evaluation

Metrics are required to evaluate the predictive capability of some model or classifier regarding classification. In this case the capability of detecting the presence of a graphic logo. The following metrics were chosen since they are used consistently throughout other solutions. Using same setup as other works, such as the same dataset and pre-conditions, we can then have a baseline for comparison. The following subsections contextualize these metrics to our current problem.

Precision and Recall

This paired criteria is one of the most used metric for classifier assessment.

Precision is a measurement of the proportion of results retrieved that are considered to be correct. Recall is a measurement of the proportion of all possible correct results that the classifier actually detects. Precision and recall generally vary inversely, that is, as precision increases recall generally decreases, and vice versa. Usually we search for the best balance between the two, and depending on the application and needs, it may be preferable to maintain an higher degree if one or the other. Both scores reach their best value at 1 and worst score at 0.

	Actual Positive	Actual Negative
Prediction outcome positive	TP	FP
Prediction outcome negative	FN	TN

TABLE 5.2: Confusion matrix. Classification systems use the data in this matrix to calculate different metrics.

TP - True Positive: Case was positive and predicted as positive.

FP - False Positive: Case was negative and predicted as positive.

FN - False Negative: Case was positive and predicted as negative.

TN - True Negative: Case was negative and predicted as negative.

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Table 5.2 shows the classification outcomes that are used to calculate precision and recall. Precision and recall are defined by Equation 5.1 and 5.2.

For a multi-class problem, precision and recall take a similar form. We will take classes A,B and C as an example. When computing precision for A, true positives share the same definition as the binary class formulation, where false positives are all A cases not classified as A. Recall also shares the same formulation, where false positives are all A cases not classified as A (A cases classified as B plus the A cases classified as C), equivalent to the total number of A cases minus the number of true positives.

Another common way to measure a model accuracy, is using the F_1 score metric shown in Equation 5.3. It considers both precision and recall to compute its score and indicates an overall utility measurement for the model or classifier.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.3}$$

Support

Support describes the number of occurrences(true instances) for each label or class.

Mean Average Precision

Where precision and recall are single values based on list of unranked predictions, Mean Average Precision (mAP) is often used in document retrieval solutions where the system returns a ranked list of relevant documents through a query. In our context, a document will be an image or a specific detection of a logo in an image. Since the retrieved list is ranked, it is desirable to consider the order that the returned documents are presented and give more relevance to correct items at the top.

To obtain the value of mAP, precision and recall values are then calculated at every position in the ranked list of documents for each query. Average precision is defined by Equation 5.4.

$$AveragePrecision = \sum_{k=1}^{N} Precision(k) \cdot \Delta Recall(k) \tag{5.4}$$

Where k is the rank in the list of retrieved documents, N the total number of retrieved documents for this query, $Precision(k)$ the precision value at the k point of the list and $\Delta Recall(k)$ the change in recall value since the last point in the list. Mean Average Precision is the mean value of average precision values obtained across all queries or classes. The Average Precision (AP) also summarizes the shape of the precision/recall curve built from the method's ranked output. Most of the recent computer vision works use it as a metric for classification and detection.

Intersection Over Union

This metric is often used to judge object detections in images using their bounding box overlap with the ground-truth object. Detections usually are considered correct when the area of overlap a_0 between the predicted bounding box and ground-truth exceeds 50% [26] using Equation 5.5.

$$a_0 = \frac{area\,(B_p \cap B_{gt})}{area\,(B_p \cup B_{gt})} \tag{5.5}$$

5.3 Earlier Models

In this section we describe the experiments using classical models, they were performed with the goal of classifying images according to graphic logo or brand present in an image. We started with a small experiment using Nearest Neighbor Search conjointly with SIFT features, followed by another using BoW and a further refinement using Fisher Vectors (FVs).

5.3.1 Nearest Neighbor matching

OpenCV is an established computer vision library, this library provides several utilities that we use throughout, from data pre-processing to actual detection systems. Using OpenCV, we started with a simple approach of matching keypoint based features, using a nearest neighbor approach, in this case using the SIFT and SURF descriptors. The process is based around keypoint extraction of the query image and then a Euclidean distance search through extracted keypoints in all training images. Random Sample Consensus (RANSAC) from Fischler and Bolles [29] is used to remove possible outliers from the found matches. The best match in terms of distance is used to determine the class of the query image. This method is adequate for searching exact matches between images, turning out to be extremely inflexible to changes in lighting conditions and high background noise, for that reason we found this approach to be unfeasible to our goals.

5.3.2 Categorization using Bag of Visual Words

Using the tools provided by OpenCV (version 2.4.9) we used its bag of visual words framework to produce results. The bag of visual words model works by collecting multiple visual words and then discretizing those words into a single global description. In this case the visual words are local feature vectors computed with keypoint descriptor algorithms such as SIFT, SURF and ORB that describe a local region of an image. In order to build a global description of the features present in the image, this collection of features must be first quantized into a vector. A dictionary of features that covers the universe of possible regions is used to describe an object. We infer the presence of an object, in this case a graphic logo, using the presence of specific local regions.

To obtain the visual dictionary the framework uses the K-Means algorithm to select representative local regions (from the feature vectors). The K-Means algorithm proceeds by assigning feature vectors to a specific cluster, and iteratively recomputing cluster centers. This means that the algorithm could converge to a local optima depending of the distribution of data and the initial positioning of cluster centers.

Keypoint based algorithms often provide an interface for feature extraction as well as detection, most of these algorithms rely on salient object detection based on peak responses across image scales. We take advantage of such algorithms to select regions for feature extraction in the image. We will discuss further this topic in Chapter 5.5.

For the final global description, the selected local region feature vectors, using distances to each element in the dictionary, are quantized into a histogram.

FIGURE 5.2: Keypoints detected in the image. Regions surrounding these points are selected for extraction, and then quantized.

To detect classes an SVM classifier is used, to accommodate the use of multiple classes a one-vs-all approach is used. The classifier takes part in several steps in order to predict the class of an unlabeled image. First regions are selected, their description is quantized using the dictionary and training of the SVM follows using the global image description, testing follows a similar process.

The number of clusters that make up the dictionary are a major contributing factor to the system capabilities, since they directly influence how the feature space is partitioned and inter-class discrimination is performed.

Fisher Vectors

We also experimented with Fisher Vector (FV) to perform feature quantization. The system follows similar steps to the BoW approach, only differing in the quantization method, instead of histograms to quantize all selected regions feature vectors Fisher

Vectors are used. We used the Vlfeat [11] computer vision library to compute the FV representation. Only the quantization step differs from the previous experiment, so a similar setup of region selection and classification is used.

5.4 Deep Neural Network Models

In this section we describe details of our experiments with the FRCN model. The experiments were performed with the goal of detecting and loosely localizing graphic logos in images.

5.4.1 Fast Region-based Convolutional Network

For our experiments, we take advantage of transfer learning and apply it for our dataset. As already mentioned, the initial step is pre-training a CNN network with a large scale image dataset, we manage to skip this step since there are several pre-trained models available. The original models and source code for R-CNN and FRCN is available in the respective authors page [7]. R-CNN and FRCN were built using the Caffe deep learning framework [2]. The original projects work upon three CNN models. We only managed to experiment with the smaller and medium size models since the larger model requires a GPU with high amounts of memory we did not have access to. Training was performed with an Amazon S2 g2.2xlarge virtual server instance with 15 GB of memory, Intel Xeon E5-2670 processor and a Nvidia Grid GPU(Kepler GK104) with 4GB of video memory.

The available implementation of FRCN is built on top of three CNNs models. The three CNNs are categorized in the original work by their size, being Caffenet, VGG_CNN_M_1024 and VGG16, small, medium and large. Caffenet structure follows the original AlexNet [39] model with minor variations, VGG_CNN_M_1024 originates from the work of Chatfield et al. [18], and VGG16 from [62]. All these models and respective pre-trained weight files are available in the Caffe Model Zoo website [3]. Figure 5.3 portrays the three variations of the model. As mentioned in the Chapter 4 the FRCN model mostly adds a RoI pooling layer towards the end of the network models.

We experimented with several learning rates for the two variations. Since we started with pre-trained networks for other dataset, as already mentioned, we had to use low learning rates. Using a learning rate value considered to be high (0.01), made both networks completely overfit and unable to cope with the test dataset. A learning rate ten times smaller of 0.001 is usually used for fine-tuning, since it updates the network weights slowly not distorting what the network has already learnt keeping overfitting low.

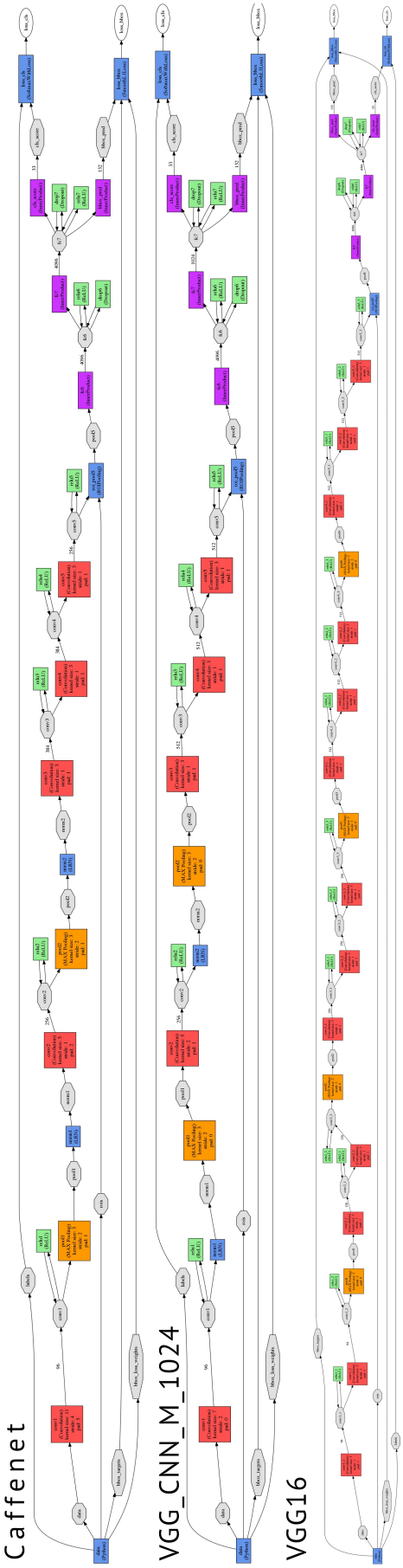


FIGURE 5.3: Network structure scheme of the three variations of FRCN, produced with Caffe draw_net utility.

The images used for training were collected from the training and validation set of our dataset. We use the full test set of the dataset for our tests, that is, 30 images per class, plus the no-logo class that contains 3000 images. Training images are sampled at each iteration, in FRCN, SGD is performed by mini-batches, each mini-batch has size 128 and is built by first sampling 2 images from the training dataset, then sampling 64 RoIs from each. Those 64 RoI are sampled from the object proposal windows according to their overlap with the ground-truth object, in particular their Intersection Over Union (IoU) value, 16 regions will have IoU > 0.5 with a graphic logo, therefore, considered positive, the remaining will contain background since they have lower IoU with the groundtruth object (a graphic logo).

Sampling is performed from regions that were pre-computed using Selective Search, Figure 5.4 shows examples of some of those regions. During the training process, each sampled region has a probability of 50% of being flipped horizontally, as a form of data augmentation.

Selective Search Mode	Minimum size (ks)	Sigma (σ)	Minimum size of region
Fast	50,100	0.8	20 pixels
Quality	50,100,150,300	0.8	20 pixels

TABLE 5.3: Selective search fast and quality mode parameters.

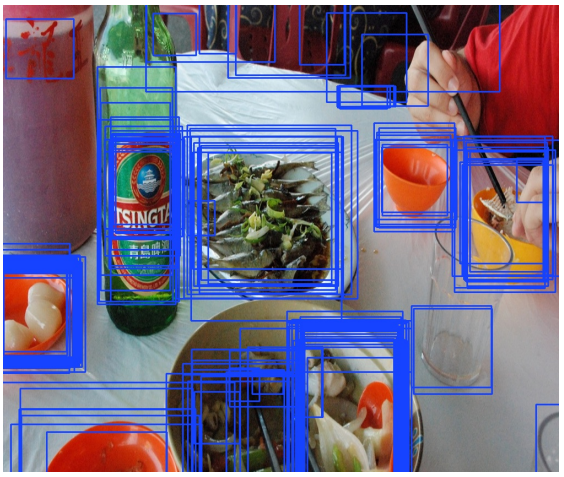
Table 5.3 shows the parameters of two variants of Selective Search, fast mode is faster but generates less window proposals, the number of window proposals for quality mode is slower since it segments the image with more criteria. All regions with less than 20 pixels are ignored. The fast version takes on average 3 seconds to generate object proposals per image, the quality version takes 15 seconds on average. For our train image set, the fast version generates on average 2132 regions per image and the quality version, 11132 regions.

Since the original dataset gives ground-truth annotations in the form of a binary mask we first translate it to bounding box annotations.

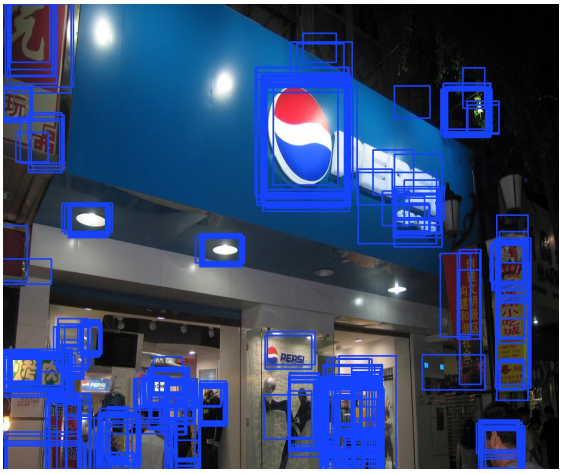
Figure 5.4 shows examples of regions that might generated using Selective Search.

The image patches defined by the RoIs, before classification, are first normalized using the image mean RGB values from the original training set of the network. The input to the CNN structure is fixed, therefore, the regions are first scaled to the correct size according to the network, Caffenet has input size 227 x 227 x 3 (color image), VGG_CNN_M_1024 and VGG16, have input size of 224 x 224 x 3.

We used a modified version of the FRCN that supports different datasets other than PASCAL VOC, it is available at [6]. We had to perform further modifications to support



(A) Tsing Tao



(B) Pepsi

FIGURE 5.4: Examples of RoI generated by SelectiveSearch (test set).

our dataset, in particular, change several network layers to support 32 classes and a special background class.

Model/Network	Base learning rate	Learning rate policy	Gamma	Step Size	Momentum	Weight Decay
CaffeNet_FRCN	0.001	step	0.1	30000	0.9	0.0005
VGG_M_FRCN	0.001	step	0.1	30000	0.9	0.0005

TABLE 5.4: Main parameters for the FRCN model.

We experimented training with up to 80000 mini-batch iterations, setting a base learning rate and lowering the value by a factor of 10 each 30000 iterations, similar to the original author experiment. Since the CNN weights were trained with, and are suitable to the ImageNet dataset (1000 classes), our training, will have to fine-tune the network weights to our 32 classes, similar to the original R-CNN where those networks were fine-tuned to the PASCAL VOC 20 classes dataset. Jointly, since FRCN is a single pass model, a softmax classifier for all our 32 classes is adapted during the fine-tuning of the CNN section. As mentioned, in an effort to significantly augment the data for training, we

joined the training and validation set of the Flickr32 graphic logo dataset. Training is then performed with a set of 40 images per class (10 from the training set, 30 from the validation set), making the total of 1280 images.

5.5 Results and Discussion

This section illustrates the results obtained with each approach and discusses some of the issues found in each. We start by discussing results obtained with classical models and their inherent flaws. Finally we illustrate in detail the results obtained using a DNN approach.

5.5.1 Earlier Models

We started by producing preliminary results using the BoW model, feature descriptor algorithms and one-vs-all SVM classifiers. We managed to achieve good recognition results using a custom split of the dataset (70% for training, 30% for test) for a reduced number of classes.

	precision	recall	f1-score	support
guiness	0.91	0.83	0.87	12
becks	1.00	1.00	1.00	12
tsingtao	0.34	1.00	0.51	12
paulaner	0.89	0.67	0.76	12
milka	0.67	0.50	0.57	12
carlsberg	0.58	0.92	0.71	12
ferrari	1.00	0.75	0.86	12
bmw	1.00	0.58	0.74	12
ford	1.00	0.50	0.67	12
cocacola	1.00	0.42	0.59	12
starbucks	1.00	0.83	0.91	12
avg / total	0.85	0.73	0.75	132

TABLE 5.5: Precision and recall values for 11 classes using the bag of visual words model. Dictionary generated with KMeans, size: 5000, ORB algorithm as a feature detector and SURF as feature descriptor. Multi-class SVM with a RBF kernel used for classification.

Table 5.5 shows the best results we managed to achieve using the BoW method. In an effort to improve the previous results, we also experimented with Fisher Vectors (FVs). According to the literature, this approach should improve the results significantly, since it takes into account the actual distribution and position of visual words in the feature space.

	precision	recall	f1-score	support
guiness	0.87	0.87	0.87	12
becks	1.00	0.69	0.82	12
tsingtao	0.53	0.60	0.56	12
paulaner	0.86	0.83	0.84	12
milka	0.41	0.45	0.43	12
carlsberg	0.46	0.73	0.56	12
ferrari	0.79	0.85	0.81	12
bmw	0.94	1.00	0.97	12
ford	1.00	0.50	0.67	12
cocacola	0.92	0.73	0.81	12
starbucks	0.96	0.92	0.94	12
avg/total	0.79	0.75	0.76	132

TABLE 5.6: Precision and recall values for 11 classes using the bag of visual words model and Fisher Vectors. Dictionary size 256, ORB algorithm as a feature detector and SURF as feature descriptor. Multi-class SVM with a linear kernel used for classification.

As shown in Table 5.6 we only manage to achieve marginal improvements using the FV method. This is attributed to the method we are using to select regions to perform feature extraction.

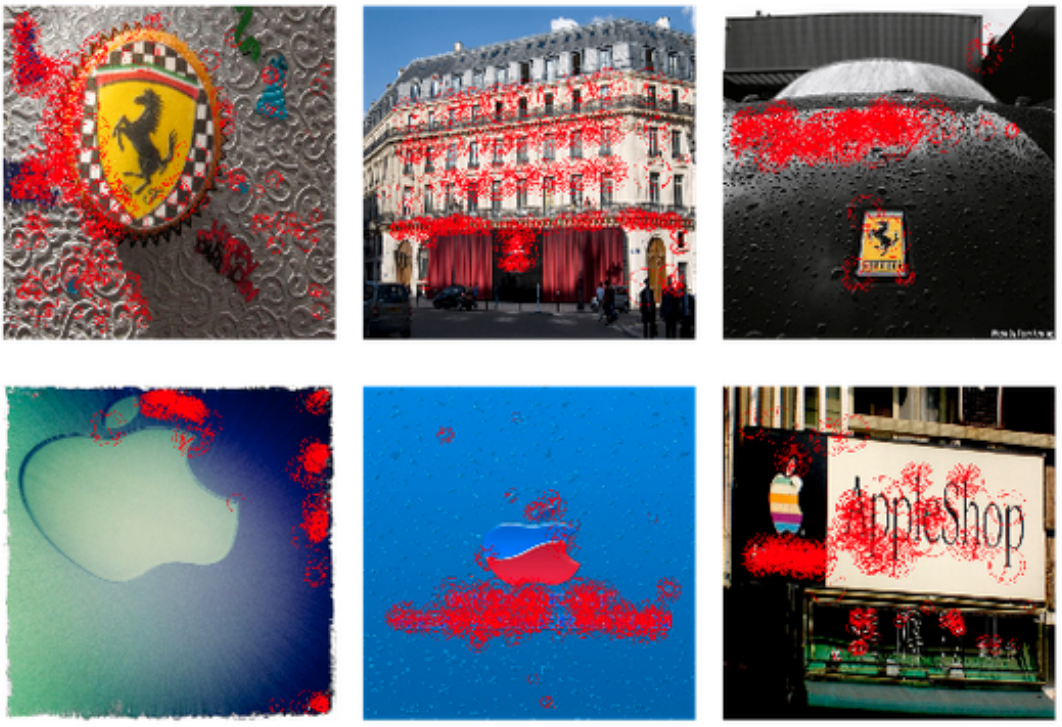


FIGURE 5.5: Regions for feature extraction selected by ORB in red. Note that most of the selected regions are background regions.

We were using the intuition that graphic logos, are normally high contrast relative to their surroundings and considered to be in salient regions. We verified that salient region

detectors such as keypoint detector algorithms (by themselves) are not adequate for the task. Since graphic logos are not always in salient regions, and sometimes themselves are not considered salient when they are flat in terms of texture, one example of such is the Apple logo. The characteristics of the dataset we used might have lead us to this fact, we started working only with images with logos, and since the dataset was built with image retrieval in mind, most of the images contain a graphic logo in a prominent regions. In a real world scenario, even if the dataset somewhat covers this situation, the graphic logo would not be the main focus, and would be blended even more into the background and be present in smaller sizes.

The model ends up selecting high amounts of background regions and few regions with graphic logos. With this approach we were trying to cope with the problem of background noise by only selecting salient regions of the image. However this lead to ignoring most of the regions with graphic logos. Since within our objective was to detect brands regardless of conditions, we then opted to pursue a more specific solution, that would deal directly with background noise.

5.5.2 Deep Neural Network Models

In this section we first discuss the quality of detections from the proposed model using mAP values, followed by an analysis using detection and recognition F1-scores not only in images with graphic logos, but also in images with no logos.

Convolutional Neural Networks features are now considered the primary candidate for visual recognition tasks [45]. The results we managed to obtain, and described in this section, seem corroborate that fact, moreover if the we compare them to our previous efforts.

A major concern from beginning, was the amount of data we had available. If we joined the training set and validation set of the dataset, we had 40 images of each class for training. Training a deep network seemed an impossible task if we take into account the dimension of datasets normally used for deep learning, such as ImageNet. We used transfer learning to avoid having to build a new dataset for specifically for this task.

The flexibility given by the original FRCN model, is part of the reason why we chose to further experiment with it. Graphic logos appear in small regions in the image and since the first step is to generate object proposals, the model gives a intuitive and modern way to solve the issue. The model also deals directly with background noise by using a special background class and also object proposals to reduce the number of regions that are processed in the image. This model is able to detect and localize multiple instances

of a graphic logo. As already mentioned, that functionality is helpful in situations where contextual advertisement insertion is needed.

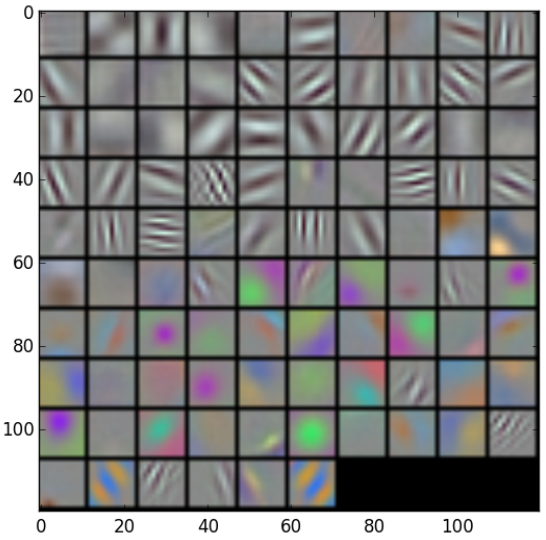
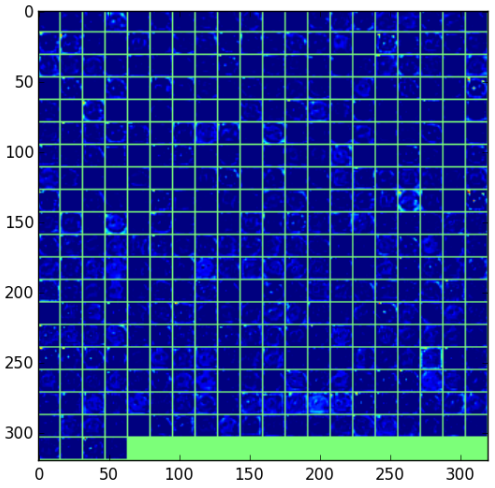


FIGURE 5.6: Filters learnt by the Caffenet_FRCN model in the first convolutional layer.

The Figure 5.6 is a visualization of the first set of filters learnt by the model, as expected, the first convolutional layer learns low level features such as edges and colors.

(A) Input image



(B) Output responses

FIGURE 5.7: Output responses from the fourth convolutional layer.

Further layers of the model learn several high level concepts, we can observe in Figure 5.7 several responses to round sections of the input image.

We will first evaluate the performance of the model in a detection context, using only images that contain a graphic logo.

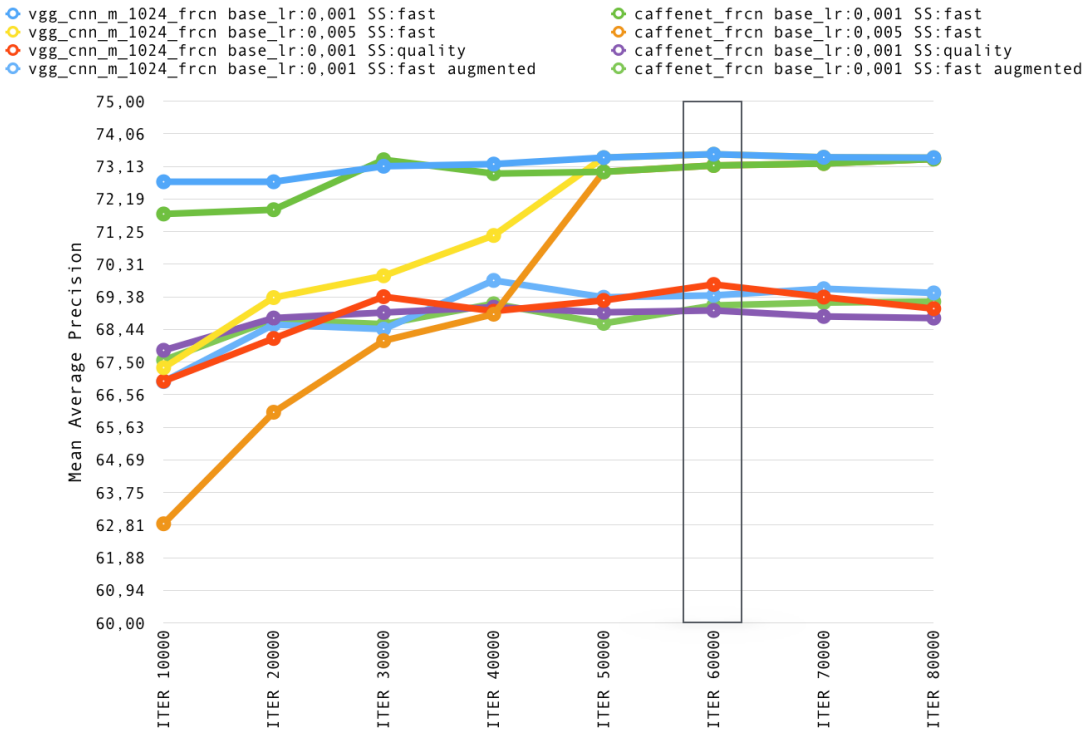


FIGURE 5.8: mAP values across iterations with each network and configurations.

In Figure 5.8 we compare the test set mAP values during the fine-tuning process for the two networks and several configurations.

Method	Base learning rate	Selective Search	mAP
BoW (tf-idf-sqrt, vocabulary-1000)[48]			0.545
Bundle Min Hashing (1p-wgc-ransac) [48]			0.568
VGG_CNN_M_1024_FRCN	0.001	Fast	0.7347
Caffenet_FRCN	0.001	Fast	0.7314
VGG_CNN_M_1024_FRCN	0.005	Fast	0.7347
Caffenet_FRCN	0.005	Fast	0.7314
VGG_CNN_M_1024_FRCN	0.001	Quality	0.6972
Caffenet_FRCN	0.001	Quality	0.6898
VGG_CNN_M_1024_FRCN - augmented dataset	0.001	Fast	0.6941
Caffenet_FRCN - augmented dataset	0.001	Fast	0.6912

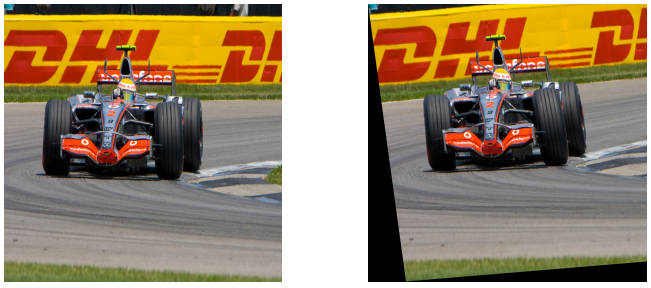
TABLE 5.7: Comparison of obtained mAP values with the work of Romberg and Lienhart [48]. Results obtained at 60000 iterations.

A system will only achieve high mAP if both precision and recall are high across detections. Table 5.7 illustrates and compares results from previous works. The choice to measure test performance across iterations is mostly due to the fact that we lost validation data by joining the training and validation set for training.

We can verify that having a higher base learning rate of 0.005 is not beneficial for learning, since both models are overfitting at 10000 iterations, whereas starting with a learning rate of 0.001 achieves higher testing performance from the beginning.

It can be observed that using quality version variant of Selective Search, that is, using a larger more diverse set regions, does not lead necessarily to improved performance. The decreased performance can be attributed to an higher number of windows that do not contain a graphic logo that need to be evaluated.

Additionally to the data augmentation performed by randomly flipping each region, we performed a small experiment by randomly distorting each image with a shear and slightly shifting color values on the image. We can observe a higher average precision for some classes, such as the Google, Guiness and Bmw, but this fact is not reflected across all classes. In practice we doubled the number of images on our training set, by saving each variation. Further study on this aspect is required, saving each image variation is not feasible due to disk space constraints, so implementing a module that does so during training is required for further conclusions. Figure 5.9 shows and example of such produced variation.



(A) Original (B) Variation

FIGURE 5.9: Example of image variations produced.

Around 60000 iterations is where we achieved the top mAP value of 73.4733% with the VGG_CNN_M_1024_FRCN model using the Selective Search fast configuration, so we will keep this value as reference in this section.

In Figure 5.10 we verify some trends across models. As expected, more flat and simplistic graphic logos present lower AP than those more elaborate. Such examples are Pepsi, Nvidia and Milka. These logos are considered more simple since they present less information and blend more into backgrounds, making them more difficult to detect. Figures 5.11 and 5.12 depict the differences between classes that achieve high and lowAP values through a precision-recall plot.

Since this model processes multiple regions in each image (depending on the image and Selective Search parameters, around 2000 regions), after getting a confidence value for each class, overlapping detections are filtered with non-maxima suppression. This system is able to cope with multiple detections in the same image even if the dataset used was not created for this purpose (one class per image).

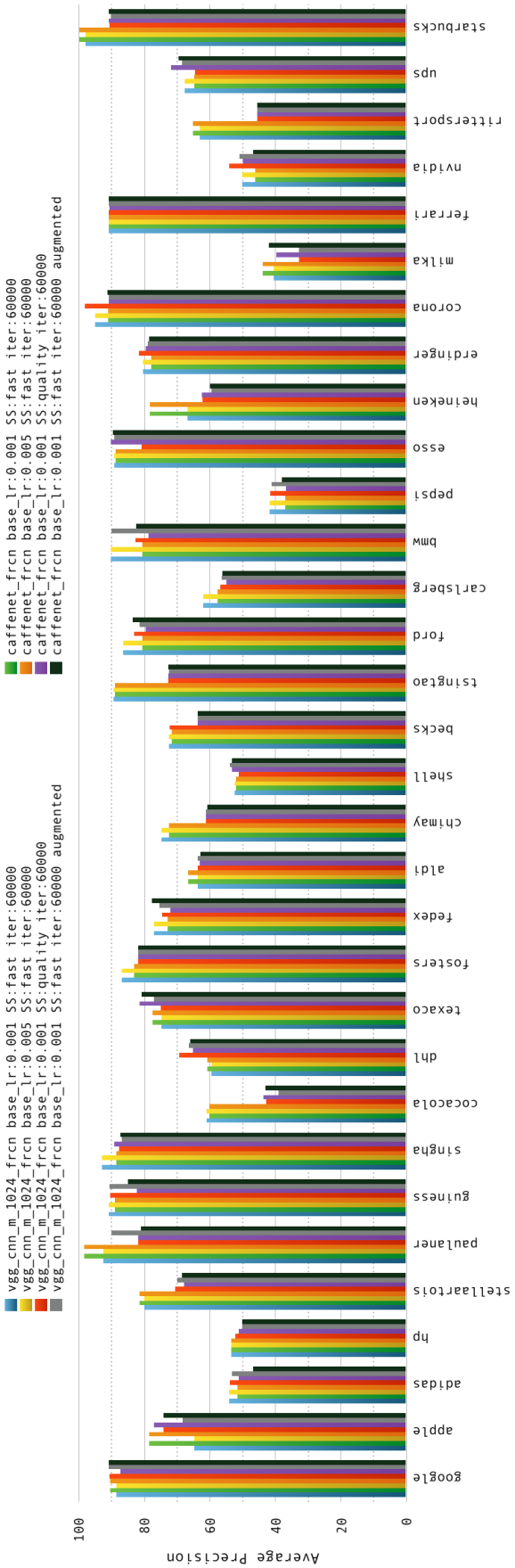
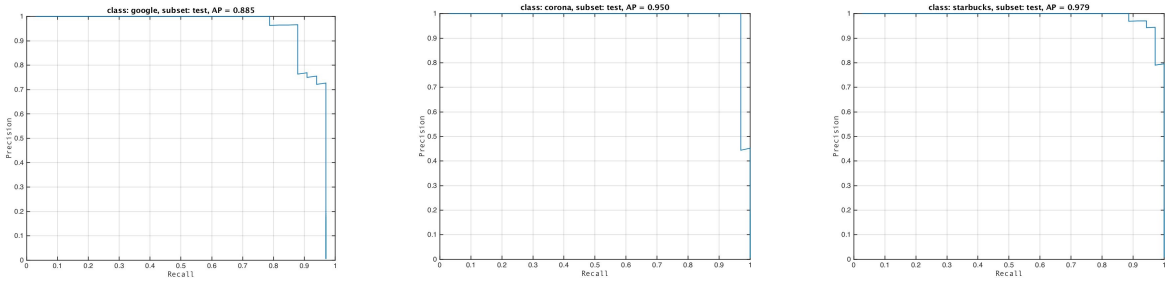
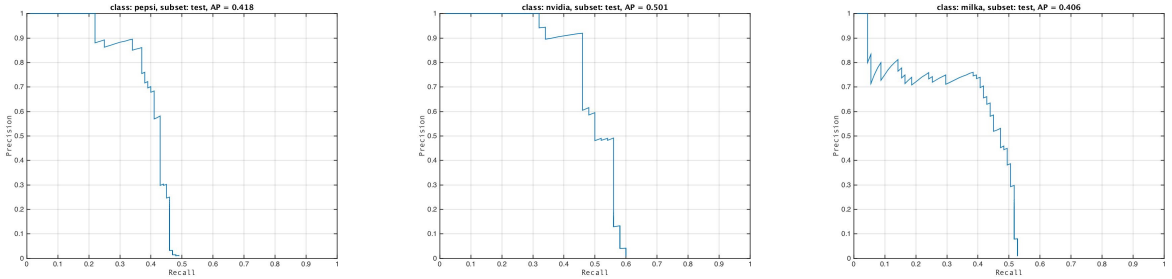


FIGURE 5.10: AP values for each class at 60000 iterations.



(A) Google (B) Corona (C) Starbucks

FIGURE 5.11: Precision-Recall for some of the classes with high AP.



(A) Pepsi (B) Nvidia (C) Milka

FIGURE 5.12: Precision-Recall for some of the problematic classes.

Since we could have multiple detections across the image, we will select the top five scored windows across all regions and classes for a clearer visualization.

(A) Google (B) Corona (C) Starbucks

FIGURE 5.13: Examples of correct detections.

Figure 5.13 shows the detections and respective confidence values for some of the classes with high AP. Even if some elements of the background are classified as logos incorrectly, the model tends to score correct regions even higher.

Since for most of the training images the graphic logos are in a visible position, directly facing the camera, the model has difficulties dealing with test images where those conditions do not apply, as shown in Figure 5.14. The ability to localize the logo is not always perfect, specially when its presented in more slanted positions, or where a major occlusion of the logo exists.

(A) Ferrari (B) Ups (C) Pepsi

FIGURE 5.14: Graphic logos in slanted angles or partially occluded are not always correctly detected.

Some of the graphic logos appear with changed colors for specific reasons, one example scenario could be a promotion where the graphic logos color is changed. Apple and Adidas logos are usually white, but in specific images appear with different colors, since the model was trained to detect specific colors. Obvious detections are sometimes missed. Additional training data for this purpose could alleviate this issue.

The dataset used, out of 32 classes, 11 classes are associated with beer and their logos are mostly present within bottles and glasses. From observing detections in a single image, that contain a beer or glass with its respective brand, it can be observed that mostly classes related to beer brands are predicted. These logos end up being really similar, with a specific format and all of them are present in similar backgrounds, making them more difficult to distinguish.

As a side note, the model was able to find some inconsistencies in the dataset, it is stated that no two graphic logos of the 32 should be present in the same image, Figure 5.15 shows two special examples where the two top windows are correctly assigned to the class, but the top detection does not correspond to dataset ground truth.

In order to measure the performance of our approach when simultaneously dealing with images that are absent of graphic logos, we introduce the no-logo class of images from the original dataset.

(A) Class: Tsingtao (B) Class: Heineken

FIGURE 5.15: Examples of two different graphic logos in the same image

The authors of the FlickrLogos-32 dataset made available a set of tools (FlickrLogos-32_eval-kit-1.0.4) that we use to evaluate our model. The recognition precision and recognition recall are the main scores used to measure performance on this dataset, although we will also evaluate the detection precision and recall. In this context, recognition consists of correctly classifying the class of images where logos are present, detection consists of correctly classifying if a graphic logo is present or not.

Since our model classifies each image based on regions, and we do not have explicit confidence scores for the no-logo class, we use the region on the image with top confidence value and a threshold value to infer if only background regions are present. If the top confidence value is below a certain threshold value, we classify that image as a no-logo image, instead of the predicted logo class.

Introducing the 3000 no-logo images from the test set, we computed recognition F1-scores for both networks across learning rates iterations and configurations, for each threshold value on a range 0-1.

We can observe in Figure 5.16, both metrics interacting. As we set higher threshold values the model starts to correctly identify more no-logo images and the detection score rises, at the same time, recognition scores start lowering since we start to incorrectly classify logo images as having no-logo. At high threshold values both scores start to go on a downward trend, since few top confidence values reside on that range. The top recognition F1-score of 0.931 was found using the VGG_CNN_M_1024_FRCN model,

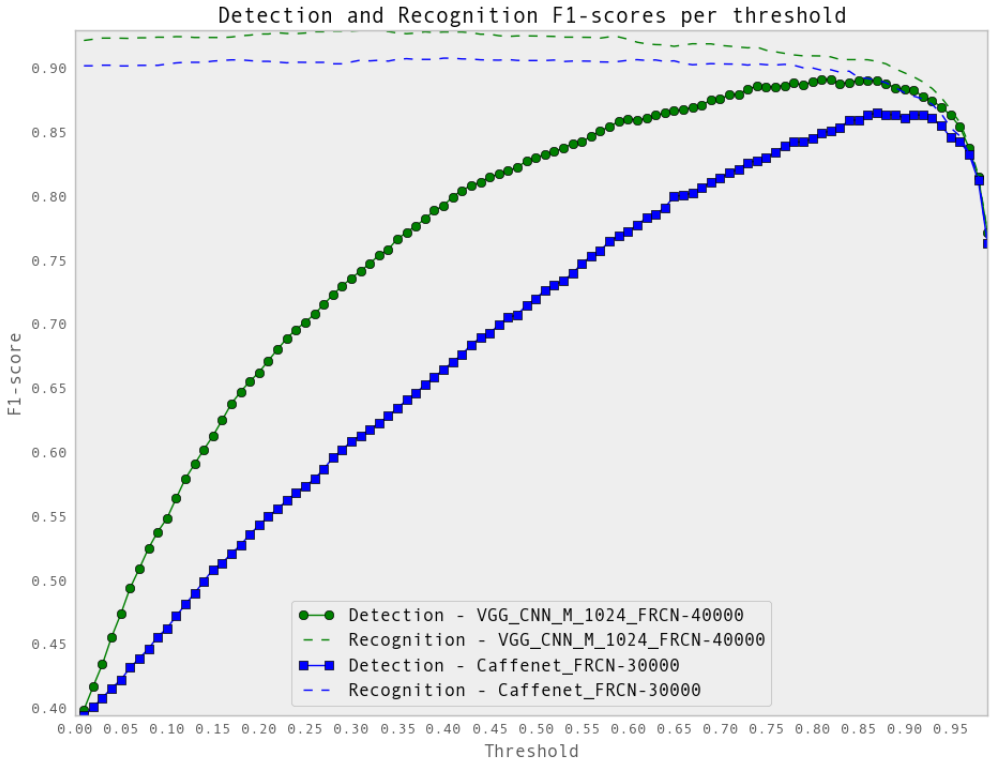


FIGURE 5.16: F1-scores per threshold value, for both VGG_CNN_M_1024_FRCN at 40000 iterations and Caffenet_FRCN at 30000 iterations, with a base learning rate of 0.001.

with a base learning rate of 0.001 at 40000 iterations and a threshold value of 0.32. The Caffenet_FRCN model achieved a top recognition F1-score of 0.909 with a base learning rate of 0.001 at 30000 iterations and with a threshold of 0.4.

Our model is able to distinguish most of the background regions from logos, but under specific conditions ends up giving high confidence values for regions that do not contain a logo from the list of 32 classes. Figure 5.17 show some of those images, for example, the Milka and Apple logo are often confused for purple backgrounds and white backgrounds respectively. Graphic logos, that are present in the images and do not belong to the 32 classes are also sometimes wrongly detected.

Method	Precision	Recall	F1
Brugman AlexNet-logos-3000	0.713	0.569	0.633
Brugman Network-In-Network-logos-3000	0.705	0.604	0.651
Brugman Caffenet-logos-3000	0.729	0.565	0.637
Romberg et al. [49]	0.98	0.61	0.752
Romberg and Lienhart [48]	0.99	0.832	0.904
Caffenet_RCNN-30000 (thresh 0.4)	0.928	0.891	0.909
VGG_CNN_M_1024_FRCN-40000 (thresh 0.81)	0.955	0.908	0.931
VGG_CNN_M_1024_FRCN-40000 (thresh 0.32)	0.987	0.846	0.911

TABLE 5.8: Recognition scores

(A) Class: no-logo (B) Class: no-logo

(C) Class: no-logo (D) Class: no-logo

FIGURE 5.17: Examples of incorrect detections in images of the no-logo class.

From Table 5.8 we can see that our model compares favorably to the current state-of-the-art.

In Figure 5.18 as a result of favoring recognition between graphic logos, we can observe that the system correctly classifies most of the logos in their midst and few images with graphic logos are incorrectly classified as no-logo. On the other hand, if we look at detection (logo or no-logo) scores, in the first row, there is an high number of images with no logo (background images) where some regions are classified as a certain logo with a confidence above our threshold.

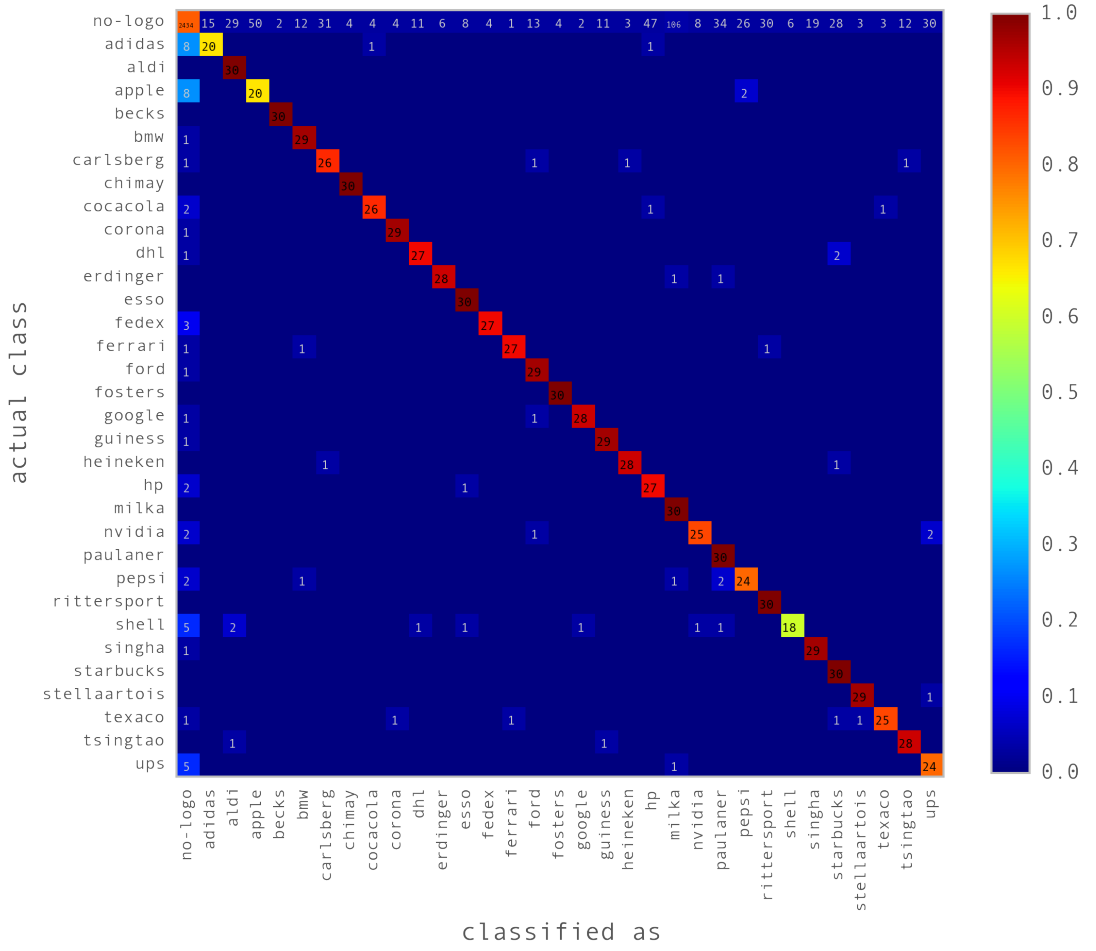


FIGURE 5.18: Confusion matrix of classification using VGG_CNN_M_1024_FRCN at 40000 iterations with threshold 0.32.

Our choice for the threshold value of 0.32 only serves as measurement against other authors works, clearly Figure 5.16 shows that is possible to achieve higher discrimination between logos and no-logos while still maintaining close recognition scores, for example, for VGG_CNN_M_1024_FRCN, setting a threshold of 0.81 would achieve 0.911 recognition F1-score and a 0.893 detection F1-score (Figure 5.16), it would be a more balanced value to cope with more images without logos in a real world scenario.

Figure 5.19 shows the results for the threshold value of 0.81, it can be observed that few logos are detected in no-logo instances, while still correctly classifying most of the images with a logo.

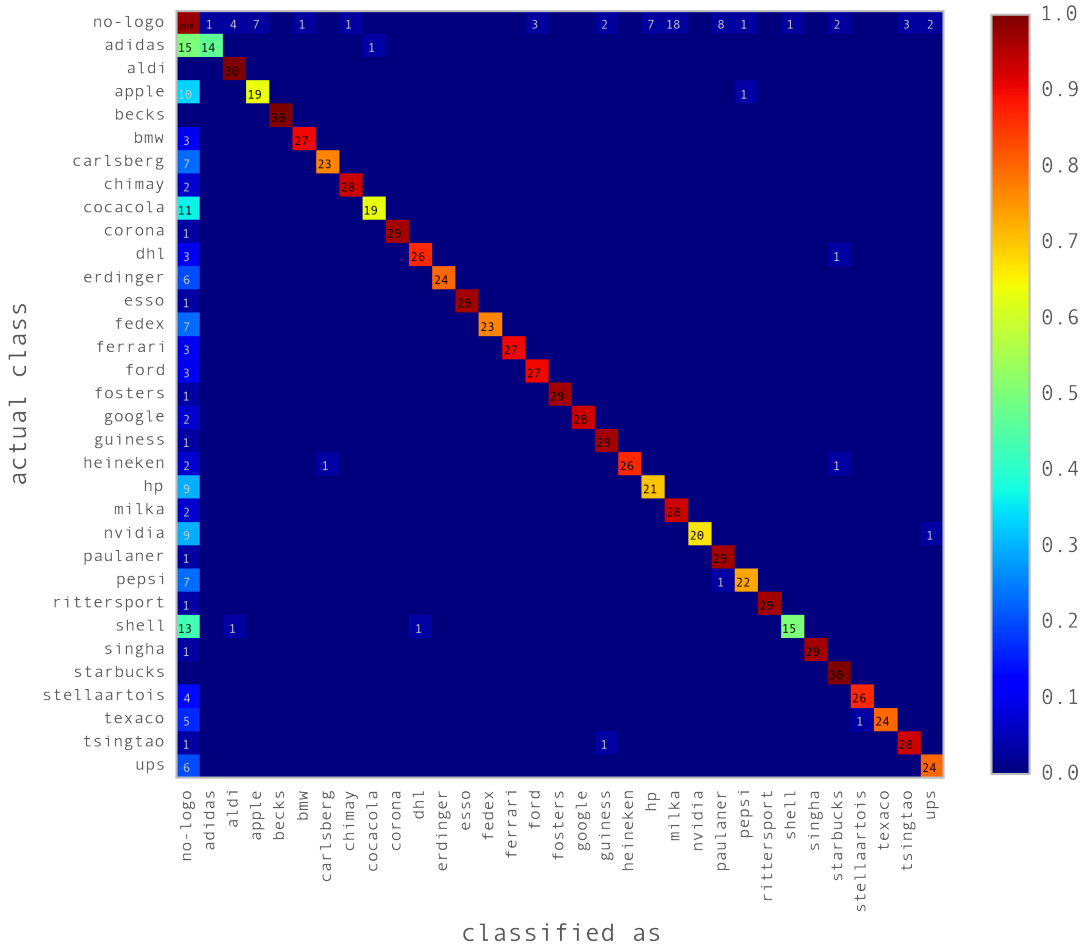


FIGURE 5.19: Confusion matrix of classification using VGG_CNN_M_1024_FRCN at 40000 iterations with threshold 0.81 .

Chapter 6

Final Conclusions and Future Work

As described in the first chapter, our main goal was to research, model, and prototype a system capable of detecting brands in images. We managed to achieve promissory results and accomplish the initial objectives. In the sections that follow, we present the conclusions of this work and possible future work that would improve this work.

6.1 Conclusions

We started our work by exploring classical approaches for object detection and classification. By reshaping the distribution of the train and test distribution we managed to achieve promissory preliminary results. We explored ways to improve the our initial approach, but technique used to select the regions for feature extraction, presented itself as a major bottleneck. Now in retrospective, the initial results are underwhelming compared to our final model.

Typically the amounts of data related to deep learning systems is extremely high, training a deep learning system from scratch using the dataset we chose seemed an impossible task. Transfer learning served to ameliorate the issue and take advantage of general image features learned by these large networks on a larger dataset. Few works using recent techniques, like CNN, over the Flickr32 Dataset [49] exist. We found the FRCN model to be a feasible and intuitive approach, given that the inner workings of the model fit into our objectives. Graphic logos generally do not occupy a large portion of the image, therefore, using regions for detection makes sense. Since we analyze by region, we have more granular control over the detection process. Although other authors experimented

with regular CNN approaches for the dataset we used, their attempts did not manage to achieve state-of-the-art performance.

A key contribution of this work has been the introduction of graphic logo detection using regions and convolutional neural networks, taking advantage of transfer learning. The model we propose compares favorably to state-of-the-art performance almost out of the box, with a wide margin for improvement at several levels. In the following section we contribute with possible improvements for the project.

6.2 Future Work

More data is the first option for improvement, gathering more graphic logo data or procedurally generating more data to augment our dataset, intuitively would allow us to achieve even better results. Inserting a single graphic logo into various background images to generate a new image is practicable, however, accurate pixel level annotations of the logo are required. Unfortunately the dataset annotations we experimented with are too inaccurate, so we did not follow with this approach.

Taking advantage of regular objects that brands are often inserted into, and building associations object-brand could be another possibility for improvement.

We used Convolutional Neural Networks adapted from other works, exploring a custom network structure for the purpose of brand detection is another option, although we showed that its possible to achieve good results using general object detection CNN.

The window proposals pre-processing step is another venue for improvement, although the method we use performs well, it could be further optimized for our purposes. There are also recent works that advocate direct learning of window proposals specific to each class. It would help reduce errors due to hard to distinguish background regions. Further research to this pre-processing step. This would also help deal with really small graphic logos on images. It is an issue for the proposed model and essential to detect brands in a wider variety of content.

The current solution is designed to work with images. Recent works on the general computer vision domain are beginning to tackle on the challenge of speed, to reduce the time taken to analyze one image. Our system main bottleneck in terms of speed is the pre-processing step (Selective Search), it ends up taking a large portion of detection time. It makes sense to improve this step to move into more demanding tasks, like brand detection in video.

Our prototype manages to achieve promissory results, and proves that its possible to build a feasible brand detection system on real world data. While it still needs improvement to be integrated into a production system, our research gives insight into future problems one might encounter and possible directions to follow. Therefore we consider our main goals fulfilled.

Appendix A

Datasets

In the next section we present some of the datasets that are relevant to the goals set in Chapter 1.

A.1 Flickr32 Logo Dataset

The Flickr32 Logo Dataset [49] contains 32 different logo brand pictures downloaded from the photo sharing website Flickr. It is meant for evaluation of multi-class logo detection, recognition systems on real-world images and logo retrieval systems.

The 32 logo classes are: Adidas, Aldi, Apple, Becks, BMW, Carlsberg, Chimay, Coca-Cola, Corona, DHL, Erdinger, Esso, Fedex, Ferrari, Ford, Foster's, Google, Guiness, Heineken, HP, Milka, Nvidia, Paulaner, Pepsi, Ritter Sport, Shell, Singha, Starbucks, Stella Artois, Texaco, Tsingtao and UPS. All images contain at least one instance of a logo. There is also a subset of images that contains no logos. All images are annotated with a positional mask of the contained logo.

Partition	Description	Images/class	#Images
Training set	Hand-picked images	10 per class	320 images
Validation set	Images showing at least a single logo under various views	30 per class	960 images
	Non-logo images	3000	3000
Test set	Images showing at least a single logo under various views	30 per class	960 images
	Non-logo images	3000	3000

TABLE A.1: Distribuition of the Flickr32 dataset partition. Training, validation, test sets are disjoint. Total of 8240 images.

A.2 Flickr Logos 27 dataset

The Flickr Logos 27 dataset [37] contains 27 logo brand pictures. Each class of brand logo contains 30 images. It is meant for evaluation of logo retrieval systems in real world images.

Partition	Description	Images/class	#Images
Training set	Images from the 27 classes	30 per class	810
Distractor set	Logo pictures outside of the 27 classes	4207	4207
Query set	Images from the 27 classes	5 per class	135

TABLE A.2: Class distribution of the Flickr 27 dataset.

The 27 logo classes are: Adidas, Apple, BMW, Citroen, Coca Cola, DHL, Fedex, Ferrari, Ford, Google, Heineken, HP, Intel, McDonalds, Mini, Nbc, Nike, Pepsi, Porsche, Puma, Red Bull, Sprite, Starbucks, Texaco, Unisef, Vodafone and Yahoo.

A.3 MICC-Logos dataset

The MICC-Logos dataset [52] is composed of 720 images downloaded from the web. It has 13 logo classes, each one represented by 15 to 87 real world images.

A.4 Imagenet

Imagenet [22] is a database that is organized by nouns according to a Wordnet hierarchy. Each node of the hierarchy can contain a massive amount of images. It has been used for a variety of large scale computer vision tasks.

A.5 TRECVID datasets

Datasets from the the past TREC Video Retrieval Evaluation challenges. Depending on the actual challenge that a specific dataset was created for, bounding box annotations for multiple kinds of targets are available. People, characters, objects and locations are some examples of such targets.

A.6 PASCAL VOC datasets

Datasets from the PASCAL Visual Object Classes challenges, each year multiple variations are introduced. One is PASCAL VOC 2012, it contains 20 classes, with annotated objects and segmentations. This is one of the most used video datasets for object detection and recognition evaluation tasks in videos. The class list is: person,bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train,bottle, chair, dining table, potted plant, sofa, tv/monitor.

Appendix B

Fast R-CNN additional results

In this appendix we present the additional results for the Fast Region-based Convolutional Network model. Namely the detailed per class report of the top recognition scores.

	precision	recall	score	support
no-logo	0.9811	0.8113	0.8882	3000
google	0.9032	0.9333	0.9180	30
apple	0.2857	0.6667	0.4000	30
adidas	0.5714	0.6667	0.6154	30
HP	0.3553	0.9000	0.5094	30
stellaartois	0.8788	0.9667	0.9206	30
paulaner	0.4412	1.0000	0.6122	30
guiness	0.7073	0.9667	0.8169	30
singha	0.6042	0.9667	0.7436	30
cocacola	0.8387	0.8667	0.8525	30
dhl	0.6923	0.9000	0.7826	30
texaco	0.8621	0.8333	0.8475	30
fosters	0.8824	1.0000	0.9375	30
fedex	0.8710	0.9000	0.8852	30
aldi	0.4839	1.0000	0.6522	30
chimay	0.8824	1.0000	0.9375	30
shell	0.7500	0.6000	0.6667	30
becks	0.9375	1.0000	0.9677	30
tsingtao	0.6829	0.9333	0.7887	30
ford	0.6444	0.9667	0.7733	30
carlsberg	0.4483	0.8667	0.5909	30
bmw	0.6744	0.9667	0.7945	30
pepsi	0.4615	0.8000	0.5854	30
esso	0.7500	1.0000	0.8571	30
heineken	0.8750	0.9333	0.9032	30
erdinger	0.8235	0.9333	0.8750	30
corona	0.8529	0.9667	0.9062	30
milka	0.2158	1.0000	0.3550	30
ferrari	0.9310	0.9000	0.9153	30
nvidia	0.7353	0.8333	0.7812	30
rittersport	0.4918	1.0000	0.6593	30
ups	0.4211	0.8000	0.5517	30
starbucks	0.4839	1.0000	0.6522	30

TABLE B.1: Per class precision and recall values using the VGG_CNN_M_1024_FRCN model, with a base learning rate of 0.001 at 40000 iterations and a threshold value of 0.32

	precision	recall	f1-score	support
no-logo	0.9821	0.7130	0.8262	3000
google	0.8788	0.9667	0.9206	30
apple	0.2427	0.8333	0.3759	30
adidas	0.3191	0.5000	0.3896	30
HP	0.3000	0.9000	0.4500	30
stellaartois	0.7941	0.9000	0.8438	30
paulaner	0.3226	1.0000	0.4878	30
guiness	0.4833	0.9667	0.6444	30
singha	0.6512	0.9333	0.7671	30
cocacola	0.8000	0.8000	0.8000	30
dhl	0.5094	0.9000	0.6506	30
texaco	0.8065	0.8333	0.8197	30
fosters	0.8108	1.0000	0.8955	30
fedex	0.9600	0.8000	0.8727	30
aldi	0.4054	1.0000	0.5769	30
chimay	0.7838	0.9667	0.8657	30
shell	0.8333	0.6667	0.7407	30
becks	0.8824	1.0000	0.9375	30
tsingtao	0.8788	0.9667	0.9206	30
ford	0.6923	0.9000	0.7826	30
carlsberg	0.7576	0.8333	0.7937	30
bmw	0.7568	0.9333	0.8358	30
pepsi	0.1348	0.8000	0.2308	30
esso	0.7143	1.0000	0.8333	30
heineken	0.8788	0.9667	0.9206	30
erdinger	0.9200	0.7667	0.8364	30
corona	0.9062	0.9667	0.9355	30
milka	0.1847	0.9667	0.3102	30
ferrari	0.9310	0.9000	0.9153	30
nvidia	0.3333	0.7667	0.4646	30
rittersport	0.6667	1.0000	0.8000	30
ups	0.1933	0.7667	0.3087	30
starbucks	0.4762	1.0000	0.6452	30

TABLE B.2: Per class precision and recall values using the Caffenet_FRCN model, with a base learning rate of 0.001 at 30000 iterations and a threshold value of 0.4

	precision	recall	f1-score	support
no-logo	0.9555	0.9797	0.9674	3000
google	1.0000	0.9333	0.9655	30
apple	0.7308	0.6333	0.6786	30
adidas	0.9333	0.4667	0.6222	30
HP	0.7500	0.7000	0.7241	30
stellaartois	0.9630	0.8667	0.9123	30
paulaner	0.7632	0.9667	0.8529	30
guiness	0.9062	0.9667	0.9355	30
singha	1.0000	0.9667	0.9831	30
cocacola	0.9500	0.6333	0.7600	30
dhl	0.9630	0.8667	0.9123	30
texaco	1.0000	0.8000	0.8889	30
fosters	1.0000	0.9667	0.9831	30
fedex	1.0000	0.7667	0.8679	30
aldi	0.8571	1.0000	0.9231	30
chimay	0.9655	0.9333	0.9492	30
shell	0.9375	0.5000	0.6522	30
becks	1.0000	1.0000	1.0000	30
tsingtao	0.9032	0.9333	0.9180	30
ford	0.9000	0.9000	0.9000	30
carlsberg	0.9583	0.7667	0.8519	30
bmw	0.9643	0.9000	0.9310	30
pepsi	0.9167	0.7333	0.8148	30
esso	1.0000	0.9667	0.9831	30
heineken	1.0000	0.8667	0.9286	30
erdinger	1.0000	0.8000	0.8889	30
corona	1.0000	0.9667	0.9831	30
milka	0.6087	0.9333	0.7368	30
ferrari	1.0000	0.9000	0.9474	30
nvidia	1.0000	0.6667	0.8000	30
rittersport	1.0000	0.9667	0.9831	30
ups	0.8889	0.8000	0.8421	30
starbucks	0.8824	1.0000	0.9375	30

TABLE B.3: Per class precision and recall values using the VGG_CNN_M_1024_FRCN model, with a base learning rate of 0.001 at 40000 iterations and a threshold value of 0.81

Appendix C

Minutes of the meetings

In this appendix we list the minutes of all formal meetings held over the year. Each one of the following sections represents a meeting.

11 September 2014

Participantes

Estagiário Gonçalo Palaio
Engº André Pimentel EyeSee
Engº Joao Redol EyeSee
Profª Bernardete Ribeiro DEI

Resumo

A reunião decorreu com a presença dos responsáveis da EyeSee, Engº João Redol e Engº André Pimentel, no Laboratório LARN, Torre E, Piso 6, Sala, das 14:30h-17:30h, e terá a seguinte ordem de trabalhos:
1- Apresentação Da Empresa
2- Objectivos Dos Estágio
3- Tecnologias
4- Datasets
5- Planificação Do Estágio
6- Templates Dos Reports Do 1º Semestre
7- Local Do Estágio: Larn/dei

Foram discutidos vários aspetos do Estágio tendo o Estagiário ficado de entregar até ao final de Setembro o capitulo estado da arte. Foram estabelecidas reuniões regulares com os responsáveis da EyeSee por skype.
Foi criada uma partilha de endereços de skype para se poder dar continuidade ás reuniões quinzenais e mensais.
Foram entregues chaves do LARN e arranjada a logística para que possa trabalhar no Laboratorio.
Foi ainda o Estagiário convidado a visitar a empresa ainda durante o primeiro semestre.

26 September 2014

Participantes

Estagiário: Gonçalo Palaio
Orientadora: Bernardete Ribeiro

Resumo

Discussão e Duvidas sobre o Estado da Arte Identificação de Problemas

Ponteiro de software de Machine Learning `http://jmlr.csail.mit.edu/mloss/`

11 October 2014

Participantes

Estagiário: Gonçalo Palaio
Orientadora: Bernardete Ribeiro

Resumo

Foram discutidos os seguintes pontos:
Relatório de progresso - De momento tenho uma breve introdução e uma versão preliminar do capitulo de estado de arte assente sobre a literatura associada e soluções comerciais.
Discussão da possibilidade de receber feedback do capitulo já referido.
Inserção do relatório de progresso na plataforma de estágios - A opção está bloqueada de momento.
Questões sobre a falta de informação das soluções comerciais.

Discussão da próxima etapa do plano de trabalho, Dataset gathering e Model Definition, ficou acordado que a partir da escolha dos datasets, abordagens para brand recognition devem ser exploradas de forma hands-on.

25 March 2015

Participantes

Estagiário Gonçalo Palaio
Estagiário Daniel Frutuoso
Engº André Pimentel, EyeSee
Profª Bernardete Ribeiro, DEI

Resumo

A reunião decorreu presencialmente das 14:30 até as 18:00 no LARN. Estiveram presentes a Professora Bernardete Ribeiro, Engenheiro André Pimentel (EYESEE) e os estagiários Gonçalo Oliveira, Daniel Frutuoso e Ana Laranjeira.
Após uma apresentação sobre o estado atual do projeto por parte do estagiário Gonçalo Oliveira, foram discutidos os seguintes tópicos:
- Abordagem atual para detecção de logotipos e problemas associados com a mesma. Escolha do método carece justificação.
- Discussão sobre a escolha dos algoritmos explorados até ao momento e viabilidade dos mesmos. Necessidade de argumentar o porquê da escolha deles.
- Resultados atuais obtidos. Necessidade de existir comparação com o state of the art.
- Próxima etapa do projeto e possíveis direções a tomar. Localizar logotipo com o intuito de melhorar os resultados obtidos. Explorar o algoritmo de clustering G-Means para a criação do dicionário de visual words.

26 May 2015

Participantes

Estagiário Gonçalo Palaio
Estagiário Daniel Frutuoso
Estagiário Ana Laranjeira
Engº André Pimentel, EyeSee

Prof[a] Bernardete Ribeiro, DEI

Resumo

A reunião decorreu das 11:00 até ao 12:30, no LARN. Estiveram presentes na reunião a Professora Bernardete Ribeiro, Engenheiro André Pimentel e os estagiários Gonçalo Oliveira, Daniel Frutuoso e Ana Laranjeira.
Após uma apresentação sobre o estado atual do projeto por parte do estagiário Gonçalo Oliveira, foram discutidos os seguintes tópicos:
- Resultados obtidos.
- Próximas etapas do projeto. Localização, ainda que não exata do logotipo para seleção de features.
- Datas de entrega.

References

[1] Recognizing and learning object categories. http://people.csail.mit.edu/torralba/shortCourseRLOC/. Accessed: 2015-08-20.

[2] Caffe deep learning framework. http://caffe.berkeleyvision.org/, . Accessed: 2015-08-20.

[3] Caffe model zoo. http://caffe.berkeleyvision.org/model_zoo.html, . Accessed: 2015-08-20.

[4] Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. Accessed: 2015-08-20.

[5] Dataset: Flickrlogos-32. http://www.multimedia-computing.de/flickrlogos/. Accessed: 2015-08-20.

[6] Fast r-cnn: Fast region-based convolutional networks for object detection - edison research version. https://github.com/EdisonResearch/fast-rcnn. Accessed: 2015-08-20.

[7] Ross b. girshick homepage. http://www.cs.berkeley.edu/~rbg/, . Accessed: 2015-08-20.

[8] Rcnn poster. http://www.cs.berkeley.edu/~rbg/slides/rcnn-poster.pdf, . Accessed: 2015-08-20.

[9] Rich feature hierarchies for accurate object detection and semantic segmentation. supplementary material. http://www.cs.berkeley.edu/~rbg/papers/r-cnn-cvpr-supp.pdf, . Accessed: 2015-08-20.

[10] Selective search for object recognition. https://ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=UijlingsIJCV2013&bib=all.bib. Accessed: 2015-08-20.

[11] Vlfeat. http://www.vlfeat.org/. Accessed: 2015-08-20.

[12] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2189–2202, 2012.

[13] Lamberto Ballan, Marco Bertini, and Arjun Jain. A system for automatic detection and recognition of advertising trademarks in sports videos. October 2008.

[14] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[15] Simon Brugman. Computationally feasible logo recognition using deep learning. Bachelor thesis, Radboud University, July 2015.

[16] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[17] John" Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[18] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[19] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[20] Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. pages 396–404, 1990.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[23] Yining Deng and BS Manjunath. Unsupervised segmentation of color-texture regions in images and video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(8):800–810, 2001.

[24] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. October 2013.

[25] Kaibo Duan, S Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In *Multiple Classifier Systems*, pages 125–134. Springer, 2003.

[26] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[27] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[28] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[29] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[30] Xavier Frazão and Luís A Alexandre. Dropall: Generalization of two convolutional neural network regularization methods. In *Image Analysis and Recognition*, pages 282–289. Springer, 2014.

[31] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. November 2013.

[33] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[34] Daniela Hall, Fabien Pelisson, Olivier Riff, and James L. Crowley. Brand identification using Gaussian derivative histograms. *Machine Vision and Applications*, 16 (1):41–46, December 2004.

[35] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *arXiv preprint arXiv:1502.05082*, 2015.

[36] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[37] Y. Kalantidis, LG. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis. Scalable triangulation-based logo recognition. In *in Proceedings of ACM International Conference on Multimedia Retrieval (ICMR 2011)*, Trento, Italy, April 2011.

[38] D Kenny and JF Marshall. Contextual marketing–the real business of the internet. *Harvard business review*, 78(6):119, 2000.

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

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[41] Wei-Shing Liao, Kuan-Ting Chen, and Winston H. Hsu. Adimage: Video advertising by image matching and ad scheduling optimization. pages 767–768, 2008.

[42] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[43] Rohit Pandey, Wei Di, Vignesh Jagadeesh, Robinson Piramuthu, and Anurag Bhardwaj. Cascaded sparse color-localized matching for logo retrieval. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 2207–2211. IEEE, 2014.

[44] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[45] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[48] Stefan Romberg and Rainer Lienhart. Bundle min-hashing. *International Journal of Multimedia Information Retrieval*, 2(4):243–259, 2013.

[49] Stefan Romberg, Lluis Garcia Pueyo, Rainer Lienhart, and Roelof van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 25:1–25:8, New York, NY, USA, 2011. ACM.

[50] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 430–443. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33832-1. doi: 10.1007/11744023_34.

[51] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[52] Hichem Sahbi, Lamberto Ballan, Giuseppe Serra, and Alberto Del Bimbo. Context-dependent logo matching and recognition. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 22(3):1018–31, March 2013.

[53] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

[56] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[57] Koen E. a. van de Sande, Jasper R. R. Uijlings, Theo Gevers, and Arnold W. M. Smeulders. Segmentation as selective search for object recognition. *2011 International Conference on Computer Vision*, (2):1879–1886, November 2011.

[58] Chengde Wan, Zhicheng Zhao, Xin Guo, and Anni Cai. Tree-based shape descriptor for scalable logo detection. In *Visual Communications and Image Processing (VCIP), 2013*, pages 1–6, Nov 2013.

[59] Jinqiao Wang, Jianlong Fu, and Hanqing Lu. Finding logos in real-world images with point-context representation-based region search. *Multimedia Systems*, December 2013.

[60] Pengfei Xu, Hongxun Yao, and Rongrong Ji. Sigma: Spatial integrated matching association algorithm for logo detection. pages 1086–1089, March 2010.

[61] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[62] Jianming Zhang, M Sameki, S Ma, B Price, R Mech, X Shen, M Betke, S Sclaroff, and Z Lin. Salient object subitizing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.