Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2020

Júlio Miguel Braz da Costa Silva

Recuperação e Identificação de Momentos em Imagens

Recovery and Identification of Moments in Images

Júlio Miguel Braz da Costa Silva

Recuperação e Identificação de Momentos em Imagens

Recovery and Identification of Moments in Images

Dissertação de Mestrado apresentada à Universidade de Aveiro, para obtenção do grau de Mestre em Engenharia Eletrónica e de Telecomunicações, sob orientação cientifica Professor Doutor António Neves, Professor do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**ABC** Professor Catedrático da Universidade de Aveiro (por delegação da Reitora da Universidade de Aveiro)
vogais / examiners committee	**DEF** Professor Catedrático da Universidade de Aveiro (orientador)
	GHI Professor associado da Universidade J (co-orientador)
	KLM Professor Catedrático da Universidade N

agradecimentos / acknowledgements

Aos meus pais e à minha irmã por ao longo dos anos terem sido sempre imprescindíveis, nunca me terem faltado, proporcionando sempre todo o apoio necessário e condições que me levaram a alcançar os meus objetivos e me tornar a pessoa que hoje sou.

Aos meus amigos de Tondela que tanta paciência tiveram para me aturar ao longo dos anos.

Aos amigos que Aveiro me proporcionou que tantas histórias criámos juntos.

À Mariana Coutinho por todos os bons momentos, apoio, carinho e companheirismo que me facilitaram este último ano.

Ao meu colega de casa Pedro Nunes a pessoa com quem tantas longas conversas tive e a que mais me chateou a cabeça para despachar as cadeiras e a dissertação.

Ao professor António Neves que apesar das dificuldades impostas pelo trabalho à distância sempre me motivou e orientou, sugerindo sempre possíveis soluções e caminhos a percorrer.

Ao Ricardo Ribeiro que tanta disponibilidade teve para me ajudar com todas as minhas dúvidas. Uma ajuda fundamental no trabalho realizado.

Palavras-Chave

Visão por Computador, Processamento Natural de Texto, ImageCLEF, Lifelogging, Recuperação de Momentos

Resumo

Na sociedade moderna, praticamente qualquer pessoa consegue capturar momentos e registar eventos devido à facilidade de acesso a *smartphones*. Isso leva à questão, se registamos tanto da nossa vida, como podemos facilmente recuperar momentos específicos? A resposta a esta questão abriria a porta para um grande salto na qualidade da vida humana. As possibilidades são infinitas, desde problemas triviais como encontrar a foto de um bolo de aniversário até ser capaz de analisar o progresso de doenças mentais em pacientes ou mesmo rastrear pessoas com doenças infecciosas.

Com tantos dados a serem criados todos os dias, a resposta a esta pergunta torna-se mais complexa. Não existe uma abordagem linear para resolver o problema da localização de momentos num grande conjunto de imagens e investigações sobre este problema começaram há apenas poucos anos. O ImageCLEF é uma competição onde investigadores participam e tentam alcançar novos e melhores resultados na tarefa de recuperação de momentos a cada ano.

Este problema complexo, em conjunto com o interesse em participar na tarefa ImageCLEF Lifelog Moment Retrieval, apresentam-se como um bom desafio para o desenvolvimento desta dissertação.

A solução proposta consiste num sistema capaz de recuperar automaticamente imagens de momentos descritos em formato de texto, sem qualquer tipo de interação de um utilizador, utilizando apenas métodos estado da arte de processamento de imagem e texto.

O sistema de recuperação desenvolvido alcança este objetivo através da extração e categorização de informação relevante de texto enquanto calcula um valor de similaridade com os rótulos extraídos durante a fase de processamento de imagem. Dessa forma, o sistema consegue dizer se as imagens estão relacionadas ao momento especificado no texto e, portanto, é capaz de recuperar as imagens de acordo.

Na subtarefa ImageCLEF Life Moment Retrieval 2020, o sistema de recuperação automática de imagens proposto alcançou uma pontuação de 0.03 na metodologia de avaliação F1-measure@10. Mesmo que estas pontuações não sejam competitivas quando comparadas às pontuações de outros sistemas de outras equipas, o sistema construído apresenta-se como uma boa base para trabalhos futuros.

Keywords **Computer Vision, Natural Language Processing, ImageCLEF, Lifelogging, Moment Retrieval**

Abstract

In our modern society almost anyone is able to capture moments and record events due to the ease accessibility to smartphones. This leads to the question, if we record so much of our life how can we easily retrieve specific moments? The answer to this question would open the door for a big leap in human life quality. The possibilities are endless, from trivial problems like finding a photo of a birthday cake to being capable of analyzing the progress of mental illnesses in patients or even tracking people with infectious diseases.

With so much data being created everyday, the answer to this question becomes more complex. There is no stream lined approach to solve the problem of moment localization in a large dataset of images and investigations into this problem have only started a few years ago. ImageCLEF is one competition where researchers participate and try to achieve new and better results in the task of moment retrieval.

This complex problem, along with the interest in participating in the ImageCLEF Lifelog Moment Retrieval Task posed a good challenge for the development of this dissertation.

The proposed solution consists in developing a system capable of retriving images automatically according to specified moments described in a corpus of text without any sort of user interaction and using only state-of-the-art image and text processing methods.

The developed retrieval system achieves this objective by extracting and categorizing relevant information from text while being able to compute a similarity score with the extracted labels from the image processing stage. In this way, the system is capable of telling if images are related to the specified moment in text and therefore able to retrieve the pictures accordingly.

In the ImageCLEF Life Moment Retrieval 2020 subtask the proposed automatic retrieval system achieved a score of 0.03 in the F1-measure@10 evaluation methodology. Even though this scores are not competitve when compared to other teams systems scores, the built system presents a good baseline for future work.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Inteliggence.

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CV Computer Vision.

IE Information Extraction.

LMRT Lifelog Moment Retrieval Task.

NLP Natural Language Processing.

ResNet Residual Network.

SSD Single Shot Detection.

YOLO You Only Look Once.

CHAPTER 1

Introduction

The pervasive creation and consumption of visual media content is ingrained into our modern world. In the past, the main purpose given to pictures was to save moments of events. Nowadays people are constantly consuming visual media content. Images have many different usages, not only we use them for social media but also we use them in engineering, in art, in science, in medicine, in entertainment and also in advertising [1].

With the rapid development of Internet of things (IOT) this growth in consumption of visual media content has increased the usage of wearable and smart technologies making the subject of lifelogging more prevalent in the recent years. Lifelogging is the task of tracking and recording personal data created trough the activities and behaviour of individuals during their day-to-day life in the form of images, video, biometric data, location and other data. The name given to the data created by lifelogging has the name of "lifelog data" and it is rich in resources for contextual information retrieval [2].

Some examples of the usefulness of lifelogging is using it as a memory extension for people who suffer from memory impairments such as Alzheimers, to find lost items during the day or even to understand human behaviour.

The technical problems related to creating, compressing, storing, transmitting, rendering and protecting image data are mostly solved. However there still exists two difficult problems to tackle which are the issues associated with image location and the continuous growth of image data (big data) [1].

"Locating images involves analysing them to determine their content, classifying them into related groupings, and searching for images. In order to solve these problems, the current technology relies heavily on the image description" [1], usually called as "image metadata". This data can either be added automatically at the capturing time or manually added afterwards.

According to the literature [1]: "In the present time the development in the area of content–based analysis (indexing and searching of visual media) is increasing, this is where most of the research in image management is concentrated. Automatic analysis of the content of images, which in turn would open the door to content–based indexing, classification and retrieval, is an inherently difficult problem and therefore progress is slow."

However, if one day a fully automatic image/video retrieval system is implemented it will vastly improve the life quality of the human kind. A great example that we can apply at the present time is that it will be possible to backtrack the last few days of humans infected with COVID-19 through their lifelog data, which in turn would help to identify more possible infected and warn more people to get tested.

1.1 | Challenges

As it has been described earlier in the chapter, creating an automatic system capable of fully analysing the content of images is a difficult problem. This difficulty comes from two main challenges which are image processing and text processing.

Creating an automatic system capable of image retrieval means that the computer has to be able to understand images and text while at the same time being capable of relating both.

For the image processing challenge, the computer has to be able to extract relevant information from images like colors, objects, places, locations, indoors, outdoors, activities happening in the photo, people, etc. However, in order to do this, many different and complex algorithms have to be implemented like object detection, activity recognition, scene recognition and others. The usage of several algorithms can require extreme computational time and resources depending on the size of the dataset to be analysed. If one image requires 1 second to be fully processed by an algorithm, a dataset of 200.000 images of the same resolution would require approximately 2 days. This processing time might be different depending of the algorithm used, since it can either extract more or less labels from the images. However, one thing is certain, the more algorithms that are implemented for label extraction, the more time it will take to process the image. It is necessary to carefully select which ones to use if there is an intent in saving computer resources and processing time.

Tackling the text processing challenge requires the usage of Natural Language Processing algorithms for the extraction of linguistic annotations from the text, and the implementation of semantic and syntax rules in order to enable the computer to automatically categorize the extracted words according predefined categories like “activities”, “locations”, “relevant things”, etc.

Finally, the computer has to be capable of comparing the extracted features from the text with extracted features from the images, in order to associate images to text.

1.2 | Contributions

Since the process of automatic image retrieval is still a complex problem this works aims at contributing with a baseline system for future investigations with some suggestions on how to improve it further. Additionally a study of the available technology is conducted that may help on finding new and better paths for future investigations on automatic image retrieval.

The main contribution of this work was the development of an automatic image retrieval system with the objective of participating in the ImageCLEF Lifelog challenge, more specifically in the LMRT subtask (described in Chapter 2). The participation in this challenge was done using two different systems, one being interactive and the other automatic. The system built in this work was capable of achieving a 0.031 score while the interactive system achieved a more competitive result of 0.517 in the F1-measure@10 metric [3].

To achieve the goal of creating an automatic image retrieval system other tasks had to be fulfilled. Firstly, a study on the state of the art of image and text processing was done in order to choose the algorithms to be used for label extraction. Subsequently, an algorithm was built capable of processing a large dataset of images with the aim of extracting relevant features. Following, another algorithm was programed capable of processing text with the intention of extracting relevant words and categorize them accordingly. Afterwards, in order to compare the extracted features from images and text another algorithm was built capable

of associating images to text. This algorithm recurs to a similarity function that computed a confidence score for every image. After computing a confidence score for every image, an algorithm was created capable of computing F1-measure@X score of the final results in order to allow the system of self evaluating. Finally, with the aim of facilitating the process a batch script was created in order to allow the system to run with one click.

As a result of the work developed and participation in the challenge the paper "UA.PT Bioinformatics at ImageCLEF 2020: Lifelog Moment Retrieval Web based Tool" (Ricardo Ribeiro & Júlio Silva, 2020) was published.

1.3 | Document Structure

This document has a total of 7 chapters and an appendix that are divided accordingly:

- Chapter 1 presents the context and motivation along with the challenges and contributions of this work.
- Chapter 2 discusses the imageCLEF Lifelog Challenge and the concept of lifelogging.
- Chapter 3 provides a survey on the subject of feature extraction from images while giving an introduction to some important concepts like computer vision, machine learning, deep learning, artificial intelligence, neural networks and so on. An overview of the current state of the art and latest achievements is also exposed.
- Chapter 4 addresses the thematic of extracting data from text. The subjects of natural language processing and respective applications, word embeddings, useful libraries and models are examined.
- Chapter 5 provides an overview on how the automatic image retrieval system was built. Firstly, the image processing stage is explained and the tests that were run are presented. Secondly, it is described how the system manages to extract information from text while categorizing the respective data in predefined categories automatically. Finally, it is made clear how the system is able to compare the extracted visual data and the extracted textual data in order to retrieve images accordingly.
- Chapter 6 presents the achieved results in the ImageClef LMRT challenge. Examples of system performance are showcased, distinctions between the submitted runs are made clear and the difference of the scores achieved between the automatic system and the interactive system are discussed.
- Chapter 7 describes the conclusions taken from the development of the work and provides some ideas for future investigations and improvements.
- Appendix A gives an overview of some of the most common neural networks architectures, object detection systems and image classification models.

CHAPTER 2

ImageCLEF Lifelog Challenge

This chapter aims at describing the ImageCLEF Lifelog challenge. Firstly in Section 2.1 an introduction is given to the challenge and the respective goals. Section 2.2 describes the tasks available for the year 2020. The concept of lifelogging is explained in Section 2.3. Finally Section 2.4 clarifies the Lifelog Moment Retrieval Task (LMRT) which is the main focus of this work, along with an introduction to the dataset, dev topics, test topics, ground truth and the evaluation methodology of the task.

2.1 | The ImageCLEF challenge

The ImageCLEF challenge is a large–scale evaluation campaign that aims at evaluating cross-language image retrieval systems. It is organized as part of the CLEF Initiative (Conference and Labs of the Evaluation Forum, formerly known as Cross-Language Evaluation Forum) and launched in 2003. Initially proposed by Mark Sanderson and Paul Clough from the Department of Information Studies from the University of Sheffiel with the goal of providing support for the evaluation of 1) language-independent methods for the automatic annotation of images with concepts, 2) multimodal information retrieval methods based on the combination of visual and textual features, and 3) multilingual image retrieval methods, so as to compare the effect of retrieval of image annotations and query formulations in several languages.

Every year an evaluation cycle campaign occurs that consist in workshops where teams can compete to achieve the best possible results while discussing new techniques and ideas. In addition to offering the evaluation platform, ImageCLEF also provides several publicly resources, such as benchmarks to evaluate retrieval systems. These benchmarks have helped researchers develop new approaches to visual information retrieval and automatic annotation by enabling the performance of various approaches to be assessed.

Since the launch of ImageCLEF, researchers within academic and commercial research groups worldwide, including those from Cross–Language Information Retrieval (CLIR), medical informatics, Content–Based Image Retrieval (CBIR), computer vision and user interaction have been participating in the challenge.

Currently, ImageCLEF main goal is to support the advancement of the field of visual media analysis, indexing, classification, and retrieval, by developing the necessary infrastructure for the evaluation of visual information retrieval systems operating in both monolingual, cross–language and language-independent contexts [1].

2.2 | The Tasks

The ImageCLEF 2020 edition presents 4 different tasks:

- **ImageCLEFlifelog**: Addresses the problems of lifelogging data retrieval and summarization. The work done in this thesis aims at participating in this task, therefore this task can be read in more detail in Section 2.4.
- **ImageCLEFcoral**: Addresses the problem of automatically segmenting and labeling a collection of images that can be used in combination to create 3D models for the monitoring of coral reefs.
- **ImageCLEFmedical**: The task combines the most popular medical tasks of ImageCLEF and continues the last year idea of combining various applications, namely: automatic image captioning and scene understanding, medical visual question answering and decision support on tuberculosis. This allows to explore synergies between the tasks.
- **ImageCLEFdrawnUI**: The task addresses the problem of automatically recognizing hand drawn objects representing website UIs, that will be further translated into automatic website code.

2.3 | The concept of lifelogging

Lifelogging is defined as a form of pervasive computing consisting of a unified digital record of the totality of an individual's experiences, captured multimodally through digital sensors and stored permanently as a personal multimedia archive. In a simple way, lifelogging is the process of tracking and recording personal data created through our activities and behaviour.

Personal lifelogs have a great potential in numerous applications, including memory and moments retrieval, daily living understanding, diet monitoring, or disease diagnosis, as well as other emerging application areas. For example: in Alzheimer's disease, people with memory problems can use a lifelog application to help a specialist follow the progress of the disease, or to remember certain moments from the last days, weeks or even months.

One of the greatest challenges of lifelog applications is the large amount of lifelog data that a person can generate. The lifelog datasets, for example the ImageCLEFlifelog dataset, are rich multimodal datasets which consist in one or more months of data from multiple lifeloggers. Therefore, an important aspect is the lifelog data organization in the interest of improving the search and retrieval of information. In order to organize the lifelog data, useful information has to be extracted from it [4] [3].

2.4 | ImageCLEFlifelog

The ImageCLEFlifelog 2020 task is divided into two different sub-tasks: the Lifelog moment retrieval (LMRT) and Sport Performance Lifelog (SPLL) sub-task. In this work, as in the previous year's challenge, it was only addressed the LMRT sub-task, as a continuous research work that is intended to be developed with the aim of giving a contribution to real problems that exist around the world that can benefit from this technology.

The UA.PT Bioinformatics, a team from the Institute of Electronics Engineering and Telematics in the University of Aveiro, participated in the LMRT subtask with two different retrieval systems. The first one is the automatic retrieval system which was a continuation of the work done in the previous year challenge [4] and the main objective of this thesis. The other one was a retrieval system capable of providing user interaction and visualization.

The interactive retrieval system is only interesting for this work in terms of comparing the achieved results, therefore this document will only give a small description of this system in Chapter 6 Section 6.3.1. More details are presented in [3].

2.4.1 | SubTask: Lifelog Moment Retrieval

In the LMRT subtask, the main objective is to create a system capable of retrieving a number of predefined moments in a lifelogger's day-to-day life from a set of images. Moments can be defined as semantic events or activities that happen at any given time during the day. For example, given the query "Find the moment(s) when the lifelogger was having an icecream on the beach" the participants should return the corresponding relevant images that show the moments of the lifelogger having icecream at the beach. Like last year, particular attention should be paid to the diversification of the selected moments with respect to the target scenario.

ImageCLEFlifelog dataset is a rich multimodal dataset which consists of 4.5 months of data from three lifeloggers, namely: images (1,500-2,500 per day), visual concepts (automatically extracted visual concepts with varying rates of accuracy), semantic content (locations and activities) based on sensor readings on mobile devices (via the Moves App), biometrics information (heart rate, galvanic skin response, calories burn, steps, continual blood glucose, etc.), music listening history and computer usage. Except for the images, all of the data was provided in a csv file. An excerpt of this file is presented in Figure 2.1 and examples of images from the dataset are showcased in Figure 2.2.

```
minute_id,utc_time,local_time,timezone,lat,lon,semantic_name,elevation,speed,heart,calories,activity_type,steps
20150223_0000,UTC_2015-02-23_00:00,2015-02-23_00:00,Europe/Dublin,53.3892,-6.15827,Home,NULL,NULL,NULL,1.2062000036239624,NULL,NULL
20150223_0001,UTC_2015-02-23_00:01,2015-02-23_00:01,Europe/Dublin,53.3892,-6.15827,Home,NULL,NULL,NULL,1.2062000036239624,NULL,NULL
20150223_0002,UTC_2015-02-23_00:02,2015-02-23_00:02,Europe/Dublin,53.3892,-6.15827,Home,NULL,NULL,NULL,1.2062000036239624,NULL,NULL
20150223_0003,UTC_2015-02-23_00:03,2015-02-23_00:03,Europe/Dublin,53.3892,-6.15827,Home,NULL,NULL,NULL,1.2062000036239624,NULL,NULL
```

Figure 2.1: Excerpt of the csv file provided by the organizers.

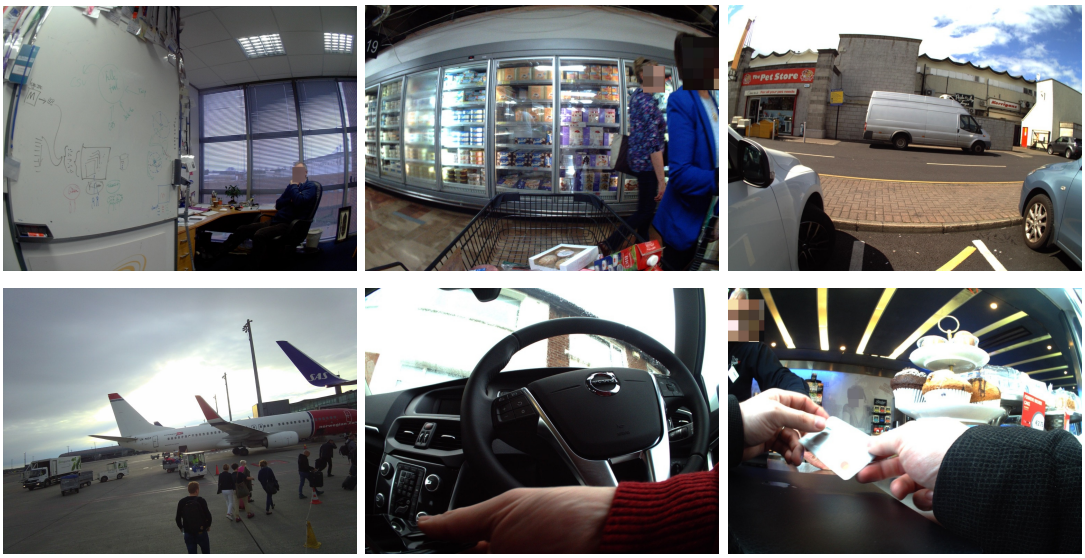


Figure 2.2: Example of images from the imageCLEF dataset.

In this work only the images, the respective semantic content and time were used [3]. This decision is mainly justified by the fact that much of the provided data like the biometric information, computer usage and music history gives no useful information for the retrieval of images. The visual concepts provided were also not utilized since the system created in this work uses its own algorithms in order to extract more accurate labels from the images.

Initially, the dev topics are firstly released along with the images dataset and the corresponding ground truth. This means that it is possible to initially create a retrieval system and analyse if it is producing good results, since thanks to the ground truth it is known which pictures should be retrieved for each textual topic.

After a few weeks the test topics for evaluation are released without the ground truth, the participants who achieve the best results are the ones who have the highest F1-measure at the top 10 images score. This is further discussed in Section 2.4.4.

2.4.2 | Dev Topic example

As discussed above the participants should return the corresponding relevant images that show the moments of the lifelogger during a predefined moment.

Those moments are provided in the format of a pdf file that contains 10 different textual query topics representing 10 different moments. Topic 1 is illustrated below to serve as an example of the query topics:

Title : "Having Beers in a Bar"
Description : "Find the moment in 2015 and 2016 when u1 enjoyed beers in the bar."
Narrative : "To be considered relevant, u1 must be clearly in a bar. Any moments that u1 drinks beers at home or outside without the bar view are not considered relevant."

- **Example of the ground truth**

The ground truth is given as text file with the following format : **[topic number, image name, cluster]**. The cluster number is used to calculate the F1-measure score which will be explained in more detail in Section 2.4.4.

```
1, b00001215_21i6bq_20150306_174513e.jpg, 1
1, b00001216_21i6bq_20150306_174552e.jpg, 1
1, b00001217_21i6bq_20150306_174713e.jpg, 1
1, b00001218_21i6bq_20150306_174751e.jpg, 1
1, b00001219_21i6bq_20150306_174822e.jpg, 1
1, b00001220_21i6bq_20150306_174858e.jpg, 1
1, b00001221_21i6bq_20150306_174935e.jpg, 1
1, b00001222_21i6bq_20150306_175048e.jpg, 1
1, b00001223_21i6bq_20150306_175126e.jpg, 1
1, b00001224_21i6bq_20150306_175202e.jpg, 1
1, b00001225_21i6bq_20150306_175316e.jpg, 1
1, b00001226_21i6bq_20150306_175355e.jpg, 1
```

Figure 2.3: Excerpt of the ground truth for the dev topic 1.

- **Example of corresponding pictures**

The dataset is composed of nearly 200.000 images. Figure 2.4 illustrates lifelog pictures from the dev topic 1 that correspond to the ground truth given in Figure 2.3:

Figure 2.4: Example of 3 images that belong to the ground truth of the topic 1.

2.4.3 | Test Topic example

An example of one of test topics used for evaluation in the challenge the test topic 7 is presented next:

> **Title** : "Seafood at Restaurant."
>
> **Description** : "Find moments when u1 was eating seafood in a restaurant in the evening time."
>
> **Narrative** : "The moments show u1 was eating seafood in any restaurant in the evening time are considered relevant. Any dish has seafood as one of its parts is also considered relevant. Some examples of the seafood can be shrimp, lobster, salmon."

Something important to notice is that the dev and test topics share similarities in the text syntax.

2.4.4 | Evaluation Methodology

In order to evaluate performance, the organizers use the F1-measure at X (F1@X) evaluation method. The F1-measure is the harmonic mean of both Cluster Recall at X (CR@X) metric and the Precision at X (P@X) measure. The Cluster recall is a metric that assesses how many different clusters from the ground truth are represented among the top X results while the Precision measures the number of relevant photos among the top X results [5]. Figure 2.5 illustrates these calculations.

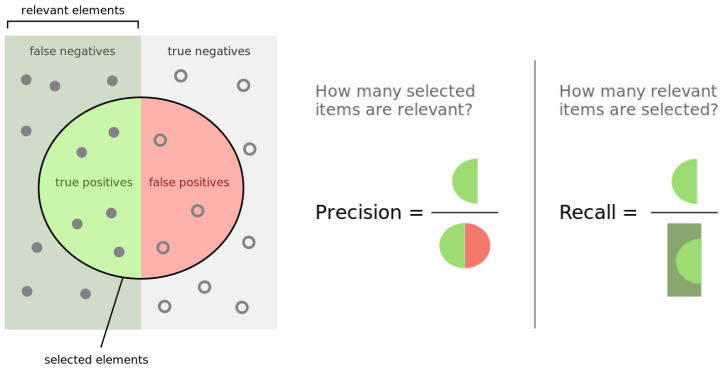


Figure 2.5: Illustration of Recall and Precision calculations [6].

However, in the challenge these calculations are done as follows:

$$\textbf{Cluster Recall at X (CR@X)} = N/Ngt$$
$$\textbf{Precision at X (P@X)} = Nr/X$$
$$\textbf{F1@X} = 2 \times (P@X \times CR@X)/(P@X + CR@X)$$

N is the number of image clusters represented in the first X ranked images. Ngt is the total number of image clusters from the ground truth. Nr is the number of relevant pictures from the first X ranked results.

This year edition official rankings are obtained trough the F1-measure@10, which gives equal importance to diversity (via CR@10) and relevance (via P@10). Another important aspect of a F1-measure@10 is that only the top 10 pictures for each topic with the highest confidence score are accountable for performance assessment.

2.4.5 | Evaluation Problems

The main problem in the evaluation of the imageCLEF lifelog subtask is that pictures that could be considered to belong to a given moment are at times not present in the ground truth and therefore decrease the score in the F1-measure evaluation. In addition to this problem, some pictures accounted in the ground truth should have not been considered.

Dev topic 4 is presented next to serve as an example.

> **Title**: "Television Recording."
>
> **Description**: "Find the moments when u1 was being recorded for a television show."
>
> **Narrative** : "To be considered relevant, there must clearly be a television camera in front of u1. The moments the interviewer/cameramen is interviewing/recording u1 are also considered relevant. This can take place at home or in another location. All recording took place in one day and in more than one location."

Figure 2.6: Sample of pictures that are considered in the ground truth of dev topic 4 but should not have been.

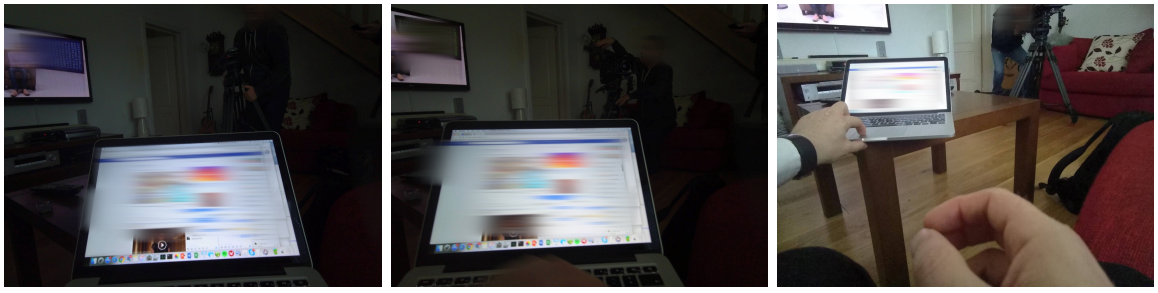


Figure 2.7: Sample of pictures that were not considered in the ground truth of dev topic 4 but should have been.

It is clear that the pictures shown in Figure 2.6 should not belong in the ground truth of the dev topic 4 since none of them are representative of an interview going on. The first picture

shows the user on the phone, the second shows two people talking and the last picture is just a door. Finally, the images presented in Figure 2.7 should belong to the moment described in the dev topic 4 since all of them show a camera pointed at the user, however they are not. This situation is important to have in account since the system can be retrieving images that should belong to the topic but the evaluation process just considers them wrong for not being in the ground truth. In short, this can have an impact not only in the dev topics but also in the test topics evaluation, since the ground truth is incomplete and does not present all the possible images related to a moment.

CHAPTER 3

Information Extraction From Images

The task of automatically recognizing and locating objects in images and videos is extremely important for computers to be able to understand and interact with their surroundings. Some major applications of this particular task of object detection are pedestrian detection, surveillance, autonomous driving, text digitalization, face detection and recognition, robotics, object counting and so on [7].

For this work the purpose of using object detection technologies is to convert images into text in order to allow the computer to compare an image with a moment described in text.

Feature extraction plays an important role in image classification and object detection systems which are two core components of computer vision. To put in simple terms, feature extraction is the first step in converting an image into text [8]. Figure 3.1 shows an example of feature extraction. However, the computer is only capable of learning that the extracted features represent a motorcycle (in the case of the example shown below) after classifying the extracted data, which is done by recurring to deep learning.

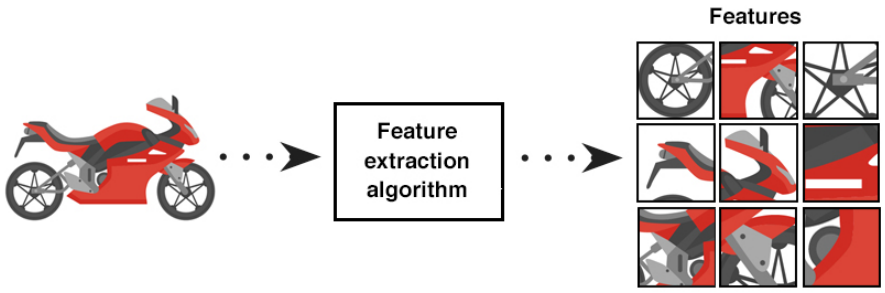


Figure 3.1: Feature extraction from an image [9].

This chapter starts with fundamental concepts in Section 3.1. Section 3.2 introduces the important subjects of machine learning, deep learning and neural networks and computer vision. Subsequently, in Section 3.3 a few examples of some of datasets used to train neural networks are presented. Afterwards, Section 3.4 gives a brief introduction of the most common computer vision libraries. Finally, a state of the art in object detection and image classification can be read in Section 3.5.

3.1 | Concepts of Label Extraction

- **Features and Feature Space**

A feature is considered to be a measurable piece of data in the image which is unique to a specific object, it can be color, texture or shape. Usually this features are extracted from the image and used in order to represent an object. Color is the most straightforward visual feature for indexing and image retrieval, while shape representation is the most difficult. This is because a 3-D real world object is represented in a 2-D plane in an image, which means that one dimension of information is completely lost. Texture features are very important in pattern recognition and is an important cue in region based segmentation of images.

The similarity between images can be determined through features which are represented as a vector.

To sum things up, feature space is a collection of features related to some properties of the object, while a feature is an individual measurable characteristics of the object [8].

- **Objects**

An object is used to identify specific items in an image or a video. It is possible to label multiple objects in an image. An example of objects in an image of a car might be wheels, headlights, etc. Usually an object is represented by a group of features in form of a feature vector that is used to recognize objects and classify them [8].

In object detection, small objects are normally the ones that give worst results and lower performance when being detected. This happens because the information available to detect them is more compressed and hard to decode without some prior knowledge or context [7].

- **Image Annotation and Classification**

Image classification is the process of associating an entire image with just one label. A simple example of image classification is labeling types of animals, cars or plants [10].

Image annotation, one of the most important tasks in computer vision, is the process of annotating an image with labels. These labels are predetermined in order to give the computer vision model information about what is shown in the image, they are a combination of a bounding box in specific coordinates of the image and a description of the object inside of it [11].

Feeding this kind of annotated image data to a computer model teaches it to recognize the visual characteristics of that specific label, this makes the model able to categorize new unannotated images of the same type of that label. Some of these models are presented in Appendix A.

- **Object Detection, Segmentation and Recognition**

Object detection is the name given to the process that combines image classification with object localization [12]. As previously explained, image classification is the prediction and assignment of a class label to an image, while object localization is the prediction and drawing of a bounding box around one or more objects in the image. In other words, object detection is the task that deals with the detection of objects of a certain class (e.g “flower”, “table”, “plane”) in images, making it a natural extension of the classification problem.

The object detection task is considered to be a supervised learning problem, since the objective is to design an algorithm which can accurately locate and correctly classify as many instances of objects as possible, in a bounding box, while avoiding false detections in a given set of training images. As an added challenge, many object detection applications require the problem to be solved in real time, which can be achieved. However, in order for a detector to be faster, accuracy is usually reduced.

Finally, object segmentation is the task of grouping pixels from the same object into a single region and object recognition is usually defined as giving the name of the category of an object that is contained in an image or a bounding box, assuming there is only one object in the image. For some authors object recognition can also involve detecting all objects in an image [7].

A survery on classification and regression based algorithms for object detection can be read in Appendix A.

- **Image Description**

Image description is the meaning of an image and humans can understand it with relative ease. However computers only see the digital representation of images, only detecting pixels, and therefore they are not able to recognize the semantic of the image. This problem makes the semantic gap the main challenge in computer vision [13]. This gap is defined by the lack of coincidence between the information extracted from visual data and the interpretation in a given situation [7].

As an example, Figure 3.2 shows an image of a family having a picnic. Feeding this image to a computer will output very different results from what a human would say.

Figure 3.2: Generic picture of a family having a picnic.

Using google cloud vision API [14] to extract information from the image, the following data is what the computer outputs:

- **Objects**: Person - 89%, Person - 86%, Person - 82%, Tableware -59%, Tableware - 55%, Package goods - 54%, Package goods - 50%.
- **Labels**: Picnic - 93%, Recreation - 86%, Sharing - 82%, Event - 74%, Summer - 70%, Child - 61%, Play - 57%, Family - 52%, Lunch - 52%.

However, the human output would be:

- **Sentence**: A family having a picnic in the park.

In the given example, the computer is only capable of outputting the objects and labels detected but is incapable of giving them any sort of meaning like a human can. However, recurring to recent image caption systems the computer might achieve some degree of similarity to a human in terms of describing an image.

3.2 | Teaching a Computer on How to Learn

Artificial Intelligence (AI) is the artificial simulation of human intelligence by a computer system in a way that it can perceive its environment, understand its behaviors and take action. Two important areas of AI are machine learning and deep learning [15].

3.2.1 | Machine Learning

Machine learning can be defined as a data analytics technique that allows computers to learn from experience. There are two types of machine learning techniques, which are supervised learning and unsupervised learning.

Normally, supervised machine learning is used to train a model to predict future outputs, this is done by inputting and outputting known data. Supervised learning uses two different techniques which are classification and regression. Classification techniques are used to classify input data into categories while regression techniques are used to predict continuous responses.

Unsupervised learning is mostly used to find hidden patterns or intrinsic structures in input data. The most common unsupervised learning technique is clustering which is used for data analysis exploration, in order to find hidden patterns or groupings in data [16].

3.2.2 | Deep Learning

Deep Learning is a subset of Machine Learning that is inspired by the structure and function of the human brain. In order to achieve this, deep learning resorts to artificial neural networks (ANNs).

The idea behind an ANN is that it tries to replicate the working of the human brain in the processing of data and creation of patterns, which is important for decision making. These ANNs are capable of learning unsupervised data that can either be unstructured, unlabeled or both. In short, deep learning is a machine learning technique that teaches computers to learn by example, like a human would [17].

Thanks to the new digital era, there has been an exponential increase in all forms of data from every region of the planet. This data is defined as "big data" and comes from sources like social media, search engines, live streaming services and many others. Even though all of this information is easily accessible, it is unstructured. The problem with unstructured data is that the human brain cannot comprehend it efficiently enough to extract relevant information. However, using deep learning, all of this unstructured data can be usable.

A computer model learns how to perform classification tasks directly from data, being it text, images or sound. Current deep learning models are able to achieve such levels of accuracy that they can outperform humans.

In deep learning, models are trained with large sets of labeled data and neural network architectures that contain several layers. This is one of the disadvantages of deep learning, in order to improve the results of an ANN it requires to be trained with large amounts of labeled data. Since deep learning deals with such great volumes of information, this introduces another

disadvantage to deep learning, which is the extreme need of higher and higher computing power.

The term "deep" comes from the usage of an extensive quantity of hidden layers in the neural network. A normal neural network usually contains 2-3 hidden layers where as a deep neural network can go up to 150 hidden layers or more.

As explained previously, deep learning models are trained by the usage of large sets of labeled data and neural network architectures that are capable of automatically extracting features from the data, without the need for manual feature extraction. This automated feature extraction makes deep learning models highly accurate and practical for computer vision tasks such as object classification.

Deep Learning also offers "end-to-end learning", this means that a network can learn how to automatically classify raw data. In addition, deep learning algorithms scale with data, where as machine learning methods bottleneck at a certain level of performance when more examples and training data are added, which gives deep learning networks a key advantage since they improve as the size of the data increases [17].

The main purpose of deep learning, for this work, will be to apply it to the images provided by the imageCLEF challenge.

3.2.3 | Neural Networks

A neural network can be considered a computer program that operates identically to how a human brain would, in the sense that it is able to be teachable to do certain tasks like problem-solving. The appeal of a neural network is the ability to emulate the human brain in pattern recognition skills.

Neural networks are composed of many small cells called neurons. These neurons are grouped into several layers that form columns. The connection between columns are formed also through their neurons. Each neuron of each layer is connected to another neuron of another layer. A visual representation of a generic neural network architecture is shown in Figure 3.3. For further readings on neural network architectures Appendix A describes most of the common architectures.

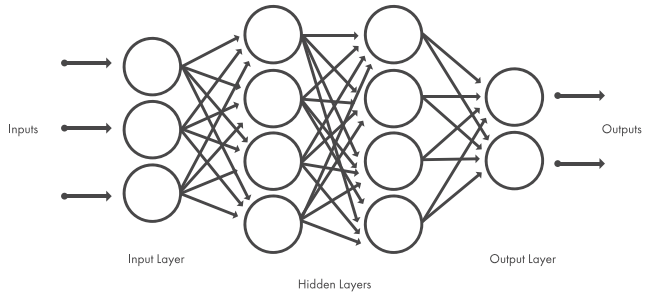


Figure 3.3: Typical neural network architecture [18].

The connections between layers are called weighted connections and they are adjusted with a real-valued number attached to them. This number is important, each neuron takes the value of the attached neuron (in their layer) and multiplies it by their connection weight. The bias value is an additional parameter in the neural network which is used to adjust the output along with weighted sum of the inputs to the neuron. The sum of the bias value with the weights is put through an activation function which mathematically transforms the value

and assigns it to the connected neuron in the adjacent layer. This is propagated through the whole network. See Figure 3.4 for a clear representation of this process.

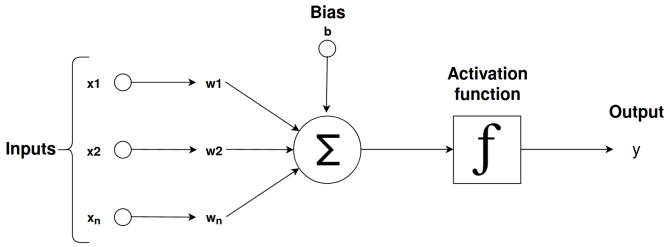


Figure 3.4: Operations done by a neuron [19].

To put it simply, a neural network can be compared to a filter that goes through all of the possibilities, so that the computer is able to come up with the correct answer.

Sometimes, an object might be too similar to another object which can make the network output a wrong answer. The solution to this problem is the usage of a back-propagation algorithm. This algorithm allows the network to adjust the connections back through the network, check if all the bias values are correct and all of the connections are weighted properly [20].

3.2.4 | Neural Network Training

The best way to train a neural network from scratch is to design a network architecture that will learn through the feeding of a large dataset of labeled data. This allows it to learn the features in order to create a model. The problem with this is that depending on the learning rate of the network and the amount of data, these networks can take a lot of time to train (days, maybe weeks).

To solve the problem of time, deep learning applications can recur to the usage of transfer learning. Transfer learning is a process that involves the fine-tuning of a pre trained model. This works by using a pre trained model, for example one trained with an existing network architecture like GoogLeNet, and feed it new data of previously unknown classes to the model. After some tweaks, the model will be able to categorize only a specific object instead of many different ones. This not only allows the model to be more precise in categorizing that one specific object, but it will also save lot of computation time [17].

3.2.5 | Learning From Images

Computer vision is a field of artificial intelligence and computer science that focuses on giving computers a visual understanding of the world [21] [22]. This is done by enabling computers to process and identify objects in a way similar to humans. This processing is done at a pixel level. Thanks to the recent innovations in deep learning and neural networks, computers are able to accurately identify, classify and even react in real time to objects and their surroundings.

3.3 | Datasets With Common Objects

A common objects dataset is a collection of images and videos that contain every day life objects that are manually labeled. State-of-the-art object detection models require deep neural networks. Training datasets are used in order to train these neural networks, as as previously explained.

A few examples of some available datasets are: MS COCO [23], ImageNet [24] , VisualGenome [25], OpenImages [26] and Pascal-VOC [27].

Some of these datasets propose challenges, where teams are able to compete in order to achieve state-of-the-art results. This subject is discussed in Section 3.5.

3.4 | Computer Vision Libraries

- **OpenCV**

OpenCV is an open source computer vision and machine learning software library originally developed by Intel in the year 2000 [28].

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc [29].

OpenCV Supports the deep learning frameworks like Tensorflow, Torch/PyTorch, Caffe and it is the most standardized tooling for computer vision.

- **Tensorflow**

Tensorflow is currently the most popular open source framework for numerical computation and large-scale machine learning introduced by google and was originally created for tasks with heavy numerical computations [30] [31].

Tensorflow is written in c++ which enables extremely fast compile times, non the less, it can still be accessed by other languages, such as Python and also supports CPUs, GPUs and distributed processing.

The name given to tensorflow comes from the inputs, since it receives inputs as a multi-dimensional array, also known as tensors. The input (tensor) goes on one end and then it "flows" throughout a system of operations and comes out on the other end as output.

Tensorboard is a feature of tensorflow that allows the monitoring of what tensorflow is doing graphical and visually.

- **VLFeat**

The VLFeat open source library implements popular computer vision algorithms specializing in image understanding and local features extraction and matching. Algorithms include Fisher Vector, VLAD, SIFT, MSER, k-means, hierarchical k-means, agglomerative information bottleneck, SLIC superpixels, quick shift superpixels, large scale SVM training, and many

others. It is written in C for efficiency and compatibility, with interfaces in MATLAB for ease of use, and detailed documentation throughout. It supports Windows, Mac OS X, and Linux [32].

- **BoofCV**

BoofCV is an open source library written from scratch for real-time computer vision. Its functionality covers a range of subjects, low-level image processing, camera calibration, feature detection/tracking, structure-from-motion, fiducial detection, and recognition.

This library is organized into several packages: image processing, features, geometric vision, calibration, recognition,visualize, and IO. Image processing contains commonly used image processing functions which operate directly on pixels. Features contains feature extraction algorithms for use in higher level operations.

Calibration has routines for determining the camera's intrinsic and extrinsic parameters. Recognition is for recognition and tracking complex visual objects. Geometric vision is composed of routines for processing extracted image features using 2D and 3D geometry. Visualize has routines for rendering and displaying extracted features. IO has input and output routines for different data structures [33].

- **GluonCV**

GluonCV is a toolkit that offers pre trained models, performance metrics of the different available models, consistent interface for when switching between the models, regular retraining and continuous integration to ensure code correctness, detailed documentation and well-documented examples. It also supports a range of different applications like : image classification, object detection, semantic segmentation, instance segmentation, pose estimation, video action recognition, depth prediction and a few others.

In short, gluonCV provides implementations of state-of-the-art deep learning algorithms in computer vision. It aims to help engineers, researchers, and students quickly prototype products, validate new ideas and learn computer vision [34].

3.5 | Recent Innovations and Improvements

Image classification and object detection are both subjects that are constantly innovating and improving upon previously results, every month new papers are published with new and more efficient networks.

In Figures 3.5 and 3.6 it is shown not only the current best methods for both image classification and object detection but also the development of the state of the art throughout the years.

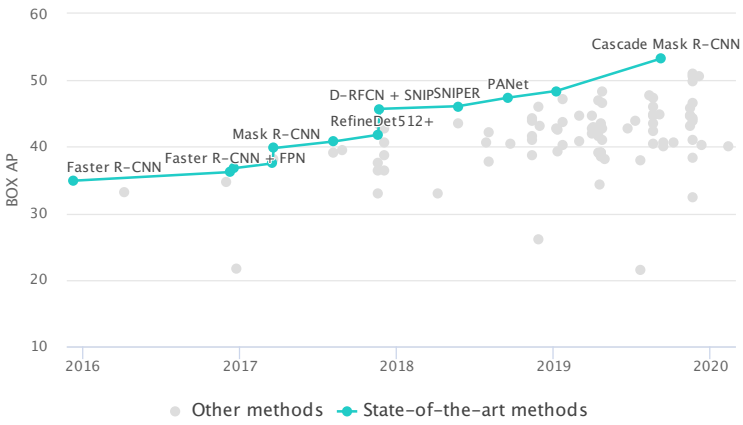


Figure 3.5: Object Detection COCO test-dev benchmark [35].

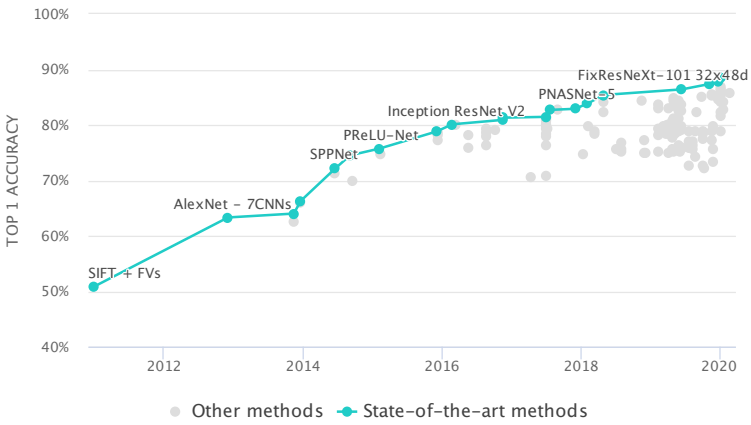


Figure 3.6: Image Classification ImageNet benchmark [36].

Due to the fact that object detection is a subject of great innovation, there is an extreme amount of papers that try to compete for the best results coming out every few months. So, in order to do a review of the state of the art, the Tables 3.1 and 3.2 show the benchmarks for both imageNet and COCO test-dev. These tables were obtained from [2] and are based on the analysis of [36] and [35] which is a website dedicated to the current state of the art for object detection and image classification.

3.5.1 | COCO Test-Dev

The COCO benchmark [23] is a dataset that places object recognition in the context of scene understanding. The evaluation metric used is the average precision (AP). Table 3.1 shows the current best architectures and their respective score for the COCO test-dev dataset.

Table 3.1: COCO Test-Dev Benchmarks.

Method	Backbone	AP (%)
Liu et al.(2019) [37]	ResNeXt-152	53.3
Tan et al. (2019) [38]	EfficientNet	51.0
Zhang et al. (2019) [39]	ResNeXt-101	50.7
Girshick et al. (2018) [40]	ResNeXt-152	50.2
Li et al. (2019) [41]	ResNet-101	48.4
Zhang et al. (2019) [39]	ResNet-101	46.3
Mahajan et al. (2018) [42]	ResNeXt	45.2
Zhao et al. (2019) [43]	VGG16	44.2
Cai et al. (2018) [44]	ResNet-101	42.8
Wang et al. (2019) [45]	ResNet-50	39.8
Lin et al. (2017) [46]	ResNet-101	39.1
Shrivastava et al. (2016) [47]	Inception-ResNet-v2	36.8
Kim et al. (2018) [48]	VGG-16	35.2

Liu et al. [37] achieved the best score in the COCO Test-Dev in 2019. They proposed better detection performance by creating a more powerful backbone network from previously existing backbones like ResNet [49] and ResNetXt [50]. They implemented a strategy for assembling multiple identical backbones (called Assistant Backbones and Lead Backbones) linked by composite connections between the adjacent backbones in order to form a more powerful backbone which was given the name of Composite Backbone Network (CBNet).

In typical CNN based detectors, the backbone network (the baseline of a network architecture) is used for basic feature extraction.

CBNet feeds the output features of the previous backbone as an input feature to the succeeding backbone through composite connections. At the final stage, the Lead Backbone outputs features for object detection.

This architecture was able to achieve the best result in the COCO Test-Dev with a 53.3% AP with single model by integrating a CBNet using triple ResNeXt-152 [50] backbones into the Cascade Mask R-CNN baseline.

Figure 3.7 presents the architecture for CBNet.

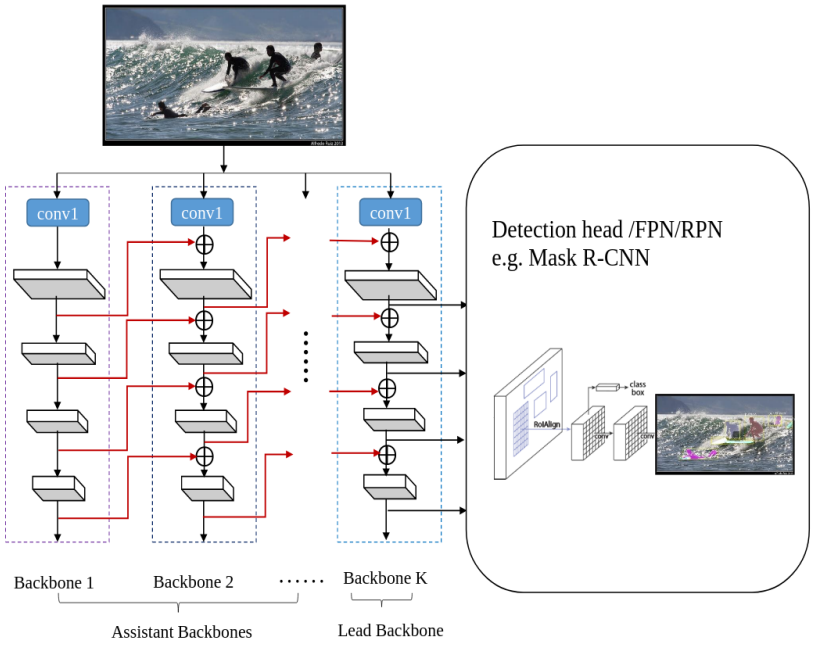


Figure 3.7: CBNet Architecture for object detection [37].

- **ResNeXt**

ResNeXt, also known as Aggregated Residual Transform Network was created by facebook researchers and it is a simple highly modularized network architecture for image classification.

The network is constructed by repeating a building block that aggregates a set of transformations with the same topology. The simple design results in a homogeneous, multi-branch architecture that has only a few hyper-parameters to set. This strategy creates a new dimension, which was given the name of "cardinality" (size of the set of transformations).

This architecture is an improvement over the Inception architectures, being more simple in design and adding more branches (towers) within modules [50].

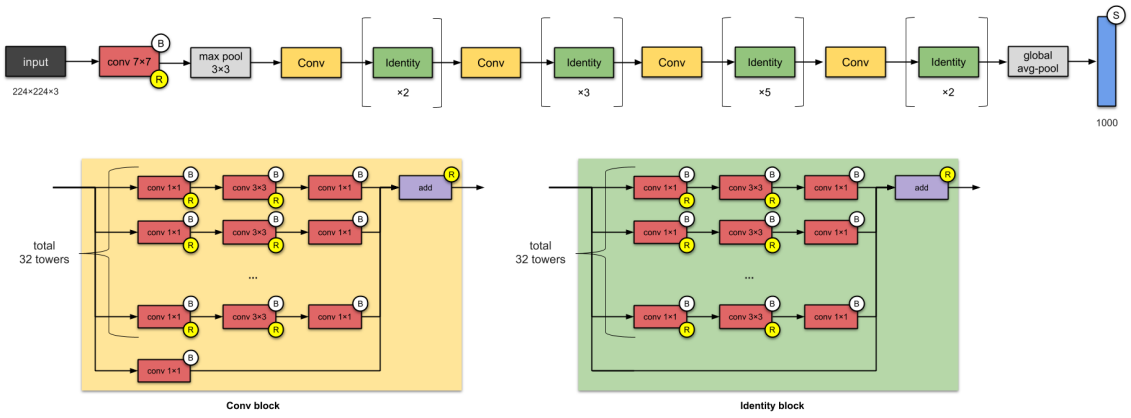


Figure 3.8: ResNeXt architecture [51].

3.5.2 | ImageNet

The imageNet Large Scale Visual Recognition challenge [52] is a benchmark for object category classification and detection. The evaluation metrics used are top-1 and top-5 accuracy.

Table 3.2: ImageNet Benchmarks.

Method	Backbone	Top-1 Acc (%)
Xie et al. (2019) [53]	EfficientNet	88.4
Kolesnikov et al. (2019) [54]	ResNet-152	87.8
Touvron et al. (2019) [55]	ResNeXt-101	86.4
Xie et al. (2019) [56]	EfficientNet	85.5
Mahajan et al. (2018) [42]	ResNeXt	85.4
Tan et al. (2019) [57]	EfficientNet	84.4
Touvron et al. (2019) [55]	ResNet-50	82.5
Szegedy et al. (2017) [58]	Inception-resnet-v2	80.1
Szegedy et al. (2017) [58]	Inception-v4	80.0
Simonyan et al. (2014) [59]	VGG-16	74.4

Xie et al. [53] stated that current state-of-the-art vision models are still trained with supervised learning, which implies the necessity of large corpus of labeled images in order to work properly. The fact that current models are only shown labeled images causes an obvious limitations in the improvement of accuracy and robustness of current state-of-the-art models, this can be improved with the usage of the large available quantities of unlabeled images available.

Having this in mind, they decided to use unlabeled images to improve the state-of-the-art ImageNet accuracy and show that accuracy has an outsized impact on robustness. For this purpose, they used a much larger corpus of unlabeled images, where a large fraction of images did not belong to ImageNet training set distribution.

Using a self-training framework the model was trained with 3 main steps which consist in:

1. Training of a teacher model on labeled images.
2. Usage of the teacher to generate pseudo labels on unlabeled images.
3. Train a student model on the combination of labeled images.

The algorithm was iterated a few times by treating the student as a teacher to relabel the unlabeled data and training a new student.

An important discovery was made during the training of the algorithm. For the method to work well at scale, the student model should be noised during its training while the teacher should not be noised during the generation of pseudo labels. This way, the pseudo labels are as accurate as possible and the noised student is forced to learn harder from the pseudo labels. To induce noise in the model it was used RandAugment data, dropout and stochastic depth

during the training. Figure 3.9 shows a brief view of how the method works. This is where the name of the method "Noisy Student"comes from, since the student is noised to learn beyond the teacher's knowledge. With this method they were able to show that it is possible to use unlabeled images to significantly advance both accuracy and robustness of state-of-the-art imageNet models.

The presented model uses EfficientNet as a backbone trained on images from imageNet dataset and was able to obtain the best results in the ImageNet benchmark dataset by achieving an accuracy of 88.4%.

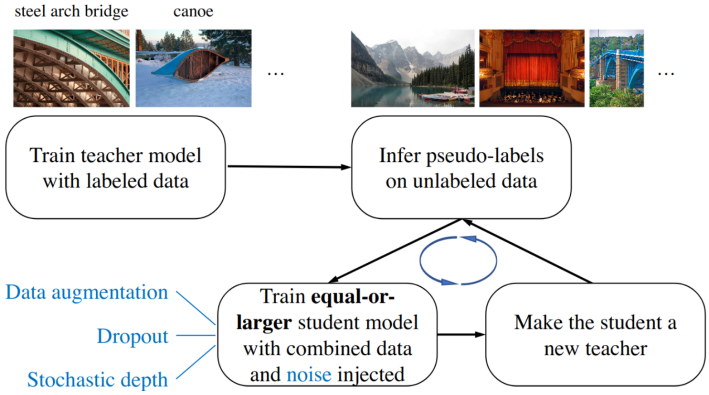


Figure 3.9: Noisy Student Method [53].

Researchers at Google decided to study the impact of scaling up CNNs (shown in Figure 3.10), in order to achieve better accuracy and efficiency. EfficientNet-B0 was developed based on a simple idea, scaling each of the dimensions of the network (width, depth and resolution) with a constant ratio, improves the overall performance [57].

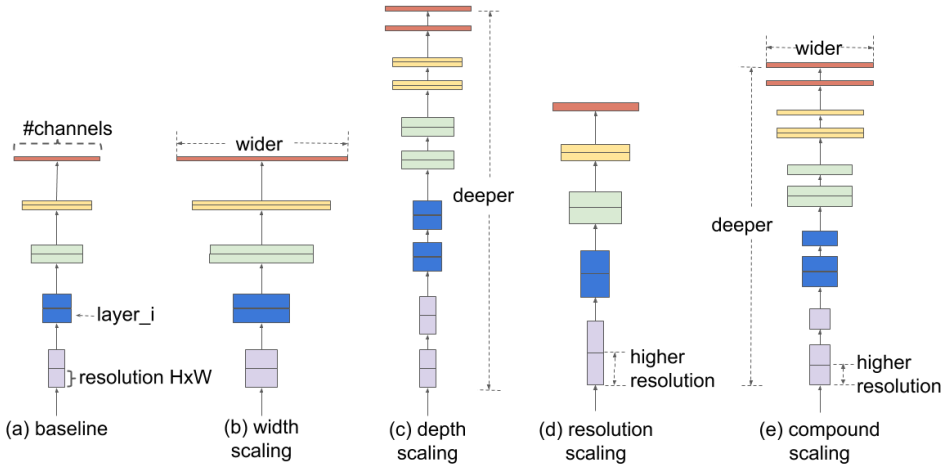


Figure 3.10: Comparison of different scaling methods: (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio [57].

The baseline network architecture, EffecientNet-B0, uses mobile inverted bottleneck convolution (MBConv), similar to MobileNetV2 [60] and MnasNet [61]. Figure 3.11 shows the baseline network architecture EfficientNet-B0.

Figure 3.11: EfficientNet-B0 architecture representation [62].

3.6 | Final Remarks

The models and algorithms experimented in this thesis were provided trough a computer vision python library called imageAI, which allows the ability to easily use state-of-the-art AI features [63]. This library required the pre-installation of TensorFlow, OpenCV and Keras libraries. The Keras library [64] is an open-source neural network library written in Python which is capable of running on top of TensorFlow.

During the development of the automatic retrieval system, several of the described algorithms and models in Appendix A were tested. For image classification the algorithms analysed were DenseNet, InceptionV3, ResNet and SqueezeNet. For object detection the models used were YoloV3, TinyYoloV3, RetinaNet and ResNetXt-101. The testing of all of these models was done in order to find the most suitable model to process the imageCLEF images dataset.

CHAPTER 4

Information Extraction From Text

As previously discussed, the problem of big data has become more relevant in the recent years. This problem does not apply only to the ever growing amount of multimedia data created by smartphones but also to the growing presence of information in the form of news, corporate files, medical records, government documents, court hearing and social media. There is an ever increasing flood of information in an unstructured form. Natural Language Processing (NLP) is related to the usage of computation methods to process such data as a mode of communication used by humans.

"There are lot many processes involved in the pipeline of NLP. At the syntactic level, statements are segmented into words, punctuation (i.e. tokens) and each token is assigned with its label in the form of noun, verb, adjective, adverb and so on (Part of Speech Tagging). At the semantic level, each word is analyzed to get the meaningful representation of the sentence. Hence, the basic task of NLP is to process the unstructured text and to produce a representation of its meaning." [65].

Information Extraction (IE) from text is the process of extracting useful information from textual sources by implementing techniques of NLP. It can be defined as the act of efficiently and effectively analyze text and extract valuable and relevant knowledge from it in the form of structured information. "The goal of IE is to extract salient facts about pre-specified types of events, entities, or relationships, in order to build more meaningful, rich representations of their semantic content, which can be used to populate databases that provide more structured input." [65].

In this thesis, NLP is implemented to serve the purpose of creating an algorithm capable of extracting relevant information from a textual source. Furthermore, the algorithm must also be able to categorize the extracted data according pre-specified categories such as "locations" and "activities". Additionally, NLP is also used for the computation of similarity values between extracted textual data and extracted visual labels from images.

This chapter starts with an introduction to Natural Language Processing in Section 4.1. Section 4.2 explains the process of representing text in a numerical vector form while describing the concept of word embeddings. Static and contextualized word embedding models are introduced in Section 4.3 and 4.4 respectively. An overview on some of the available NLP libraries is presented in Section 4.5. Finally, Section 4.6 gives the final remarks of this chapter.

4.1 | Natural Language Processing

Natural language processing is a subfield of linguistics, computer science, information engineering and artificial intelligence, which is devoted to the engineering of computational models and processes to give the ability of human-like comprehension of texts/languages to computers [66].

Human language is extremely complex and rarely precise. In order to understand it, not only it is required to comprehend what words alone mean, but also how linking them together creates meaning. The nature of the human language makes NLP one of the most difficult tasks in computer science.

Figure 4.1 shows the classification of NLP, which consists in two major components, Natural Language Understanding (NLU) and Natural Language Generation (NLG) [66].

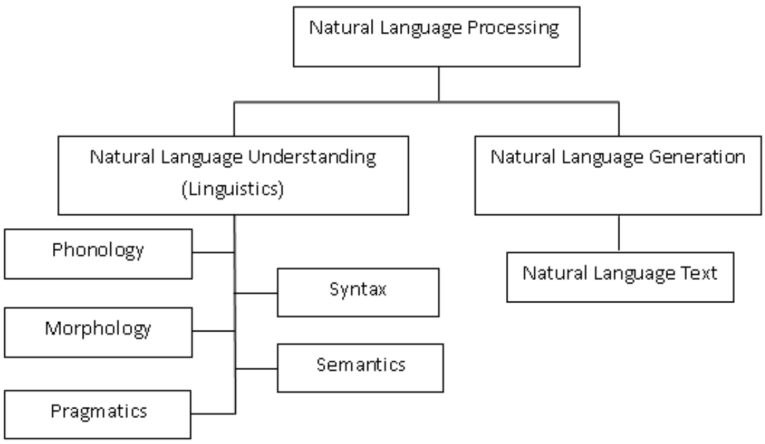


Figure 4.1: Classification of NLP [66].

Natural Language Understanding is the process of understanding text. It is related to the science of Linguistic that studies the meaning of languages, context and various forms of language.

Natural Language Generation is the process of generating text, sentences and paragraphs that are meaningful from an internal representation [66].

Using the visual representation of Figure 4.1 the important terminologies of NLP are as follows [66]:

- **Phonology**: The part of Linguistics which refers to the systematic arrangement of sound;
- **Morphology**: In linguistics, morphology is the study of words, how they are formed, and their relationship to other words in the same language. The different parts of the word represent the smallest units of meaning known as Morphemes;
- **Lexical**: In Lexical the focus is the interpretation of the meaning of individual words;
- **Syntax**: Syntax refers to the study of the grammatical structure of the sentence;
- **Semantic**: Semantic processing determines the possible meanings of a sentence by pivoting on the interactions among word-level meanings in the sentence;

- **Discourse**: Discourse focuses on the properties of the text as a whole that convey meaning by making connections between component sentences;
- **Pragmatic**: Subfield of linguistics that studies the ways in which the context of a sentence contributes to the meaning.

The field of NLP can be divided in two broad sub-areas: core areas and application areas. The core areas address fundamental problems such as language modeling, morphological processing, parsing and semantic processing [67].

- Language modeling is considered the most important task in NLP and an essential piece in any application of NLP. It is the process of creating a model capable of predicting words or simple linguistic components given previous words or components. It can capture syntactic and semantic relationships among words or components in a linear neighborhood, making it useful for tasks such as machine translation and text summarization.
- Morphological processing is the process of finding segments within single words, including roots and stems, prefixes and suffixes.
- Parsing examines how different words and phrases relate to each other.
- Semantic processing is the task of understanding the meaning of words and phrases. This is done recurring to word embedding models, like Word2Vec. This will be further discussed in this chapter.

The application areas address topics such as extraction of useful information from text (e.g named entities and relations), translation of text, summarization of written documents, automatic answering of questions, chat bots, email spam detection and many others [67].

4.2 | Numerical Representation of Text

Machine learning algorithms and most of all deep learning architectures are incapable of processing strings of text, this is because they require numbers (vectors) as an input in order to perform linear algebra operations [68] which is not possible with words. A human can easily tell that the word "dog" and the word "cat" are identical, since they both represent an animal, however a computer would assume that they are completely different things since all the letters in those words are different.

4.2.1 | Word Embeddings

The dominant approach to solve this problem is the usage of word embeddings, which is a type of word representation that allows words with similar meaning to have a similar representation by mapping a set of words, or phrases in a vocabulary, to vectors of numerical values. For example, the word "happy" can be represented as a vector of 4 dimensions [0.24, 0.45, 0.11, 0.49] and "sad" as the following vector [0.88, 0.78, 0.45, 0.91]. The reason for this vectors to exist is so that a machine learning algorithm can perform linear algebra operations on numbers (vectors) instead of words [69]. Word embedding methods learn a real-valued vector representation for a predefined fixed size vocabulary from a corpus of text [70]. A

vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else.

As an example, the sentence "Word Embeddings are Word converted into numbers" can be converted to the following dictionary using the one-hot encoded vector representation : ["Word","Embeddings","are","Converted","Word","into","numbers"]. Using this representation the word "numbers" in the one-hot encoded vector is [0,0,0,0,0,1] and for the word "converted" is [0,0,0,1,0,0]. This is considered to be the most simple method to represent words in vector forms [68]. Figure 4.2 showcases the given example.

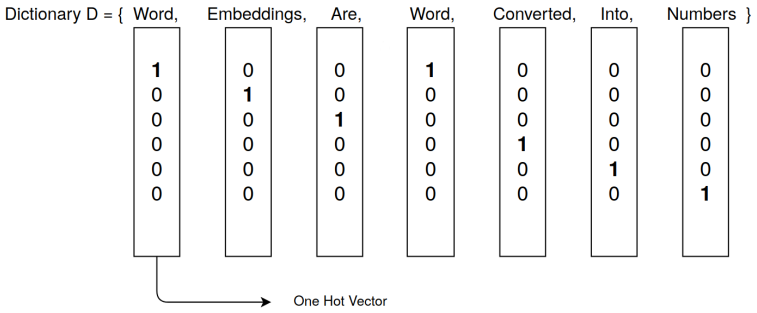


Figure 4.2: Example of text representation by one-hot vector.

4.3 | Static Word Embedding Models

This section introduces some common static word embedding models to learn word embeddings from text.

Static word embedding have the fundamental problem which is that they generate the same embedding, in different contexts, for the same word and once learned they do not change it. They map each word type to a single vector, making it harder to deal with the polysemy problem. This is due to the fact that each word has a single vector, regardless of context [71]. Therefore, these models assume that the meaning of any given word is the same across the entire text.

As an example, having the following two phrases:

- "I saw her at the library."
- "Pass me the saw to cut the log in half."

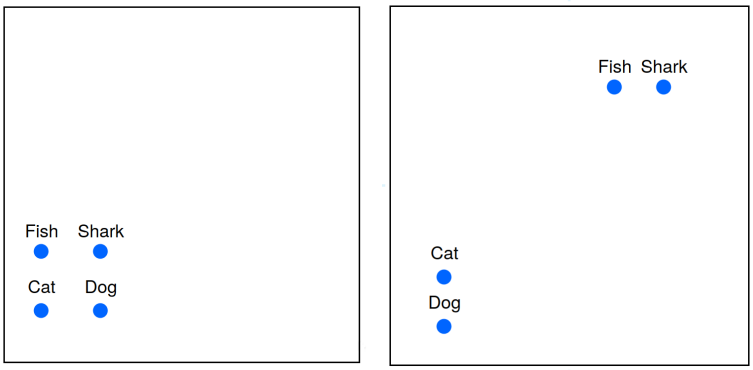
In this case, the word "saw" has two different meanings. In the first phrase the word "saw" refers to the verb "see" and for the second phrase it refers to the tool "saw". However, for static word embedding models, words only have one single meaning and therefore the word representation for "saw" would be the same in both cases.

Dynamic word embeddings models represent "saw" differently according to the contexts, while static embedding can not distinguish the semantic difference between two "saws" and represent with the same vector no matter the context [72] [73].

4.3.1 | Word2Vec

One of the most important and most popular models developed in NLP was Word2Vec. Created by Tomas Mikolov, et al. [71] at Google in 2013, Word2Vec is a two-layer neural network that processes text by "vectorizing" words with the purpose of grouping vectors of similar words together in vectorspace. These similarities are detected by creating vectors that are distributed numerical representations of word features, without human intervention. A vocabulary is outputted from Word2Vec where each item has a vector attached to it. This can be fed into a deep-learning network or queried to detect relationships between words, like similarities. The similarities are measured trough a cosine similarity, having no similarity is expressed as a cosine similarity of 0 since it is 90 degree angle, while a full similarity is expressed as cosine similarity of 1 and it is a 0 degree angle, complete overlap. Sweden is equal to Sweden therefore the cosine similarity is equal to 1, while Norway has a cosine distance of 0.760124 from Sweden [74].

In a regular one-hot encoded vector all words have the same distance between each other, even though their meanings are completely different, this creates a lost in information at the encoding. However, Word2Vec is capable of learning vectors by understanding the context in which words appear. This results in vectors in which words with similar meanings end up with a similar numerical representation in the vectorspace. Figure 4.3 illustrates this situation. For instance, cats and dogs are more similar than fish and sharks. This extra information is useful for machine learning algorithms [75].



(a) One-hot encoding resulting vector [75].

(b) Word2Vec encoding resulting vector [75].

Figure 4.3: One-hot encoding vs Word2Vec encoding.

Furthermore, using a word offset technique, Word2Vec is capable of performing simple algebraic operations to the word vectors. An example is that the vector("King") - vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word "Queen" [71].

Word2Vec is composed of two different models, CBOW (Continuous Bag of words) which predicts a word given the context and Skip-Gram which predicts context given a word [71] [74] as shown in Figure 4.4.

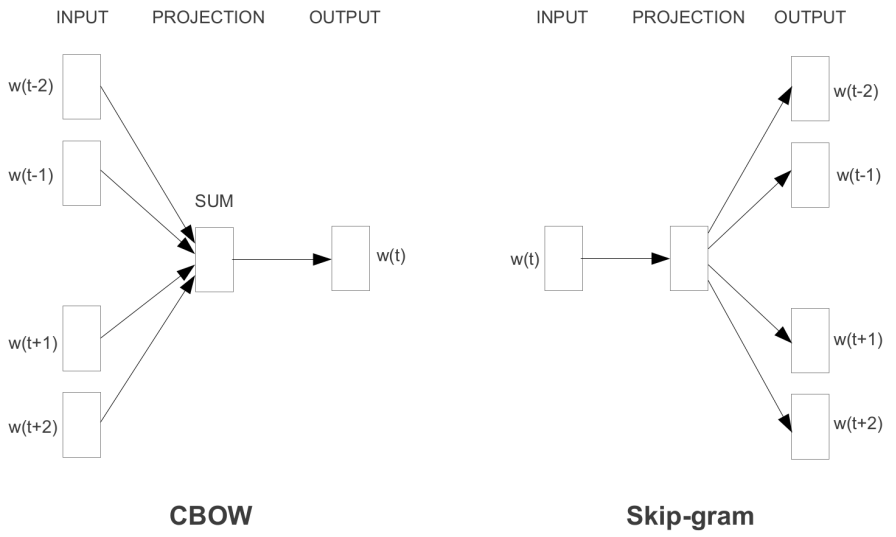


Figure 4.4: CBOW and Skip-gram models [71].

- **Continuous Bag of Words**

CBOW allows to take a big amount of text and train a neural network to predict a word by inputting the N words at each side. In the given example in Figure 4.4 N = 2. Using the example given in [76] : "The monkey is eating a banana", the word "eating" is predicted given that the previous two words are "The" and "monkey" and the next two are "a" and "banana".

Using Figure 4.4 again, the inputs of CBOW would be: w(t+2) = "monkey", w(t+1) = "is", w(t-1) = "a", w(t+2) = "banana" and the output (prediction) would be w(t) = "eating" [76].

- **Skip-gram**

Skip-gram is much identical to CBOW but instead of predicting a word given the context, it predicts the context given a word.

Once again recurring to Figure 4.4, the input of Skip-gram would be: w(t) = "eating" and the outputs would be : w(t+2) = "monkey", w(t+1) = "is", w(t-1) = "a", w(t+2) = "banana" [76].

4.3.2 | GloVe

GloVe stands for Global Vectors for Word Representation and was a new approach created by Pennington et all. in 2014 [77] to generate word embeddings with unsupervised learning. Glove main goals are to create word vectors that capture meaning in the vector space and to take advantage of global count statistics instead of using only local information.

The problem with Word2Vec is that it only takes local information into account, and does not consider global context. This means that the semantics learnt for a given word are only affected by the surrounding words.

GloVe works by aggregating global word-to-word co-occurrence matrix from a of corpus text. This means that if two words keep appearing together in a corpus of text they either

share a linguistic or a semantic similarity. Simply put, similar words will be placed together in the high-dimensional space. Therefore, GloVe can be seen like an extension to the Word2Vec model.

4.3.3 | FastText

FastText, created by Facebook's AI Research (FAIR) lab in 2016, is a fast text classifier based on the skipgram model used for efficient learning of word representations and sentence classification. Popular models like word2Vec and GloVe are based on continuous word representations that create vectors directly from words in a sentence while ignoring the morphology of words, this is done by assigning a distinct vector to each word, fastText uses a different approach treating each word as bag of characters n-grams. A vector representation is associated to each character n-gram and words are represented as the sum of these representations. This allows fastText to work with rare words not seen in the training data since the word is broken down into n-grams to get the corresponding embeddings [78].

Using the word "where" as an example and n=3, the representation of this word in a fastText model is <wh, whe, her, er, re> and the special sequence <where>. The angular brackets serve as boundary symbols to distinguish the n-gram of a word from the word itself, this means that if the word "her" was part of the vocabulary it would be represented as <her>, which allows the preservation of the meaning of shorter words and the understanding of suffixes and prefixes.

4.4 | Contextualized Word Embedding Models

Contextualized words embeddings aim at using different embeddings for different word contexts to address the issue of polysemous and the context-dependent nature of words [73]. Using the example given in Section 4.3, these models would be able to distinguish the different meaning of the word "saw" given the two different sentences.

4.4.1 | Context2vec

Context2Vec is an unsupervised model capable of learning efficiently generic context embedding of wide sentential contexts, using a bidirectional long short-term memory (LSTM) recurrent neural network.

A large plain text corpora is utilized in order to learn a neural model capable of embedding entire sentential contexts and target words in the same low-dimensional space, which is optimized to reflect inter-dependencies between targets and their entire sentential context as a whole.

In contrast to word2vec that uses context modeling mostly internally and considers the target word embeddings as their main output, the focus of context2vec is the context representation. Context2vec achieves this objective by assigning similar embeddings to sentential contexts and their associated target words [79].

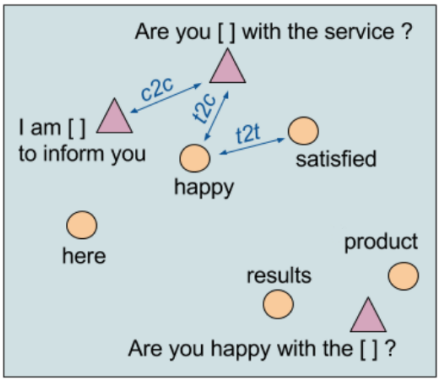


Figure 4.5: A 2D illustration of context2vec's embedded space and similarity metrics. Triangles and circles denote sentential context embeddings and target word embeddings, respectively [79].

Sentential Context	Closest target words
This [ ] is due	item, fact-sheet, offer, pack, card
This [ ] is due not just to mere luck	offer, suggestion, announcement, item, prize
This [ ] is due not just to mere luck, but to outstanding work and dedication	award, prize, turnabout, offer, gift
[ ] is due not just to mere luck, but to outstanding work and dedication	it, success, this, victory, prize-money

Figure 4.6: Closest target words to various sentential contexts, illustrating context2vec's sensitivity to long range dependencies, and both sides of the target word [79].

4.4.2 | ELMo

ELMo (Embeddings from Language Models) is a NLP model with context-aware representation, it understands different meanings for the same word since it takes into account the surrounding words unlike traditional word embedding models such as Word2Vec and GLoVe. In order to achieve this, ELMo attributes an embedding for each word after looking at the entire context in which it is used, instead of using fixed embeddings for each word. Therefore, the same word might have different word vectors under different contexts.

It models both syntax and semantics of word use and how these uses vary across linguistic context. The word vectors are learned through the usage of internal states of a deep bidirectional LSTM algorithm, trained on a large corpus of text. Bidirectional implies that the algorithm takes into account the words before and the words after it in both directions. LSTM (Long Short-Term Memory) is one type of neural network that is able to retain data in memory for long periods of time, allowing it to learn longer-term dependencies. This language model can predict both the next word and the previous word and it is a character based model allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens not presented during training [80].

Figure 4.7 presents an example of the differences between GLOVe that is a non-context aware model and ELMo biLM (bidirectional Language Model) that is context aware.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Figure 4.7: Nearest neighbors to "play" using GLoVe and context embeddings from a biLM [80].

GLoVe only uses the word "play" as source, therefore the obtained neighbors for that word are spread across several parts of speech however they all focus on the sports-related sense of the word "play". ELMo biLM uses the entire sentence as source, this means that it is able to understand the context of the word. Therefore, in both cases, the biLM is able to disambiguate both the part of speech and word sense in the source sentence [80].

4.5 | Available NLP libraries

There is a wide array of NLP tools and services available. Knowing their features is important in order to find the most appropriate one for the project at hands. Some might be better for smaller project and others better for experts working with big data projects. Furthermore, NLP libraries solve the problem of requiring superior knowledge of mathematics, machine learning, and linguistics. Using these tools, developers can simplify text processing so that they can concentrate on building machine learning models.

- **SpaCy**

SpaCy is a free, open-source library for advanced natural language processing written in Python and Cython published by Explosion AI. It was designed specifically for production use and to help in the building of applications that process and "understand" large volumes of text data. Some use cases for this specific library are to build information extraction or natural language understanding systems, or to pre-process text for deep learning [81]. Some of the features that SpaCy offers are:

- Tokenization: The segmentation of text into words, punctuation, etc

- Part-of-Speech Tagging: The assignment of word types to tokens, like verb, noun, etc

- Similarity: The comparison between different words, phrases or text documents and how similar they are.

- Lemmatization: The assignment of base forms of words.

- **Natural Language ToolKit**

Developed by Steven Bird, Edward Loper and Ewan Klein in the Department of Computer and Information Science at the University of Pennsylvania, NLTK (Natural Language ToolKit) is a suite of open source program modules, tutorials, problem sets and a leading platform for building Python programs to work with human language data. NLTK covers symbolic and statistical natural language processing, and is interfaced to annotated corpora. This library provides easy-to-use interfaces such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning [82].

- **Stanford Core NLP**

Developed at Stanford University, Core NLP is a library written in Java with wrappers for different languages, including Python. This library is fast and some of its components can be integrated to NLTK which boosts efficiency. CoreNLP enables users to derive linguistic annotations for text, including token and sentence boundaries, parts of speech, named entities, numeric and time values, dependency and constituency parses, coreference, sentiment, quote attributions, and relations [83].

- **Gensim**

Gensim ("Generate Similar") is a Natural Language Processing open-source library for unsupervised topic modeling (a technique to extract the underlying topics from large volumes of text) and for natural language processing. This python-cython library specializes in finding the semantic similarity between two documents through vector space modeling and topic modeling toolkit. It is capable of building document or word vectors, corpora, performing topic identification, performing document comparison (retrieving semantically similar documents) and analysing plain-text documents for semantic structure. In terms of producing word embedding, gensim allows for the usage of Word2Vec and fastText [84].

- **TextBlob**

TextBlob, also a Python library, offers an API for performing NLP tasks, like part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, language translation, word inflection, parsing, n-grams, and WordNet integration [85].

- **Flair**

Flair allows the usage of state-of-the-art NLP models for entity recognition (NER), part-of-speech tagging (PoS), sense disambiguation and classification [86].

- **Polyglot**

This python NLP package supports various multilingual applications and offers the following tasks language detection (196 languages), tokenization (165 languages), named entity recognition (40 Languages), part of speech tagging (16 languages), sentiment analysis (136 Languages) [87].

4.6 | Final Remarks

In this thesis the NLP library that was chosen for the development of the text processing phase of the automatic image retrieval system was SpaCy. This decision was based of the fact that there was some previous knowledge thanks to the work developed last year for the ImageCLEF challenge [4]. In addition, SpaCy also provides state-of-the-art NLP models, many useful features, it is easy to use and it offers a simple and well written documentation which makes it very beginner friendly.

CHAPTER 5

Proposed Approach

This chapter aims at making clear how the automatic image retrieval system was built and how it operates.

Initially in Section 5.1 the system workflow architecture is illustrated trough a diagram and a small description is presented. Section 5.2 describes some of the preliminary experiences done to image recognition algorithms and object detection models provided by the ImageAI library. Afterwards, Section 5.3 describes a very raw first attempt at an image retrieval system using only object detection. The implementation of the scene recognition algorithm is demonstrated in Section 5.4. Following up, in Section 5.5 it is presented a description of the differences between both submitted runs in the image processing stage. Subsequently, the text processing stage is introduced in Section 5.6, in this section there is an explanation on how the system was able to extract linguistic annotations, what kind of syntax and semantic rules were implemented in the algorithm and how the categorization and extraction of relevant words from text was done. Section 5.7 explains how the system is capable of automatically retrieving images for a given moment described in text, how it is able to compare the similarity of the mined text words with the visual concept words and all the calculations it does in order to compute weights and confidence scores. Lastly, Sections 5.8 and 5.9 explain both of the submitted runs for the automatic image retrieval system.

5.1 System Workflow Architecture Diagram

In the diagram presented in Figure 5.1 it is possible to see that the system starts by firstly processing all images from the dataset provided by the imageCLEF challenge with scene and object recognition algorithms. During the process the labels are categorized according pre-defined categories.

Subsequently, text is processed using an NLP model that extracts linguistic annotations from textual sources. With the implementation of syntax and semantic rules alongside with the provided linguistic annotations, relevant words are extracted and categorized in pre-defined categories.

Following up, when all of the textual and visual labels are extracted and correctly categorized the confidence score computation stage starts. This stage has several steps that can be summarized in two main steps: the computation of weights for each category and the calculation of similarity scores.

Finally, in the retrieval stage, the system checks for the 50 highest confidence scores for a given topic and stores it in a csv file. These 50 images are the ones retrieved for a given moment.

As an extra step, an evaluation stage was created to analyse system results for the dev

topics with the objective of emulating the evaluation methodology for the challenge.

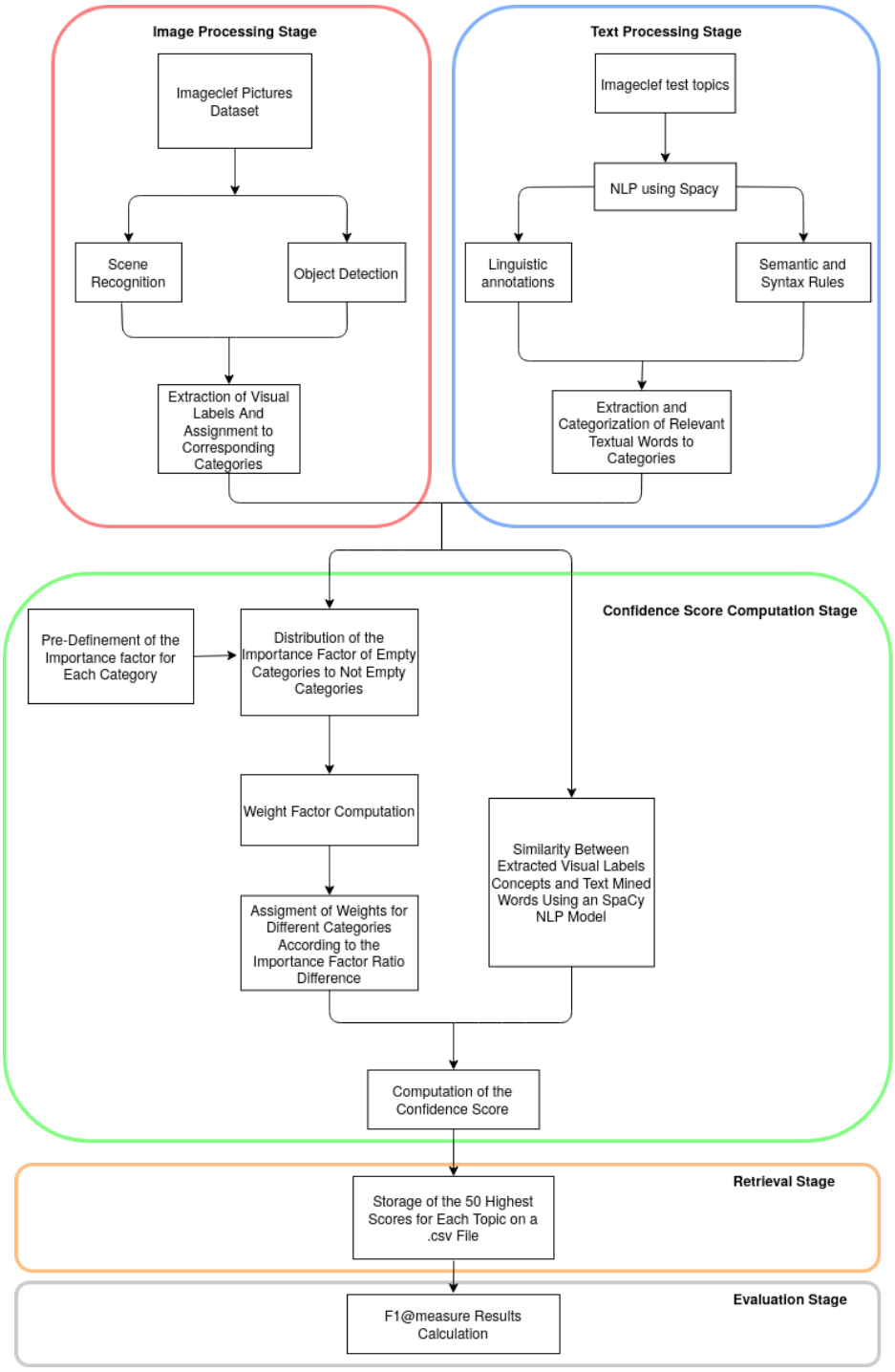


Figure 5.1: System Architecture.

5.2 | Preliminary Experiences

With the goal of comparing the different image recognition algorithms and object detection models a few test runs were made. The models and algorithms were provided trough a computer vision python library called imageAI, which allows the ability to easily use state-of-the-art AI features. It supports algorithms for image prediction, custom image prediction, object detection, video detection, video object tracking and image predictions trainings [63].

The test runs consist on feeding each of the models and algorithms with one picture manually chosen beforehand. Each model produces predictions on what the image represents or what is the objected detected. The prediction probability ranges in an interval between [0,100]. This prediction probability represents the certainty of the model or algorithm in the respective prediction.

Sections 5.2.1 and 5.2.2 provide examples of the performance test runs done to the models and algorithms.

5.2.1 | Image Recognition Preliminary Experiences

The imageAI library allows the usage of 4 image recognition algorithms for image recognition which are DenseNet, inceptionV3, ResNet50, and SquezeeNet that were pre-trained on the the ImageNet-1000 dataset. This means that they can predict/recognize 1000 different objects in any image or number of images. The experiences done to evaluate these algorithms consist in analysing one image and compare the results of the different algorithms with the goal of finding which one was the best performer and had the most accurate prediction with the highest prediction probability.

In order to given a few examples of these preliminary experiences, the images in Figure 5.2 were fully analysed with the image recognition algorithms.

(a) (b) (c)

Figure 5.2: Samples of pictures used for image recognition preliminary experiences.

- **Obtained performance graphs in the image recognition tests**

Figures 5.3, 5.4 and 5.5 show three examples of the performance achieved of these algorithms in the processing of the pictures shown in Figure 5.2.The graphs are structured in the following manner : the X axis represents the predictions, the Y axis represents the percentage probability certainty for the respective prediction and the color represents the algorithm used.

In the first example a picture of a dog (breed Saluki) is analysed. Figure 5.3 shows that InceptionV3 is the best performer in the first run, predicting correctly with an efficiency of 96.38% and out performing the other 3 neural networks by a large margin, being that the second best is the ResNet50 with an efficiency of 42.32%.

For the second example a picture of a car was processed. Figure 5.4 illustrates that InceptionV3 predicted with a 99.99% that the car was a minivan. The shown car is not a minivan but its similar to one, so it is possible to assume that the prediction is correct.

The final example a picture of an espresso coffee is analysed. In this example all image recognition algorithms predicted correctly that the image represents an espresso. However, SquezeeNet only achieved 63.88% efficiency while InceptionV3 predicted with 100.0% efficiency as shown in Figure 5.5.

From the 3 examples that were shown, it is clear that the inceptionV3 algorithm achieved the best results and out performed the other algorithms.

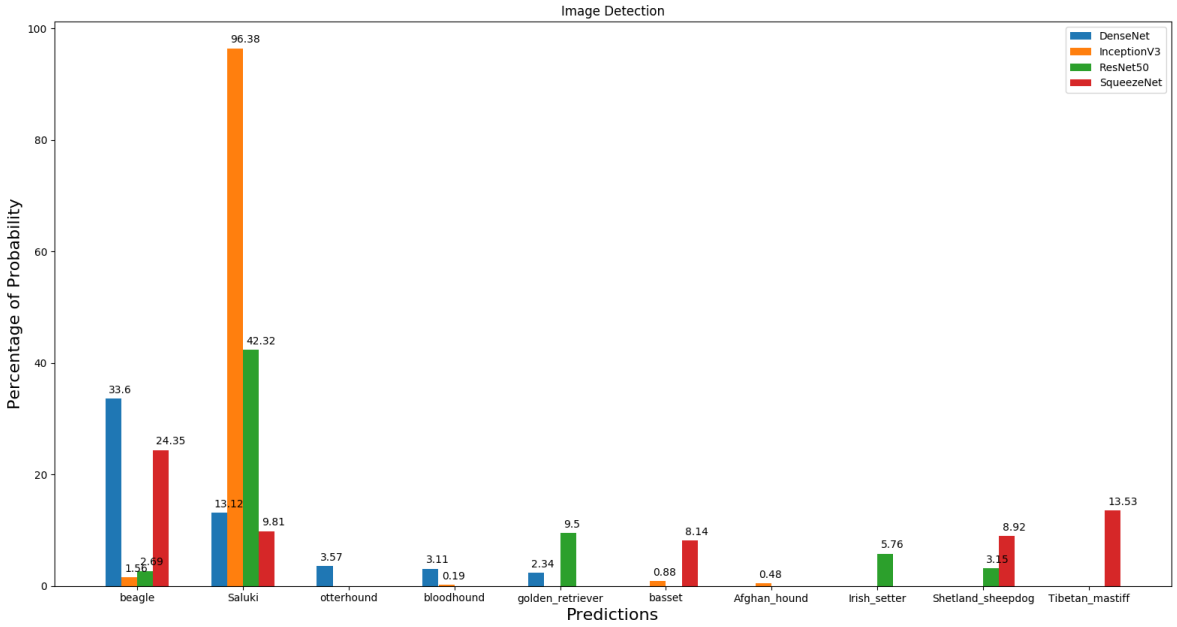


Figure 5.3: Performance achieved of different algorithms in the processing of Figure 5.2 a).

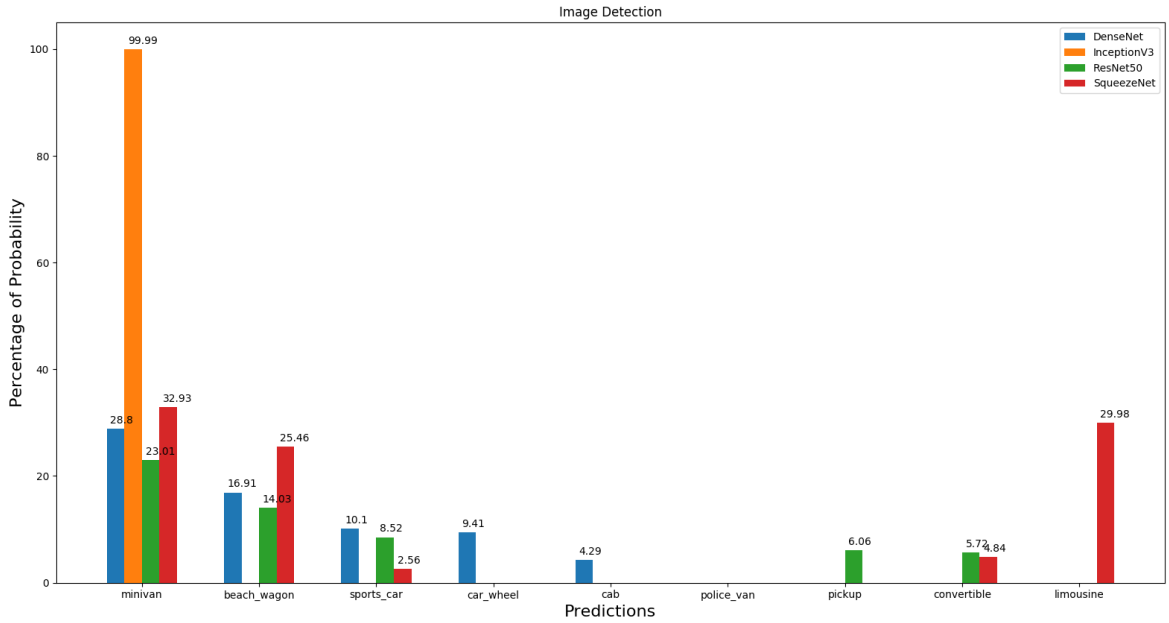


Figure 5.4: Performance achieved of different algorithms in the processing of Figure 5.2 b).

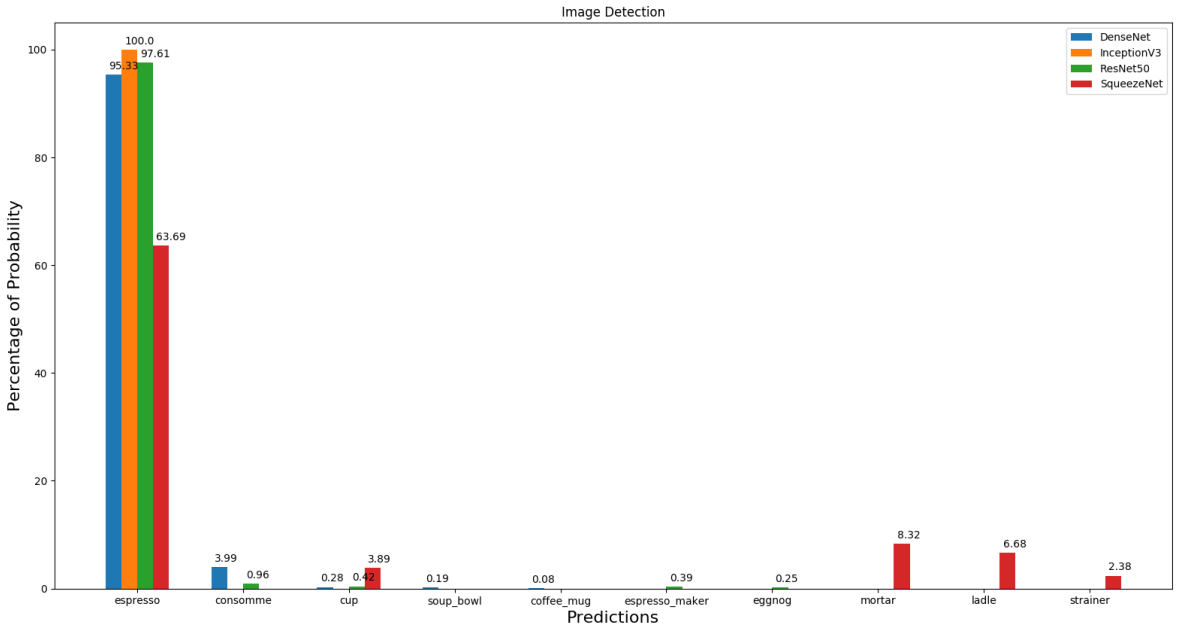


Figure 5.5: Performance achieved of different algorithms in the processing of Figure 5.2 c).

5.2.2 | Preliminary Experiences in Object Detection

ImageAI provides 3 different models trained on the COCO dataset for object detection that are able to identify up to 80 of the most common objects in everyday life. Figure 5.6 illustrates all of the objects that these models are able to detect in an image. The models that are provided include RetinaNet, YOLOv3 and TinyYOLOv3. [63].

```
person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop_sign,
parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra,
giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard,
sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket,
bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange,
broccoli, carrot, hot dog, pizza, donot, cake, chair, couch, potted plant, bed,
dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven,
toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair dryer, toothbrush.
```

Figure 5.6: Available labels for detection.

This section presents the tests done with object detection models, which are structured in the following way: on the left is the picture with the respective detections and on the right is an image with a graph representation of the detections. The color represents the detected object (label).

The pictures used for the testing of the described models are presented in Figure 5.7.

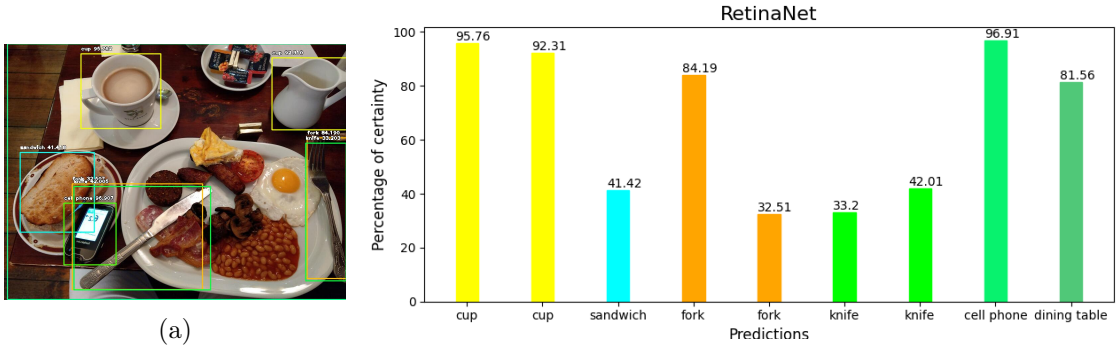
Figure 5.7: Sample pictures used for testing the object recognition models.

Figures 5.8, 5.9, 5.10 present the performance achieved for the first image. Figures 5.11, 5.12, 5.13 show the results obtained for the second image. Finally, Figures 5.14, 5.15, 5.16 illustrate the performance obtained in the processing of the third picture.

From the different test runs done it is possible to analyse that the TinyYOLOv3 model under performs severely compared to RetinaNet and YOLOv3. This is expected, as explained in Appendix A Section A.3.3, the TinyYOLOv3 model is a smaller model of YOLOv3 that requires less computational resources and is better suited for more constrained environments with smaller targets.

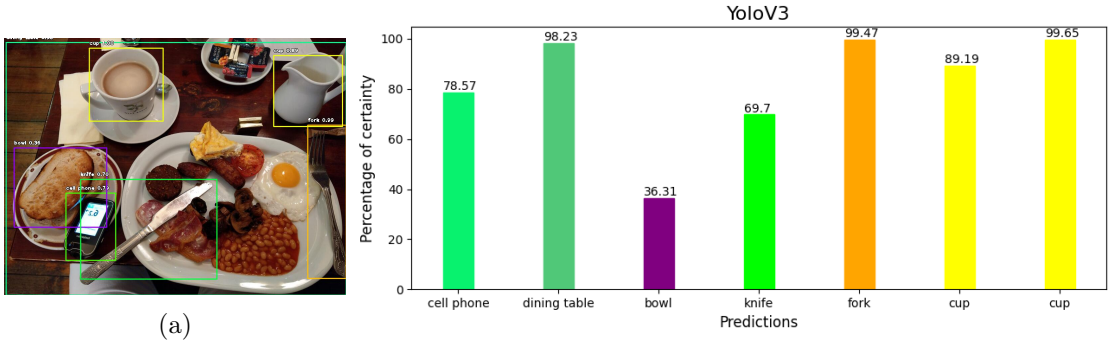
Comparing RetinaNet to YOLOv3 it is possible to conclude that YOLOv3 is more accurate than RetinaNet. For example, in the first run, RetinaNet detects knifes and forks in the same place as shown in Figure 5.8 a), in the third example RetinaNet detects a bus in the place of a building as illustrated in Figure 5.14 a) while YOLOv3 was capable of correctly detecting a stop sign that no other model detected, as shown in Figure 5.15.

- **Object Detection Experience Number 1**



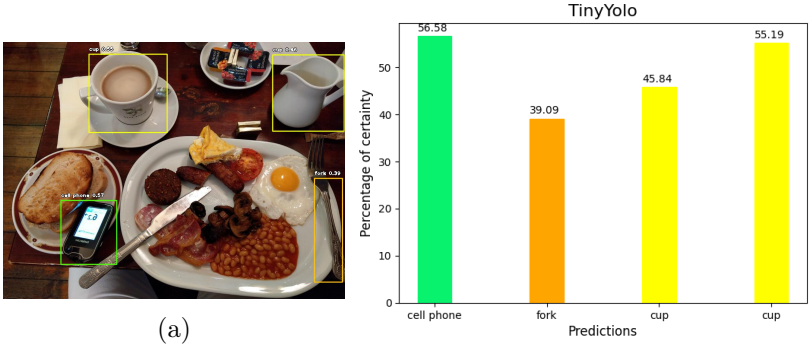
(a) (b)

Figure 5.8: Test run 1 with RetinaNet; a) Analysed picture with detections; b) Performance achieved on detections.



(a) (b)

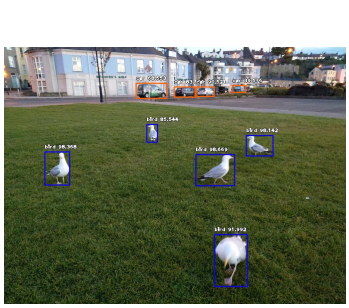
Figure 5.9: Test run 1 with YOLOv3; a) Analysed picture with detections; b) Performance achieved on detections.



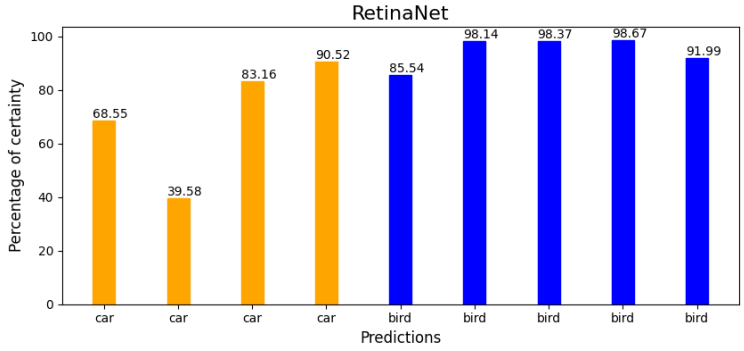
(a) (b)

Figure 5.10: Test run 1 with TinyYOLOv3; a) Analysed picture with detections; b) Performance achieved on detections.

- **Object Detection Experience Number 2**

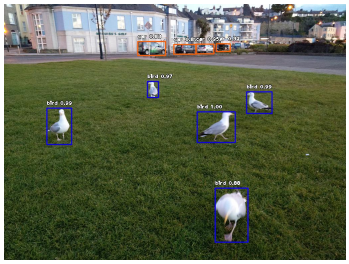


(a)

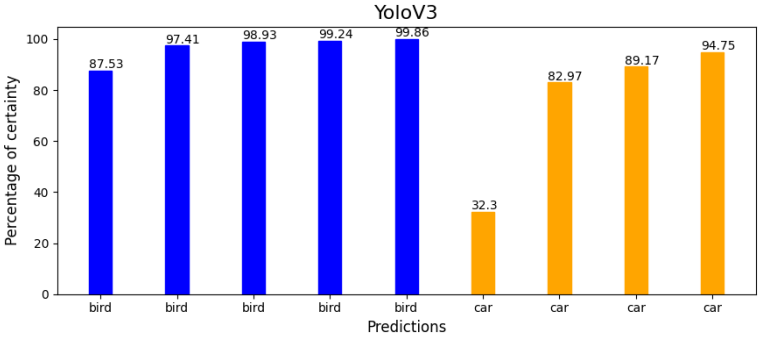


(b)

Figure 5.11: Test run 2 with RetinaNet; a) Analysed picture with detections; b) Performance achieved detections.

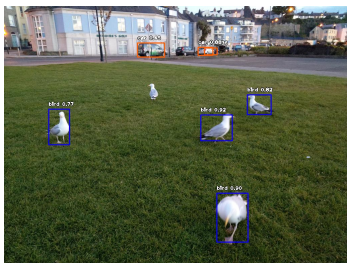


(a)

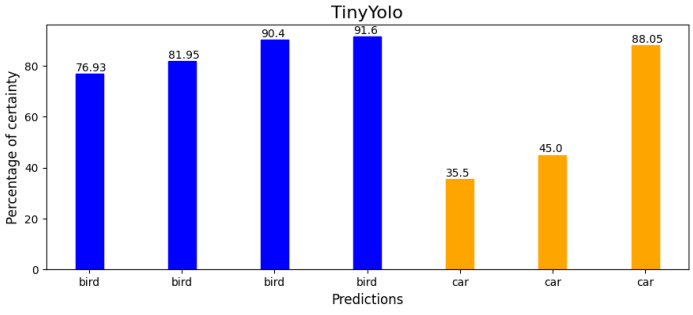


(b)

Figure 5.12: Test run 2 with YOLOv3 model; a) Analysed picture with detections; b) Performance achieved on detections.



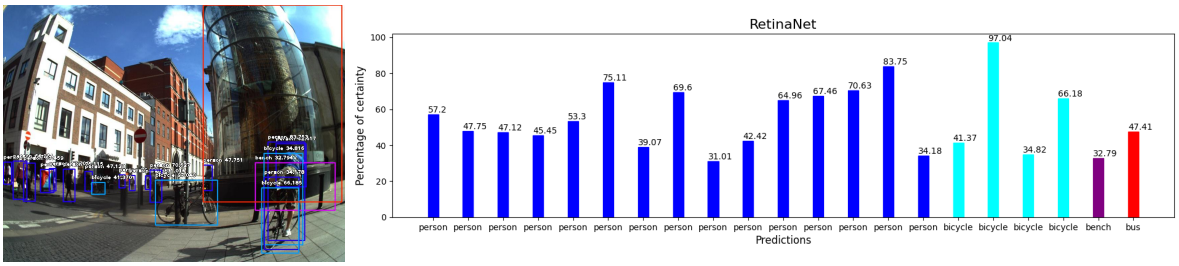
(a)



(b)

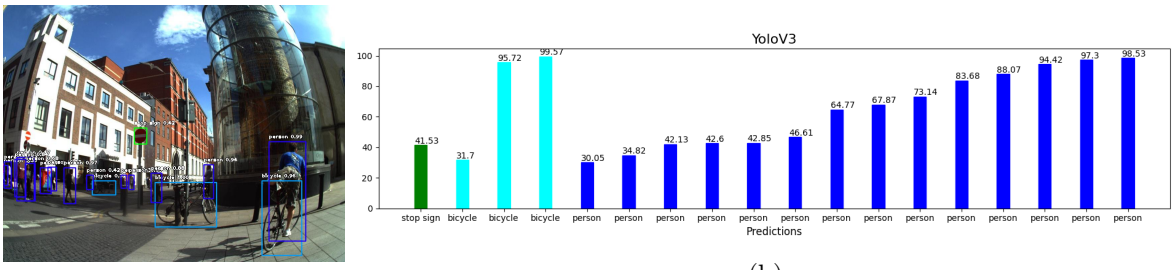
Figure 5.13: Test run 2 with TinyYOLOv3; a) Analysed picture with detections; b) Performance achieved on detections.

- **Object Detection Experience Number 3**



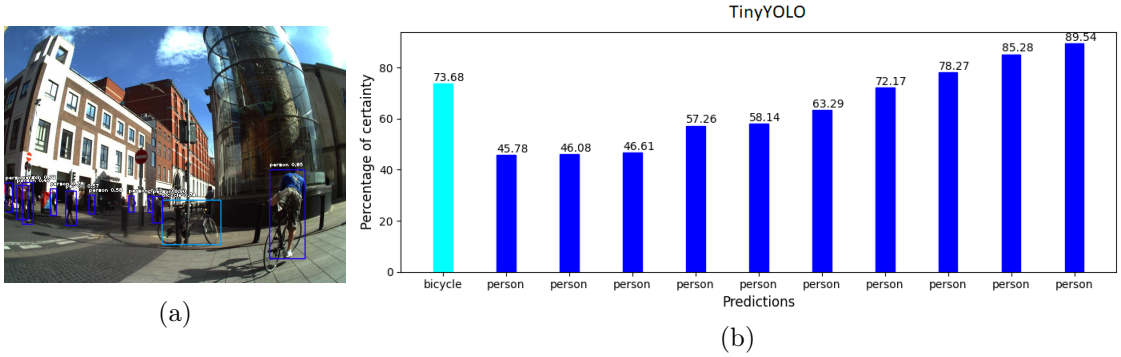
(a) (b)

Figure 5.14: Test run 3 with RetinaNet; a) Analysed picture with detections; b) Performance achieved detections.



(a) (b)

Figure 5.15: Test run 3 with YOLOv3 model; a) Analysed picture with detections; b) Performance achieved on detections.



(a) (b)

Figure 5.16: Test run 3 with TinyYOLOv3; a) Analysed picture with detections; b) Performance achieved on detections.

5.2.3 | Object Detection Word Clouds Preliminary Experiences

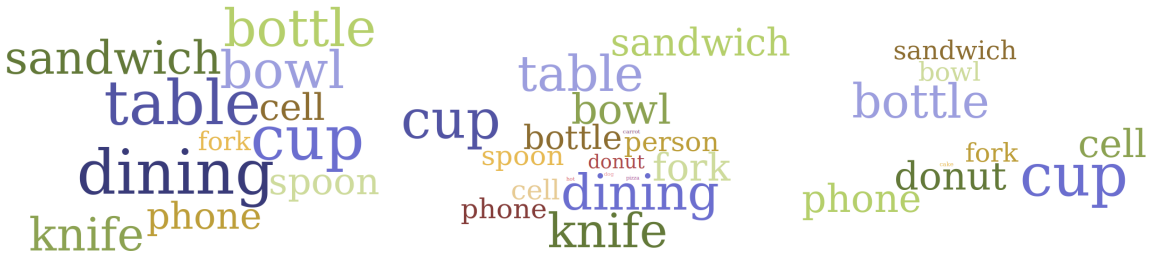
In order to make the labels extraction more easily visible and still achieve some degree of performance comparison between the 3 object detection models, word clouds were generated. In a word cloud, the bigger a word is the more times that label was detected in the pictures. For this test 6 images were processed with identical setting in order to generate 1 word cloud for each object detection model with all of the extracted labels.

Figure 5.17 presents the images that were used for word cloud generation:

Figure 5.17: Used images for word cloud generation.

As for the word clouds generated, it is possible to notice that in the YOLOv3 cloud and the RetinaNet cloud (shown in Figure 5.18 a) and b) respectively) there are many more words than in the TinyYOLOv3 cloud (shown in Figure 5.18 c)). Again, TinyYOLOv3 is severely under performing when compared to the other 2 models. Looking at the YOLOv3 model word cloud its possible to notice some consistency because most of the words have the same size. In the RetinaNet word cloud there are many words of different sizes, this can occur because RetinaNet wrongly detects 1 or 2 object like “pizza”, “donut” and “person” in some of the images. In addition, it is also possible to analyse that much of the extracted labels for all of the models were the exact same, this happens because all of them were trained using the same dataset, as previously explained. Using a different model of YOLOv3 trained with other dataset would probably provide a word cloud with different labels.

These test runs allow for the conclusion that the YOLOv3 model is the better performing one and therefore it was the one chosen to do object detections for the automatic image retrieval system.



(a) (b) (c)

Figure 5.18: Generated Word Clouds; a) YOLOv3 word cloud; b) RetinaNet word cloud; c) TinyYOLOv3 word cloud

5.3 | Example of a Raw Retrieval System

As a first step in building a fully automatic retrieval system an "raw retrieval system" was created without any text processing and very raw on the way it worked. Simply put, a user just needs to write a label, according to one of the words available for detection, and the system will scan all the images that are inside a directory and return the images that have detections of that specific user inputted label. The user is also able to input the minimum percentage probability for the detections, therefore, if the user chooses "cup" and "40%", objects that are not "cup" or that are "cup" but below the threshold of 40% wont be retrieved.

Figure 5.19 illustrates the "raw retrieval system" user interface and Figure 5.20 presents the retrieved images for the given example.

```
------------------------------------------------------------ Alpha Stage Retrieval System ------------------------------------------------------------

All images in the directory :

['2018-05-20 17.53.06.jpg', '2018-05-09 18.06.16.jpg', '2018-05-20 15.27.58.jpg', '2018-05-20 12.37.28.jpg', '2018-05-21 15.33.36.jpg', '2018-05-12 20.
02.51.jpg', '2018-05-21 15.24.45.jpg', '2018-05-09 18.06.08.jpg', '2018-05-12 21.34.19.jpg', '2018-05-11 16.13.09.jpg', '2018-05-21 15.24.49.jpg', '201
8-05-12 20.02.50.jpg', 'results', '2018-05-11 16.14.42.jpg', '2018-05-20 17.53.48.jpg', '2018-05-20 11.51.47.jpg', '2018-05-21 12.14.43.jpg', '2018-05-
21 06.55.46.jpg', '2018-05-20 21.27.51.jpg', '2018-05-20 11.23.44.jpg', '2018-05-20 13.20.09.jpg', '2018-05-20 13.13.48.jpg', '2018-05-20 16.11.26.jpg'
, '2018-05-21 11.16.11.jpg', '2018-05-21 16.09.41.jpg', '2018-05-12 21.34.16.jpg']

Words available for detection:

dict_keys(['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parkin
g meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'su
itcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle',
 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'ca
ke', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven
', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair dryer', 'toothbrush'])

Choose a word for detection : cup
Your chosen word is : cup
Choose a minimum percentage for the detection :40
Your chosen percentage is : 40%

------------------------------------------------------------------------ END OF MENU ------------------------------------------------------------------------
```

Figure 5.19: System capable of detecting specific user inputted labels in multiple images.

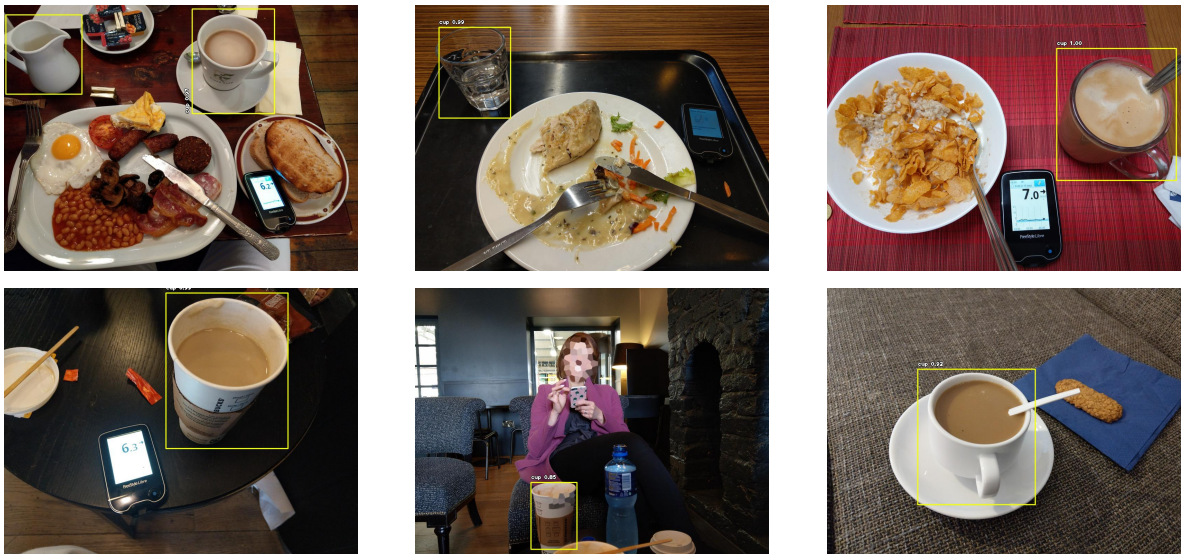
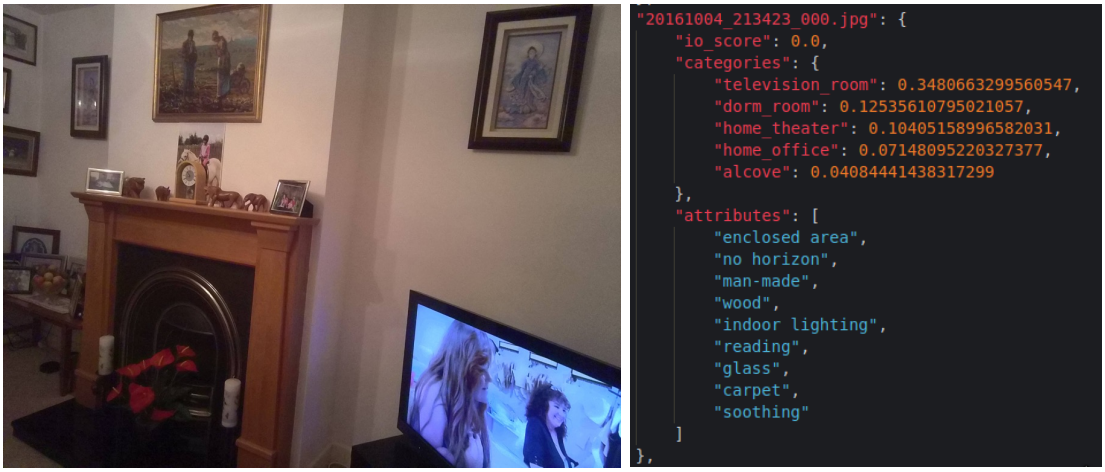


Figure 5.20: Retrieved images for the label “cup” with “40%” threshold.

5.4 | Scene Recognition

In order to detect indoors, outdoors and places a pre-trained model provided by Zhou et al., 2018 [88] trained on the Places365 standard dataset was used. The following example shown in Figure 5.21 presents the extractions done to a sample picture.




```
"20161004_213423_000.jpg": {
    "io_score": 0.0,
    "categories": {
        "television_room": 0.3480663299560547,
        "dorm_room": 0.12535610795021057,
        "home_theater": 0.10405158996582031,
        "home_office": 0.07148095220327377,
        "alcove": 0.04084441438317299
    },
    "attributes": [
        "enclosed area",
        "no horizon",
        "man-made",
        "wood",
        "indoor lighting",
        "reading",
        "glass",
        "carpet",
        "soothing"
    ]
},
```


(a) (b)

Figure 5.21: Example of scene recognition; a) Picture 20161004_213423_000.jpg from the imageclef dataset; b) Scene recognition model output for that image.

The “io_score” represents the indoor vs outdoor certainty. It ranges from 0 to 1. If its close to 0 it means the image is probably an indoor scenery and if it is close to 1 it means it is probably an outdoor scenery. In this example, the image represents an indoor scenery and the “io_score” is 0, therefore the model predicted correctly that the image is in fact an indoor scenery.

Following up, the "categories" is where the model tries to predict what the image represents in terms of a scene. In this case the model predicts with 34.8% accuracy that it is a "television_room" which is also correct, since the image represents a division of a house with a television.

Finally, for the "attributes" is where the model tries to describe the picture. Some of the attributed labels were "enclosed area", "man-made", "indoor lightning" and so on which are all correct since the image represents a man-made enclosed space structure with indoor-lightning.

5.5 | ImageCLEF Submissions

In this year ImageCLEFlifog LMRT subtask, 3 different runs were submitted. The first 2 runs belong to the automatic image retrieval system and the last to the interactive retrieval system.

Since the interactive system processed the images with detections from a combination of ResNeXt-101 and Feature Pyramid Network architectures in a basic Faster Region-based Convolutional Network (Faster R-CNN) pretrained on the COCO dataset proposed by Mahajan et al. [42] it was decided to reutilize those labels for the automatic system, since the labels were already readily available. Therefore, for the first submitted run of the automatic image retrieval system it was used the ResNeXt-101 labels and for the second run it was used the YOLOv3 [89] extracted labels.

The usage of two different set of extracted labels from two different object detection models was mainly done in order to have an understanding if the the difference in labels and the respective scores would help in achieving better results. Furthermore the images were also fully processed with the Places scene recognition model. This exhaustive approach of processing so many images took a lot of computer processing time and resources

Figure 5.22 shows the fully processing of the image shown in Figure 5.21 a) with YOLOv3 and PLACES365:


```
"20161004_213423_000.jpg": {
    "local_time": "2016-10-04_21:34",
    "concepts": {
        "clock": {
            "score": 0.40071901679039,
            "box": [
                295.0,
                263.0,
                317.0,
                302.0
            ]
        },
        "tv": {
            "score": 0.991506814956665,
            "box": [
                654.0,
                485.0,
                1013.0,
                765.0
            ]
        },
        "potted plant": {
            "score": 0.325846791267395,
            "box": [
                44.0,
                533.0,
                354.0,
                756.0
            ]
        }
    },
    "activity": "NULL",
    "location": "Verbena Avenue",
    "categories": {
        "television_room": 0.3480663299560547,
        "dorm_room": 0.12535610795021057,
        "home_theater": 0.10405158996582031,
        "home_office": 0.07148095220327377,
        "alcove": 0.04084441438317299
    },
    "attributes": [
        "enclosed area",
        "no horizon",
        "man-made",
        "wood",
        "indoor lighting",
        "reading",
        "glass",
        "carpet",
        "soothing"
    ],
    "io_score": 0.0
```


Figure 5.22: Fully processed image with YOLOv3 and PLACES365.

Figure 5.23 shows the fully processing of the image shown in Figure 5.21 a) with ResNeXt-101 and PLACES365:

Figure 5.23: Fully processed image with ResNeXt-101 and PLACES365.

The “activity” and “location” were extracted from the data provided by the organizers in a csv file. However, this data is not accurate enough, and a good option would be to use activity recognition algorithms to extract activities from images. However, the implementation of this processing algorithm was not done since the processing time was already to high with the current setup. The “local_time” was extracted directly from the picture name. The “box” is the pixels location of the respective “concept” in the image.

5.6 | Text Word Extraction and Categorization

In the text mining/processing stage, the query topics are analysed using Natural Language Processing tools to extract relevant words in order to retrieve the desired moment. Those words are compared with the visual concepts words obtained in the image processing stage.

The SpaCy model [81] is used to analyse the query topics fully, extracting relevant words from the title, the description and narrative. This model is an english multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl and is a general-purpose pre-trained model to predict named entities, part-of-speech tags and syntactic dependencies. It can be used out-of-the-box and fine-tuned on more specific data [81]. This model was chosen since it is specific for the English language and it is medium sized (48 MB) which is a good balance between loading speed and accuracy. The larger SpaCy model is 756 MB (79 times larger) which makes it much more slower to load and process. However, the difference in performance between the two models for the given task is negligibly.

The words extracted are divided into 10 categories being them : “relevant things”, “activities”, “dates”, “locations”, “inside”, “outside”, “negative relevant things”, “negative activities”, “negative locations” and “negative dates”. The importance of this extraction and categorization is done in order to later compare this extracted textual data with the extracted visual data from all of the images. To give a clear example, it is crucial to correctly extract the concept “fast food” from Figure 5.24 and categorize it in the “relevant things” textual category in order to compare with the “concepts” visual category. It is also important to extract the word “airport” and categorize it in the “locations” category in order to compare with the “location” visual category. It would not make sense to extract the word “fast food” and categorize it in the “locations” textual category for later comparing with the “location” visual category.

In order to assign words to each category some linguistic rules were defined, such as semantic and syntactic rules. Semantic rules build the meaning of the sentence from its words and how words combine syntactically. Syntax rules refer to the rules that govern the ways in which words combine to form phrases and sentences.

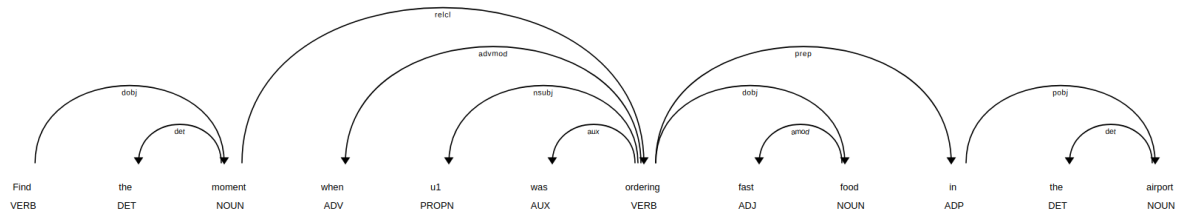


Figure 5.24: Linguistic annotations generated by the SpaCy library [81] for the narrative of the topic 6 of the test topics.

Using Figure 5.24 to illustrate, the extracted words for the the narrative of the test topic 6 are presented next:

- **relevant things** : "fast food".
- **activities** : "ordering", "ordering fast food".
- **locations** : "airport".
- **inside** : "True".
- **outside**: "False".

5.6.1 | Implemented Syntax Rules

The syntax rules were created from a continuous work from last year [4]. However, this year many more rules were implemented and more categories for word categorization were created like the negatives category. Something very important to note is that the syntax from the dev topics and test topics is very specific and very similar, therefore some syntax rules that were created probably wouldn't work so well for other kinds of documents with different syntaxes.

An excerpt of few of the most simple syntax rules that were applied to the extraction and categorization of the textual data are presented next:

- If the word is a "VERB" and ends with "ing", like "ordering" then it probably is an activity and if the words that follows are "NOUN" then those words probably refer to a location or an object.
- If the word is and "ADP" (adposition) and its either "in" or "at" then the words that follow are probably locations and usually means that the moment occurs inside a location and not outside, the category "inside" is then flagged to "True" and "outside" is flagged to "False".
- If the word is a "NUM" (number) then it probably refers to a year.
- If the word is a "VERB", and the following word is a "poss" (possession modifier) and if the following word to the last is a "NOUN", then all of this sentence refers to an activity and the NOUN to an object. For example "Repairing his car". "Repairing" and "Repairing his car" are the activities and "car" is the relevant thing.
- If the sentence has an auxiliary verb, the main verb usually corresponds to an activity and the words that follow the main verb may be objects or locations.
- If the word is an "ADJ" (adjective) then the following word is probably an object. It can also be a bi-gram like "ice cream" or in this case "fast food".
- If the sentence ends with "not considered relevant" the extracted words go to the negative categories.
- Rules for the extraction of dates, like the day of the week, the month or even years were also created. However do the syntax of the test topics, since they had no reference to dates in the text, the dates category was discarded in order to save time. Nevertheless,

for the dev topics dates were used and produced good results. As an example if in the topic it was said that the moment happened on a "wednesday" or in year 2014, only pictures from wednesday or the year 2014 were retrieved.

The main idea in creating these syntax rules was to look at the linguistic annotations that the SpaCy model provided and see which words were the ones important for extraction. This was something that was complex to do, but after making the algorithm work for the dev topics, the test topics did not require much iteration since the syntax was very much identical to the dev topics. Furthermore, many more syntax rules than the ones that were presented were also implemented which made the algorithm increasingly complex. However, the word extraction and categorization is not flawless and sometimes it places things to wrong categories or misses one or two words for extraction. Nevertheless, in the current state, it can be considered that it it working well enough.

5.7 | Image Retrieval

In the retrieval step the images are recovered according to the desired query topic. As an example Figure 5.25 represents the test topic number 7.

```
<topic>
      <id>007</id>
      <type>recall</type>
      <uid>u1</uid>
      <title>Seafood at Restaurant</title>
      <description>Find moments when u1 was eating seafood in a
restaurant in the evening time</description>
      <narrative>The moments show u1 was eating seafood in any
restaurant in the evening time are considered relevant. Any dish has
seafood as one of its parts is also considered relevant. Some
examples of the seafood can be shrimp, lobster, salmon.</narrative>
</topic>
```

Figure 5.25: Test topic number 7.

The extracted words of the title, description and narrative are as follows:

- **relevant things** : "seafood", "parts", "shrimp", "lobster", "salmon".
- **activities** : "eating", eating seafood.
- **locations** : restaurant, "evening time".
- **inside** : "True".
- **outside**: "False".
- **dates**: NULL.
- **negative relevant thing**: NULL.
- **negative activities**: NULL.
- **negative locations**: NULL.
- **negative dates**: NULL.

An important detail to be noticed is that the extracted words "evening time" are wrongly categorized on the "locations" textual category. The main reason for this error to occur is probably a direct consequence of one of the syntax rules that did not work as intended for this specific case. The sentence is written in the following manner "The moments show u1 was eating seafood in any restaurant in the evening time..." and one of the rules for locations extraction is, as previously discussed, if the word is an "ADP" (adposition) and its either "in" or "at" then the words that follow are probably locations. However, in this case, there are two different "in", one refers to "restaurant" which is a location and the other to "evening time" which is not a location. This is an example of the algorithm not working as intended.

5.7.1 | Retrieving Images According to the Similarity Between Words

In order to retrieve images according to a defined moment in a textual topic a comparison is made between the extracted labels from the images and the words extracted from the topics. This comparison is done trough the calculation of similarity score obtained by an NLP model provided by the SpaCy library. This similarity score alongside the weight attributed to each category is critical for the calculation of the confidence score for each image. The images with the highest confidence scores for a given topic are selected for retrieval.

- **Similarity Score Computation**

The SpaCy model allows the computer to calculate the similarity between the visual concepts and the extracted data. As an example of similarity between words using the described model, the word "television" and "seafood" have a similarity of approximately 0.07 (7%) while "television" and "screen" have a similarity of approximately 0.43 (43%).

Using Figures 5.22 and 5.23 to illustrate the different categories, the calculation of the similarity scores is done trough the comparison of the extracted words and the visual concepts:

1. The visual category "concepts" is compared to the textual category "relevant things" and "negative relevant things";
2. The visual category "activity" is compared to the textual category
3. The visual category "location" is compared to the textual category "locations" and "negative locations";;
4. Depending if the textual category "inside" = "True" / "False" the visual category "io_score" value will have different impact on the confidence score.

- **Category Weight**

The weight for each category is obtained through two different factors, an importance weight factor and a distributed weight factor. The weight of the category is therefore the sum of the importance and the distributed weight factor.

$$\textbf{Weight} = ImportanceFactor + DistributedFactor$$

- **Importance Weight Factor**

The importance weight factor is a value that is assigned to the positive available categories that are not empty. It is a pre-defined value and is the factor that gives more importance to one category than others for the contribution of the confidence score computation. This factor ranges from 0 to 1 and the sum of all of the factors is equal to 1. For example, if the category of "relevant things" is worth 0.4 and if the category of "activities" is worth 0.2, then "relevant things" has a weight always two times higher than "activities" for the computation of the confidence score.

- **Distributed Weight Factor**

The distributed weight factor is the distribution of the importance weight factor from an empty category to all other categories that are not empty. To make it clear, if the text mining algorithm didn't extract any words to the "activities" category then there is no use in calculating millions of similarity scores that are comparing the images category "activities" with the NULL "activities" textual category .

Assuming the following example:

1. (Not Empty category) "Relevant Things" importance weight factor = 0.4 (40%);
2. (Empty category) "Activities" importance weight factor = 0.2 (20%);
3. (Not Empty category) "Inside/Outside" importance weight factor = 0.3 (30%);
4. (Not Empty category) "Location" importance weight factor = 0.1 (10%);
5. Sum of all importance factors = 1.

In this example, the "Relevant Things" category is worth 40% of the sum of all importance factors while "Activities" is worth 20%, "Inside/Outside" is worth 30% and "Location" is worth 10%. Therefore, if by any means the "Activities" category is empty, the importance weight factor of this category will be distributed to all other categories in the following way:

1. Distribution : 0.4x + 0.3x + 0.1x = 0.2 (=) x = 0.25;
2. Weight "Relevant Things" = 0.4 + 0.4*0.25 = 0.5 (50%);
3. Weight "Activities" = 0;
4. Weight "Inside/Outside" category = 0.3 + 0.3*0.25 = 0.375 (37.5%);
5. Weight "Location" = 0.1 + 0.1*0.025 = 0.125 (12.5%);
6. Sum of all weights = 1.

This distribution achieves two important things, the sum of all weights is always 1 (100%) and the ratio between categories is always the same. In the begin "Relevant things" was 40% and "Location" was 10% which is a ratio of 4 and in the end "Relevant things" is 50% and "Location" is 12.5% which is still a ratio of 4. This means that in this situation "Relevant things" has a weight of 50% for the computation of the confidence score.

The negative categories works the same way, but instead of contributing for the confidence score, it decreases the value.

- **Confidence Score Computation**

The confidence score is a value that also ranges from 0 to 1 (0% to 100%) and is the final value that gives the verdict on how an image relates to a given moment. A score of 20% means that the image is not related while a higher score of 80% means that an image probably relates to the moment.

The computation of the confidence score is also influenced by the score of the image concepts obtained trough the image processing phase. This means that an image with low prediction score (from the image processing stage) will have a lower confidence score.

Since the images dataset consists in 200.000 images and the test topic dataset consists on 10 topics, approximately 2.000.000 confidence score calculations had to be computed. This step is extremely exhaustive on computer time and resources, which made it hard to make adjustments, correct errors and bugs in the code.

In order to have a better visualization of how all of this calculations are done the follow equations illustrate all the steps needed in order to compute the confidence:

$\textbf{ConceptScore} = [Weight_1] \times [HighestSimilarityScore] \times [VisualScore]$
$\textbf{LocationScore} = [Weight_2] \times [HighestSimilarityScore]$
$\textbf{ActivityScore} = [Weight_3] \times [HighestSimilarityScore]$
$\textbf{InsideScore} = [Weight_4] \times (1 - [ioscore])$
$\textbf{OutsideScore} = [Weight_5] \times ([ioscore])$
$\textbf{PositiveScore} = ConceptScore + LocationScore + ActivityScore + (Inside||Ouside)Score$
$\textbf{NegativeConceptScore} = [Weight_6] \times [HighestSimilarityScore] \times [VisualScore]$
$\textbf{NegativeLocationScore} = [Weight_7] \times [HighestSimilarityScore]$
$\textbf{NegativeActivityScore} = [Weight_8] \times [HighestSimilarityScore]$
$\textbf{NegativeScore} = Negative(ConceptScore + LocationScore + ActivityScore)$
$\textbf{ConfidenceScore} = PositiveScore - NegativeScore$

- **Image Retrieval**

Finally, a script runs through all the selected confidence scores for a given query topic and stores the 50 pictures with the highest confidence score for each topic. The 10 highest pictures for each topic are the ones who count for the F1@10 score.

- **Discarded Categories**

Due to the fact that the PLACES365 scene recognition model extracts scenes with very low scores, rarely above 30%-40% it was decided to discard the category "categories" and "attributes" from contributing to the confidence score in order to save processing time.

- **General Thresholds**

A few general thresholds were defined:

1. Visual concept label score minimum threshold : 0.3;
2. “Relevant Things” category score minimum threshold: 0.15;
3. Confidence score minimum threshold : 0.25.

The thresholds were implemented through some trial and error during the test phases, and they merely serve the purpose of saving some computational time so that confidence scores of images with low scores are not fully processed.

5.8 | Submitted Run 1

The first run that was submitted for the imageclef LMRT subtask used a combination of ResNeXt-101 and Feature Pyramid Network architectures in a basic Faster Region-based Convolutional Network (Faster R-CNN) pretrained on the COCO dataset for the extraction of visual concepts.

In this run all of the importance weight factors for all categories were the same. This means that each category counts the same for the computation of the confidence score. No category is more or less important. When a category is empty, their respective importance factor is equally apportionated to all other categories but never to the negative categories. If a negative category importance factor is negative, their factor is apportionated to the other negative categories. And if a positive category is empty, their factor is apportionated to the positive categories.

Another aspect of Run 1 is that the negative categories can only impact the confidence score up to 50% in the image confidence score.

5.9 | Submitted Run 2

In the second run, the object detection algorithm used is the YOLOv3 model pretrained in the COCO dataset. It was decided to define the importance weight factor differently for each category. It was given a bigger importance to specific categories like “relevant things” and “io_score”. Categories like ”activities” and ”locations” get a lesser importance weight factor since they are being compared to the organizers labeled data which is limiting and lesser accurate. Another difference between Run 1 abd Run 2 is that all of the negative categories were discarded from contributing to the confidence score. This was done in order to save more processing time and to see if this had a good or bad impact on final results.

CHAPTER 6

Results

This chapter aims at presenting and discussing the results achieved in the ImageCLEF LMRT challenge. Firstly, in Section 6.1 a few examples of some tests that showcase the fine-tuning of the system are presented. Secondly, in Section 6.2 an example on how the system performed in a given topic is showcased and some insight is given on the differences in performance between the submitted runs. Finally, in Section 6.3 the results achieved on the challenge are presented.

6.1 System Fine-Tuning Using The Dev Topics

The first tests using the system architecture described on the previous chapter were run on a laptop. The dataset of pictures used was smaller (20.000 images) and not all topics were fully analysed. This tests took between 8h-10h each and they were done in order to find a good weight distribution between each category, detect bugs on the code and overall fine-tune the system before sending the code to the main processing computer (where the fully processing of the dataset took up to 1-2 days).

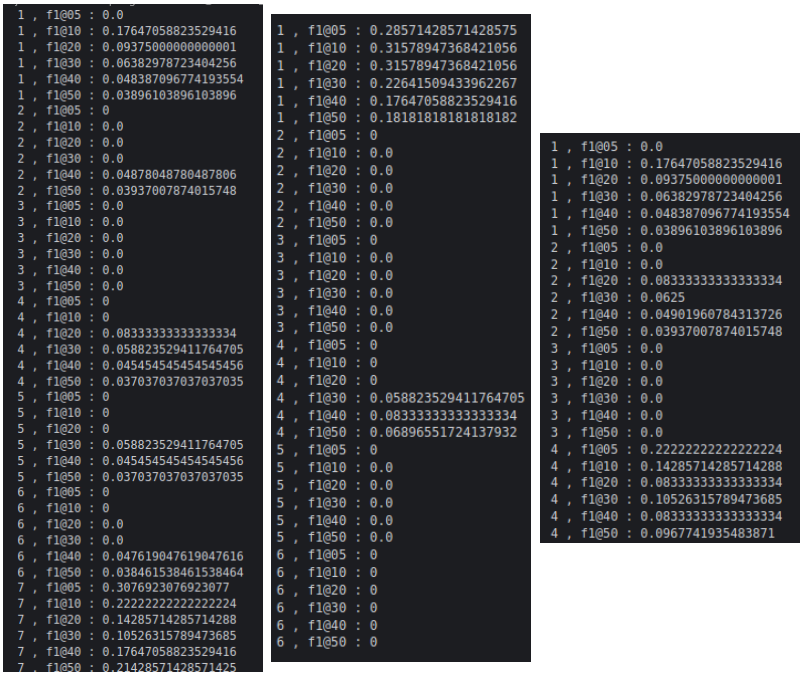
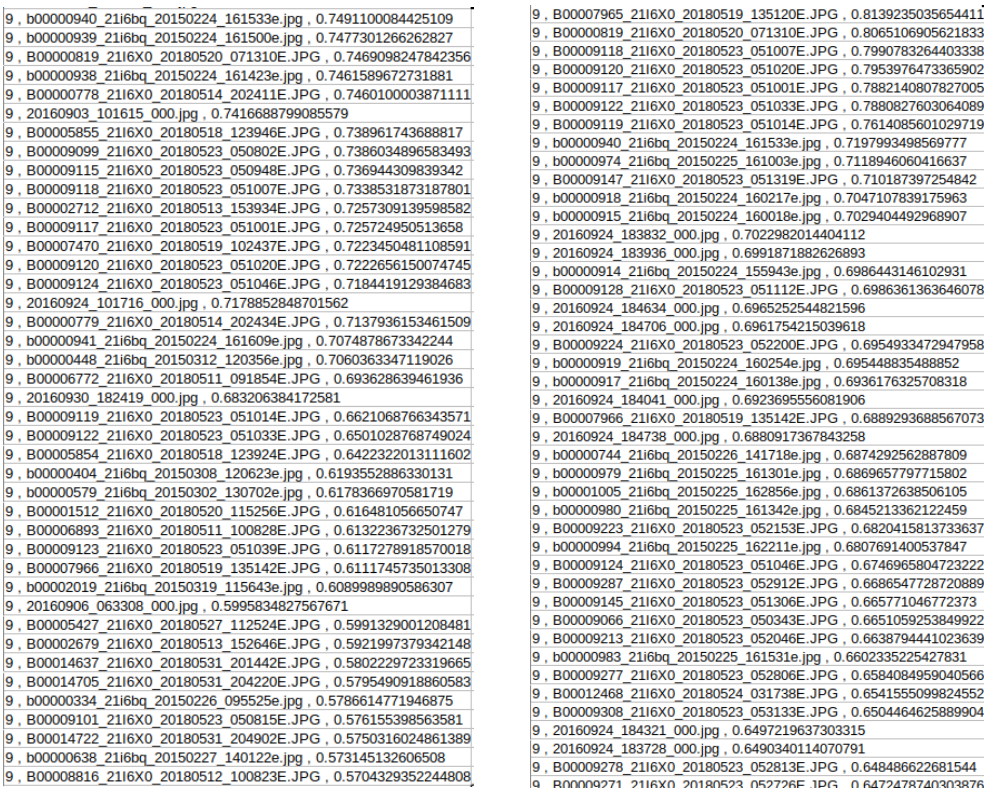


Figure 6.1: Examples of some different results with the fine-tuning of the weight distribution.

The examples in Figure 6.1 show the different achieved performances of the system in the F1-measure@XX during testing. In some cases the importance weight factor attributed to specific categories might be higher or lower, in other cases the general thresholds were changed and in other cases the negative categories were discarded. This tests were conducted during various days with the objective of using the case where the scores were better for the processing of the full dataset. However, since only a small sample of the dataset was processed during testing and fine-tuning, there were no guarantees that having good results during testing would produce good results when processing the full dataset.

6.2 | Final System Performance Example

After computing all confidence scores for every image and every topic, the system generates an csv file used for evaluation. The file is organized in the following way: **[topic id number, image name, confidence score]**. The 50 highest scoring pictures for each topic are stored in this file, however only the 10 highest pictures for each topic are used for the challenge evaluation. An excerpt showing the performance of both runs in test topic 9 is presented in Figure 6.2. This will be further discussed in Section 6.2.1.



(a) Run 1 (b) Run 2

Figure 6.2: Achieved results on topic 9 of the test topics.

6.2.1 | Topic 9 Performance Analysis

In order to critical analyse the retrieved images from both runs it is important to understand the moment described in topic 9, which is presented next:

Title: "Eating pizza".
Description: "Find the moments when u1 was eating a pizza while talking to one man".
Narrative: "To be considered relevant, the u1 must eat or hold a pizza with a man visible in the background. The moments that u1 was talking to more than one person are not relevant".

Analysing the images retrieved by both runs for topic 9 (illustrated in Figure 6.3 and 6.4) it is clear that run 1 achieved better performance for this specific topic, since 3 of the top 10 images returned belong to the moment described while for run 2 only 2 images from the top 10 belong to the moment.

As shown in Figure 6.3 for run 1 the top 1, 2 and 4 all belong to the moment since all of these images show the user eating pizza with a man visible in the background. Furthermore, the top 5 image from run 1 also shows the user with a pizza in the background, however there is no man in the background. Subsequently, the top 6 image has a poster of a pizza. Image top 7,8,9 and 10 are not related to the moment at all, however all of the pictures are inside interiors and illustrate food which is similar to the description of the moment.

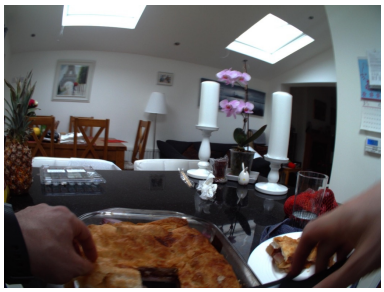
For run 2 (represented in Figure 6.4), the top 8 and 9 are images that are related to the topic, since both of them also show the user eating pizza with a man on the background. All of the other images are related to food being eaten inside an interior, however they do not belong to the described moment. Top 1 and top 2 images show an image of a lasagne which is similar to pizza where the scenario can be considered identical.

Some possible reasons that lead run 1 to achieve better performance than run 2 are presented next:

- The negative categories on run 1 might have decreased the confidence score on pictures that don't belong to the topic.
- The object detection algorithm on run 1 might have provided better detections than the algorithm used for run 2.
- The category weight distribution on run 2 might have decreased the weight on some categories that were important for the confidence score calculation in the images that belong to the topic.
- The difference in the similarity score between the different visual concepts on each run might also impact the performance of the system.

(a) Top 1

(b) Top 2



(c) Top 3

(d) Top 4

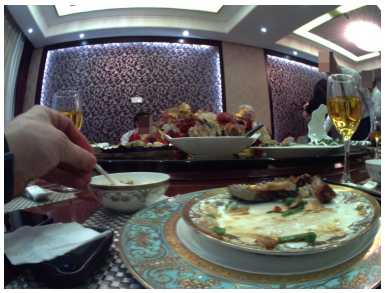
(e) Top 5

(f) Top 6

(g) Top 7

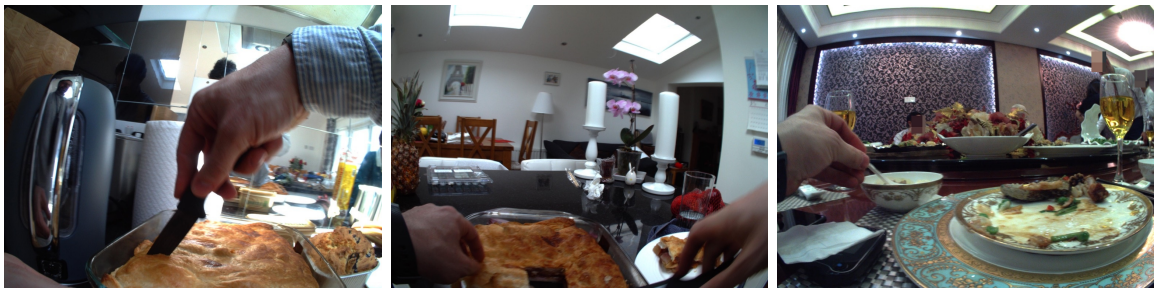
(h) Top 8

(i) Top 9



(j) Top 10

Figure 6.3: Top 10 retrieved pictures for topic 9 on run 1



(a) Top 1 (b) Top 2 (c) Top 3

(d) Top 4 (e) Top 5 (f) Top 6

(g) Top 7 (h) Top 8 (i) Top 9

(j) Top 10

Figure 6.4: Top 3 retrieved pictures for topic 9 on run 2

6.3 | Achieved Overall Performance Results

Tables 6.1 and 6.2 provide an overview of the performance results achieved in different runs for different systems that were obtained in the year 2019 and 2020 for the ImageCLEFlifelog LMRT subtask.

2019 Results			
Team	System Type	Run Name	F1-measure@10
UA.PT Bioinformatics	automatic	Run 1	0.016
UA.PT Bioinformatics	automatic	Run 2	0.026
UA.PT Bioinformatics	automatic	Run 3	0.027
UA.PT Bioinformatics	automatic	Run 4	0.027
UA.PT Bioinformatics	automatic	Run 5	0.036
UA.PT Bioinformatics	automatic	Run 6	**0.057**
Best Results Achieved by a Team			
HCMUS	interactive	Run 2	**0.61**

Table 6.1: Results obtained in 2019 from UA.PT Bioinformatics [4] and the best team [90].

2020 Results			
Team	System Type	Run Name	F1-measure@10
UA.PT Bioinformatics	automatic	Run 1	0.031
UA.PT Bioinformatics	automatic	Run 2	0.031
UA.PT Bioinformatics	interactive	Run 3	**0.517**
Best Results Achieved by a Team			
HCMUS	interactive	Run 10	**0.81**

Table 6.2: Results obtained in 2020 from UA.PT Bioinformatics [3] and the best team [5].

6.3.1 | Overall Performance Analysis

Comparing the Table 6.1 that shows the results of the year 2019 and Table 6.2 that shows the results of this year challenge, it is clear that there was no overall improvement on an automatic system performance. Furthermore, it is also possible to clearly see the difference in performance between interactive systems and fully automatic systems. Having user interaction and visualizations yields much better results than a fully automatic system [3].

The tables shown make a strong argument that for the ImageCLEF LMRT sub-task an interactive approach is a much better suited method, the user visualization and interaction with the application allows for much more accurate results since the user can manually choose the picture that he thinks is correct for the corresponding moment.

Another important aspect to notice is that the results of the automatic approach this year achieved the same exact F1-measure@10 score, this is highly due to the fact that even when using different state-of-the-art object detection algorithms, different weights for each category

and even using negative categories on one run and not on the other, much of the data used for both runs was provided by the organizers which is a highly faulty and inaccurate.

The interactive system is built around a web application where the user has interaction with 3 different stages of the application. These stages are upload, retrieval and visualization.

In the upload, the user uploads the dataset to be processed. During the retrieval stage, the user inputs the words extracted from the query topic into several categories and a comparison is done with the inputs and the app database information. This comparison is done in order to compute a confidence for each image for the assigned topic. Finally, in the visualization stage the user visualizes the images retrieved in forms of image gallery or data tables and manually selects the relevant clusters for the query topic. In order to improve the results, the user can exclude several irrelevant images from the selected clusters. To improve the cluster recall of the run, the user can change the confidence of a relevant image of each selected timestamp clusters [3].

This last visualization step is the critical difference between both systems. In the interactive system the the user can exclude images that he thinks that do not belong to the moment described in the query topic. This makes the automatic system incapable of competing with the results of the interactive system.

Figure 6.5 shows a screenshot of the application retrieval view and Figure 6.6 illustrates the different stages presented in the interactive system.

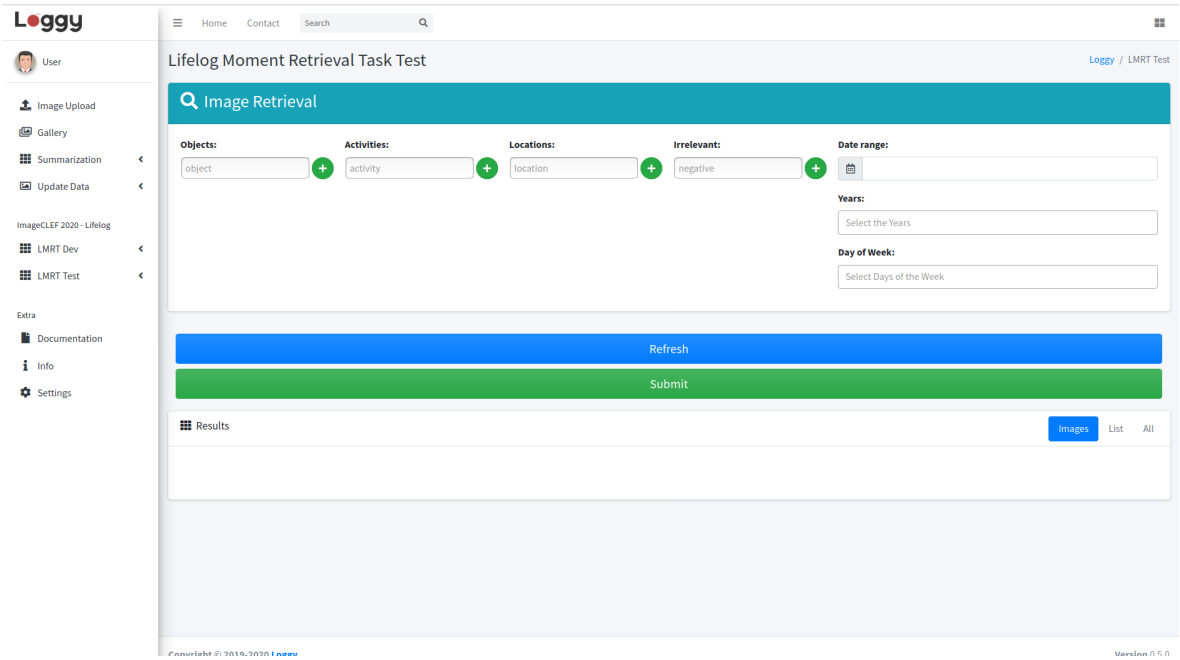


Figure 6.5: Web application retrieval view [3].

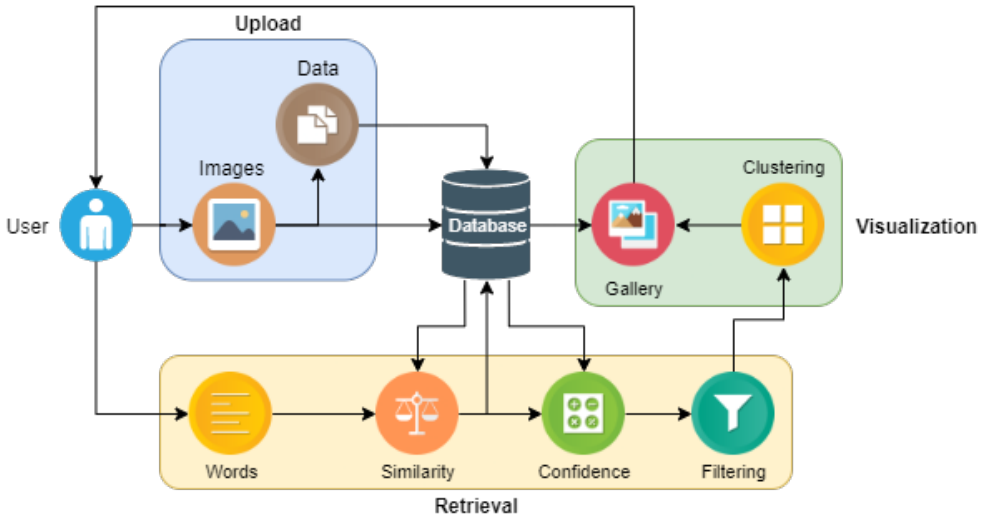


Figure 6.6: General representation of the developed web application. The user interacts with the three stages of the application: Upload, Retrieval and Visualization [3].

CHAPTER 7

Conclusions

The objectives defined in the first chapter for this thesis were fully achieved. However, like last year, the end results are not satisfactory enough.

Given the complexity and difficulty of creating a fully automatic image retrieval system and considering the limitation in research in this area and the extreme amount of computing processing time and resources a system of this kind requires, it can be concluded that the results of this work open the possibility for more exploration into to the development of a more robust system.

With the current state of the art and computer technology fully automatic retrieval systems cannot surpass interactive systems in performance. Some very good reasons for this is that interactive systems offer user visualization, user interaction and user decision. This helps the system to be tremendously more accurate than a computer, since the user can correct the computer if the retrieved images are not related to the topic.

However even though the end results have low-score, it was still possible to present results that prove that the automatic retrieval system built was capable, in some situations, of working like intended. This was shown in the given example in Chapter 6 where the pictures returned in the first run did indeed belong to the moment that was described in the test topic. This concludes that a system like this can work, and can contribute to the improvement of the quality of life of the human kind, however it still requires a lot of improvement. Furthermore, like described in Chapter 2, the ground truth of the challenge does not consider all possible images for a moment, therefore it is possible that the present system might actually perform better than the challenge results achieved demonstrate.

7.1 System Advantages

- Capable of fully processing a big dataset of images, extracting dates, object labels and scene labels;
- The system is modular, which means it can be easily added new algorithms to fine-tune the image processing or more linguistic rules that can achieve better text mining results;
- Capable of text processing and word extraction to specific predefined categories;
- Capable of self-evaluation using the F1-measure@xx if the ground truth is available.

7.2 System Disadvantages

- Requirement of a lot of processing time in order to retrieve images of a big dataset;
- Requirement of good computer specs;

- Low-score end results;
- Dependence on constants/heuristics.

7.3 | Future Work

A few things can be done in order to improve system performance for the 2021 imageCLEF LMRT sub-task.

Firstly the text processing and word extraction stage can be improved in order to only extract meaningful words. Sometimes, in the extractions, words that had no meaningful use and were only clutter were extracted. For example "evening time" was in the category "locations". In most cases this wont influence the confidence score by much, but will increase a lot the processing time. This is because "evening time" will be compared with all of the images extracted "locations" from the organizers data. Another aspect that needs improving is the extraction of negative words which is very dependent on how the sentence is worded.

Using more powerful computers and improve system optimization is essential in order to conduct a more large quantity of tests to fine tune the system. Currently the system is so slow that the fine-tuning process becomes complex.

Since most of the dataset is comprised of folders of images of one full day, it is theoretically possible to link a set of images like a video and implement activity video recognition algorithms in order to extract the activity of the images instead of using the organizers data, which was in most cases inaccurate. Furthermore, better scene recognition and even color recognition algorithms will definitely improve the f1-measure@10 scores. Another suggestion is using algorithms to remove blurred images, which can help the time it takes for the system to process the dataset by removing low quality images. In addition, using two or three different object detection algorithms, all pre-trained with different datasets, will allow for the extraction of different image labels , which might improve performance, however this will require a lot of processing time in order to extract and then to compare the image labels with the textual data.

Another future improvement would be to implement the google cloud vision API [14], which extracts labels that are more related to the words extracted than the algorithms used in this work. Figure 7.1 shows a few examples of the extracted labels from google cloud vision API.

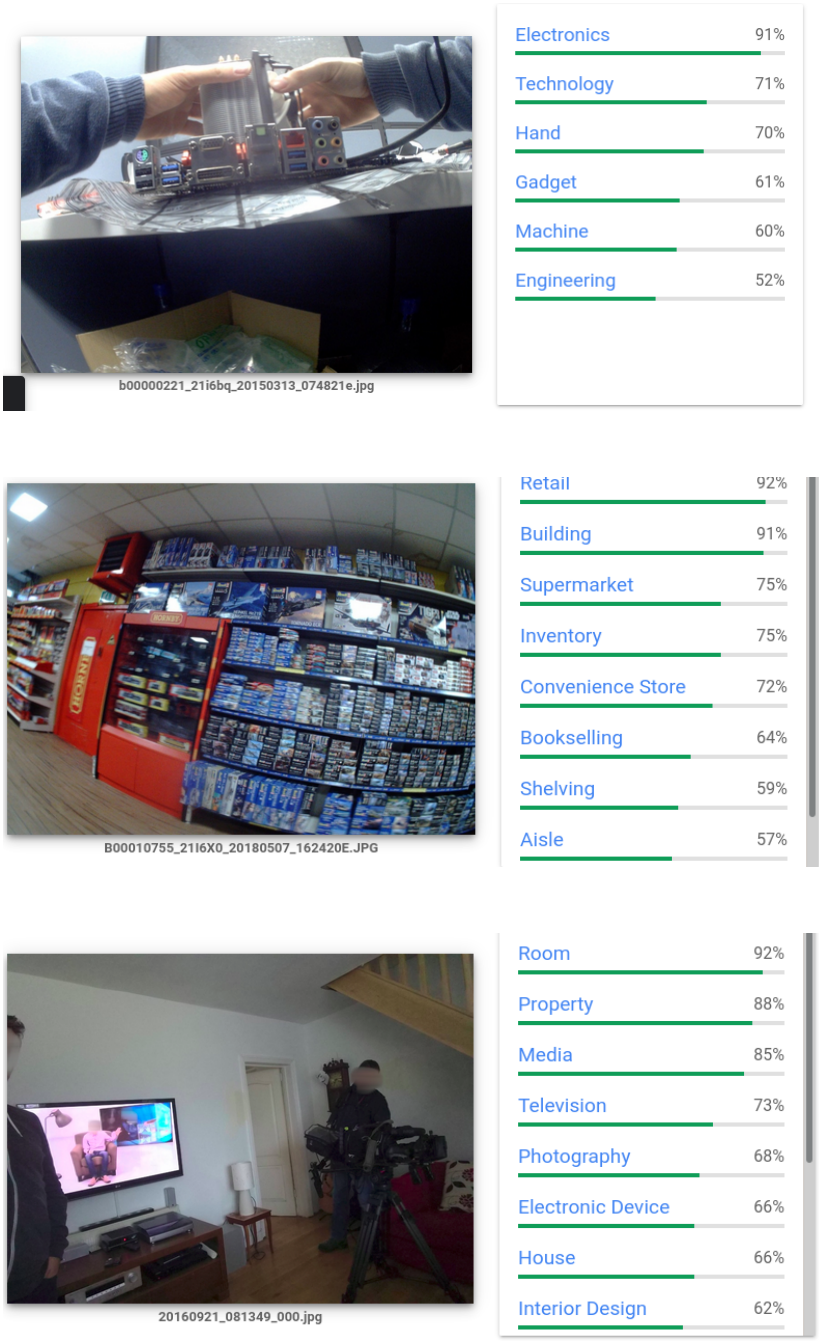


Figure 7.1: Examples of google cloud vision API extracted labels [14].

APPENDIX A

Neural Networks

This appendix provides an overview of the most common neural networks architectures. In addition, it also presents a survey on several image classification models, object detection classification based algorithms and regression based algorithms.

A.1 Types of Neural Networks architectures

A.1.1 Feedforward Neural Network

A Feedforward neural network has the most simple architecture, the data only travels in one single direction. It goes through the input node and exits at the output node. Since there is no back-propagation algorithm this neural network is not able to correct itself [20] [91].

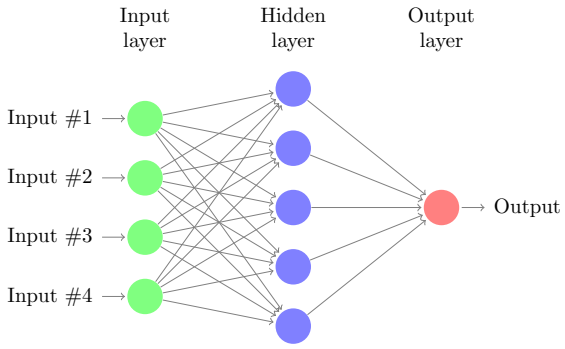


Figure A.1: Example of a Feedforward Neural Network with one hidden layer (with 5 neurons) [92].

A.1.2 Radial Basis Function Neural Network

This network is composed of two layers. In the first one, features are combined with a radial basis function in the inner layer. The second layer is the output, where these features are taken in consideration while computing the same output in the next function.

A radial basis function means that the distance of a point is considered with respect to the center [20].

A.1.3 | Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are designed to recognize sequential data characteristics and use patterns to predict the next likely scenario. In these kind of neural networks the signals are propagated in both directions as well as within the layers. They work on the principle of saving the output of a layer and feeding it to the input to help in the prediction of the outcome of the layer.

RNNs use the back-propagation algorithm which allows to make sure that the output is correct almost 100% of the time [20].

A.1.4 | Convolutional Neural Network (CNN)

CNNs, also known as ConvNets, are a class of deep neural networks that employ the mathematically convolutional operation in at least one of its layers and have a deep feed-forward (not recurrent) architecture [2]. They share similarities with feedforward neural networks, since neurons also have weights and biases that are able to learn. In this network the input features are taken like a filter, which allows the network to have memory, since it can remember the images in parts and compute operations like conversion of the image from RGB or HSI to grayscale, allowing the detection of edges and images that can be classified into different categories [20].

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows the encoding image-specific features into the architecture, making the network more suited for image-focused tasks, while further reducing the parameters required to set up the model [93].

CNN convolves learned features with input data, and uses 2D convolutional layers which make this architecture one of the best to process 2D data, such as images. They also remove the necessity of manual feature extraction. There is no need to identify features used to classify images since CNNs work by extracting them directly from images. This is important because relevant features are not pretrained, they are learned while the network trains on a dataset [17]. Figure A.2 illustrates the CNN architecture.

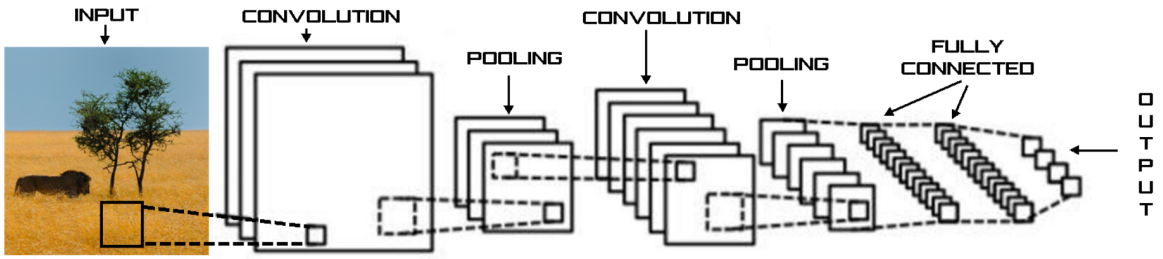


Figure A.2: CNN architecture [2].

Input Layers

The input layer is the first layer of a CNN and serves the purpose of resizing an image in order for it to pass onto further layers for feature extraction [2]. It also holds the pixel value of the images [93].

Convolutional Layers

The convolutional layers extracts low-level features from an image, such as edges, color or gradient orientation, according to the applied filter or kernel [2].

Activation Functions

The activation function introduces nonlinearity in order for CNNs to learn functionalities. They serve as decision functions and help in learning complex patterns. Some examples of activation functions are sigmoid, softmax and ReLU [2].

Pooling Layers

The pooling layers serve the purpose of reducing the parameters required and computation in the network by controlling the overfitting. This is achieved by reducing the spatial size of the network [2].

Overfitting happens when a model learns the detail and noise in the training data to an extent that it negatively impacts the performance of the model on new data [93].

Fully Connected Layers

The final layer in a CNN is usually a fully connect layer used for classification purposes. They take all features from the previous layer and compute class probabilities or scores. These features are then translated into a different class [2].

A.2 | CNNs architectures For Image Classification

A.2.1 | SquezeeNet

SquezeeNet is a deep neural network for computer vision that is more efficient for distributed training, since it requires less parameters to be transferred. The main goal of SquezeeNet creation was to obtain a smaller neural network with fewer parameters that could more easily fit into a computer memory, making it more easily transmitted over a computer network. This neural network was firstly implemented on top of the caffe deep learning software framework and later ported to the chainer deep learning software framework and Apache MXNET framework.

The basis of SquezeeNet consists in 3 ideas [94]:

- Replacing 3x3 filters with 1x1 filters and reduce the number of input channels. This improves computation speed and alleviates the computer resources required, since 1x1 filters have 9 times less parameters than 3x3 filters.
- Utilize 1x1 filters as a bottleneck layer to help reducing the computation required for the following 3x3 filters.
- Keeping a big feature map by down sampling late.

This neural network is built with fire modules, which are represented in Figure A.3.

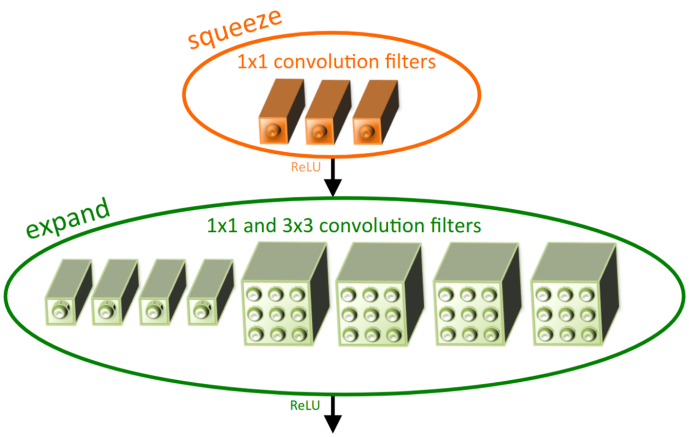


Figure A.3: SqueezeNet fire module [94].

The fire module contains both a squeeze layer and an expand layer. SquezeeNet stacks fire modules and pooling layers (this can be seen in Figure A.4). The squeeze layer and expand layer maintain the same feature map size, while the pooling layers reduce the depth to a smaller number, later increasing it. Reducing the depth means the expand layer has fewer computations to do, boosting the speed.

Squeeze layer architecture: Consists on 1x1 convolutions, it essentially combines all the channels of the input data into one (and thus reducing the number of input channels needed in the next layer).

Expand layer architecture: Consists on 1x1 convolutions mixed alongside 3x3 convolutions. The 1x1 convolutions combine the channels of the previous layers in various ways. The 3x3 convolutions detect structures in the image since 1x1 convolutions can't.

SquezeeNet architecture: SquezeeNet doesn't fully connect layers and it consists of 8 fire modules and a single convolution's layer as input and output. It uses Global Average Pooling, taking each channel from the previous convolution layer and builds an average over all values.

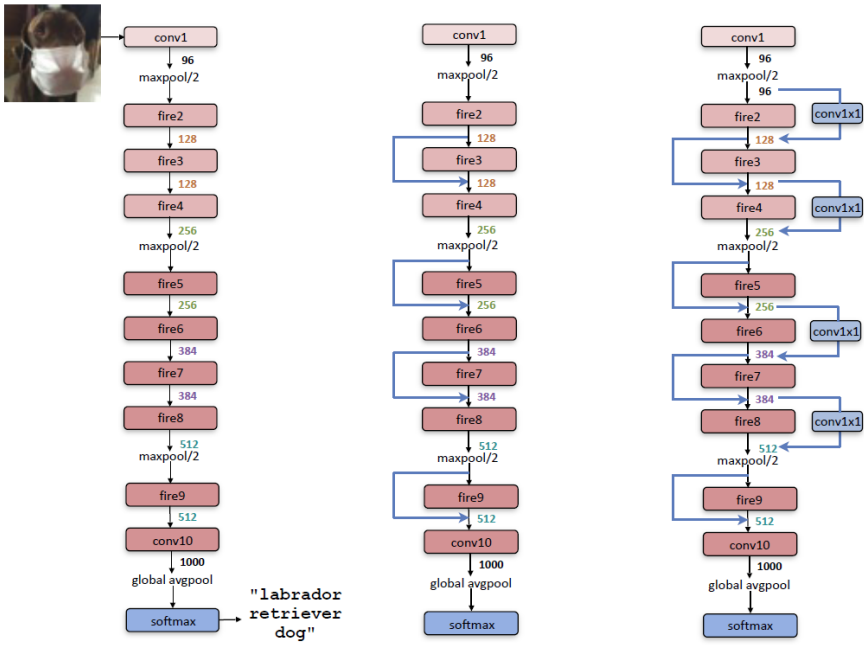


Figure A.4: SqueezeNet architecture. [95]

A.2.2 | ResNet

The core idea of ResNet (Residual Neural Network) is introducing skip connections (also called identity shortcut connection, represented in Figure A.5). The way this works is by adding the output of an earlier layer to a later layer in order to jump over some layers.

The vanishing of gradients problem makes deep neural networks hard to train, this happens because as the gradient is propagated back to earlier layers, repeated multiplications may turn the gradient too small, this results in a rapidly performance degradation. Skipping over layers helps avoiding the vanishing of gradients problem and improves the accuracy of the neural network. Having the skip connection allows the training of extremely deep neural networks, more than 150 layers, successfully and still being able to achieve a compelling performance [49].

This architecture is represented in Figure A.6.

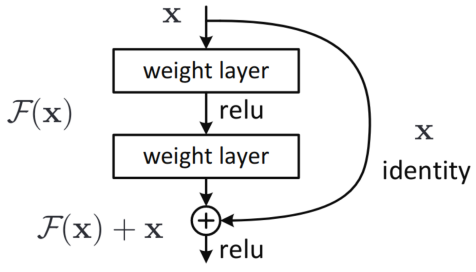


Figure A.5: Skipping connection example [49].

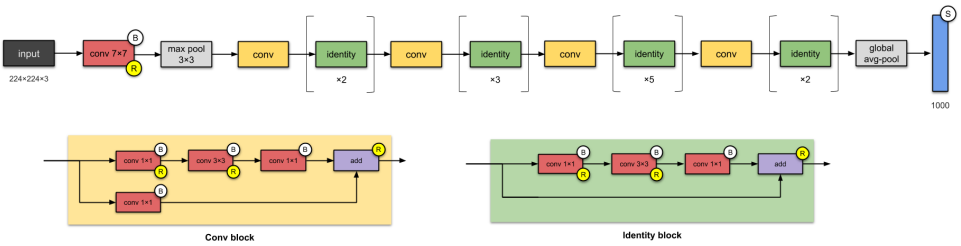


Figure A.6: ResNet architecture [51].

A.2.3 | InceptionV3

Initially named GoogLeNet, the Inception-v1 architecture was proposed by researchers of Google company and was the winner of the ILSVRC 2014 competition, making it historically significant in Convolutional Neural Networks.

This network, trained on the imageNet dataset, introduced inception modules (shown in Figure A.7) that allowed for a more efficient computation and deeper network.

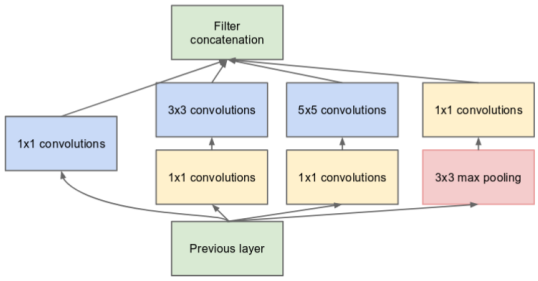


Figure A.7: Inception module [96].

The Inception architecture (Inception-v1) was improved by the introduction of batch normalization (Inception-v2) [2].

InceptionV3, is 48 layers deep and able to classify images into 1000 different categories. The improvement over its predecessors is the adding of factorization ideas (Figure A.8 shows an example of this).

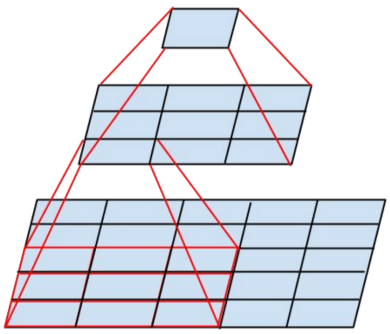


Figure A.8: Mini-network replacing the 5×5 convolutions (Example of factorization) [97].

This third interaction aims at factorizing convolutions, reducing the number of connections/parameters required while maintaining network efficiency. As an example, using a layer of 5x5 filter requires 5x5=25 parameters, this layer can be replaced by two 3x3 layers which reduce the number of parameters required by 28%, since 2x(3x3) = 18 parameters. Reducing the number of parameters required reduces the computational resources required and also prevents overfitting. This enables the network to go deeper [98].

The inceptionv3 architecture can be seen in the figure below.

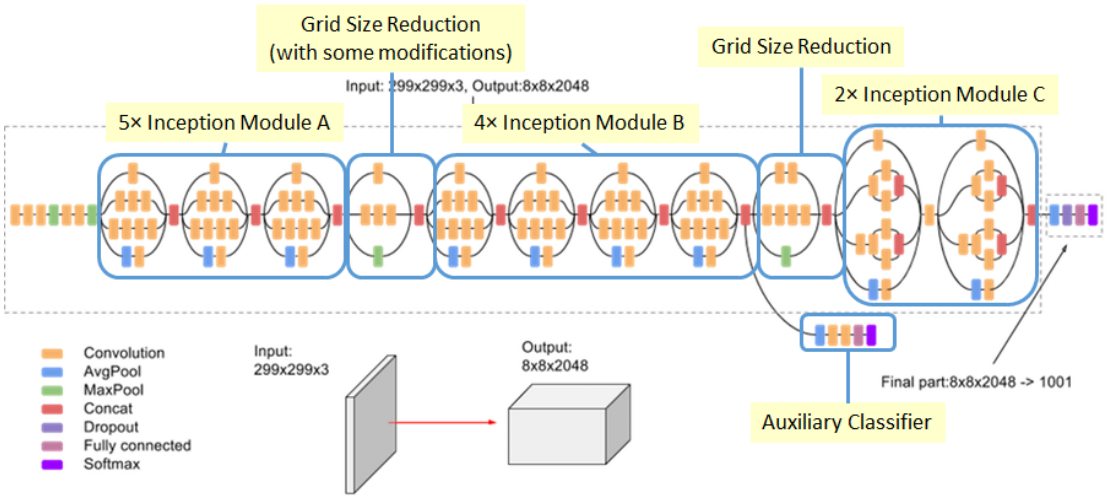


Figure A.9: InceptionV3 architecture [98].

Even though InceptionV4 [58] is already available it was not used in this work. The improvements over its predecessor are as follow:

1. Converting Inception modules to Residual Inception blocks.
2. Adding more Inception modules.
3. Adding a new type of Inception module (Inception-A) after the Stem module.

A.2.4 | DenseNet

Densely Connected Convolutional Networks aim at expanding the depth of deep convolutional networks by connecting each layer to every other layer, in a feed forward fashion (which can be seen in Figure A.10), this reduces the number of parameters required and alleviates the problem of the vanishing-gradients, while improving feature propagation (ensuring maximum information and gradient flow) and feature reuse which allows the learning of more compact and accurate models. This kind of neural network simplifies the connectivity pattern between layers introduced in other architectures (such as ResNets) [97].

The improved flow of information and gradients makes DenseNets easier to train, since each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision.

DenseNets scale naturally to hundreds of layers, while exhibiting no optimization difficulties.

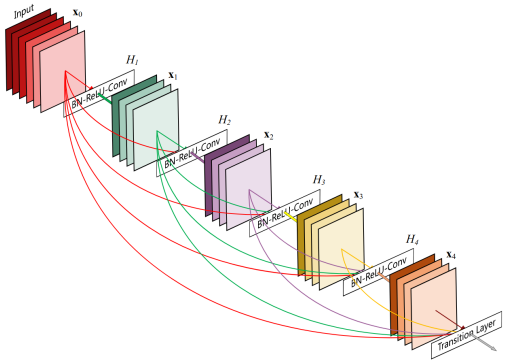


Figure A.10: A 5-layer dense block. Each layer takes all preceding feature-maps as input [97].

A.3 | Regression based algorithms for Object Detection

Regression based algorithms (also called single stage detectors) work differently than classification based algorithms. Instead of selecting multiple interesting parts of an image, they predict classes and bounding boxes for the entire image in one single run of the algorithm.

These algorithms are extremely fast but are not so accurate as classification based algorithms [46]. RetinaNet, YOLO and SSD are a few examples of object detection algorithms of this type.

A.3.1 | RetinaNet

RetinaNet is a one-stage object detector presented at the 2017 International Conference on Computer Vision by the Facebook AI Research.

In order to improve performance a loss function was implemented, called Focal Loss, allowing the network to focus more on difficult samples. With the loss function, alongside a one-stage network architecture, RetinaNet is able to achieve state-of-the-art performance in terms of accuracy and running time.

This neural network is essential composed of one backbone network and two subnetworks. The backbone network is called Feature Pyramid Net [99], built on top of ResNet, and has

the purpose of computing convolutional feature maps of an image. Both subnetworks serve different purposes, one is for object classification using the backbone network output and the other subnetwork is responsible for performing the bounding box regression using the backbone network output [46].

In Figure A.11 its observable the Feature Pyramid Network (FPN) on top of the convolutional neural network ResNet as a backbone network (a) to generate a rich convolutional feature pyramid (b). The class subnet (c) is for classifying anchor boxes, and the box subnet (d) is for regressing from anchor boxes to ground-truth object boxes.

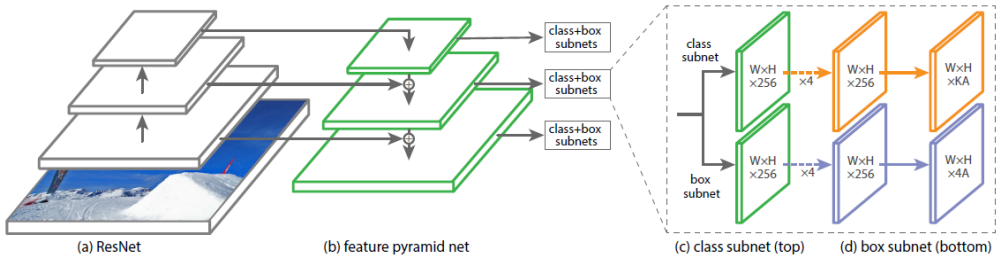


Figure A.11: RetinaNet architecture [46].

A.3.2 | YOLOv3

YOLOv3 (You Only Look Once version 3) is a state-of-the-art, real-time object detection that's in the third iteration of the original YOLO, it's extremely fast and accurate (on par with the accuracy of focal loss from RetinaNet, but 4 times faster). YOLO allows the user to tradeoff between speed and accuracy simply by changing the size of the model.

Compared to other classification networks that perform predictions multiple times for various regions in an image, YOLO architecture is more like a fully convolutional neural network do to the fact that it takes an image as input and passes it only once through the FCNN. The network divides the image into regions and predicts bounding boxes (weighted by predicted probabilities) and probabilities to each region, outputting a vector of bounding boxes and classes predictions.

YOLO works by dividing an image in an SxS grid and assuming B bounding boxes per grid. Each of the bounding box predicts 4 coordinates, object and class probabilities [7].

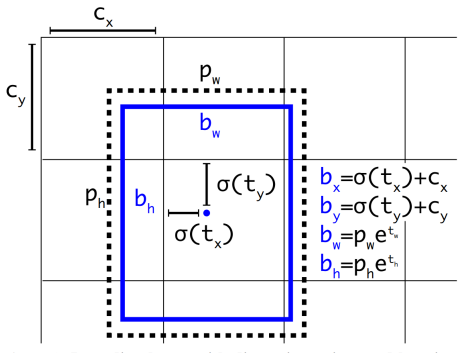


Figure A.12: Bounding box prediction: predicted box (blue), prior box (black dotted) [89].

YOLO image predictions are informed by global context in the image since it can look at the entire image at the test time. This gives it several advantages over classifier-based systems. In addition, this algorithm also uses an open source neural network called Darknet-53 for feature extraction, this neural network is written in C and CUDA and it supports CPU and GPU computation [89].

The full architecture of YOLOv3 is represented in Figure A.13.

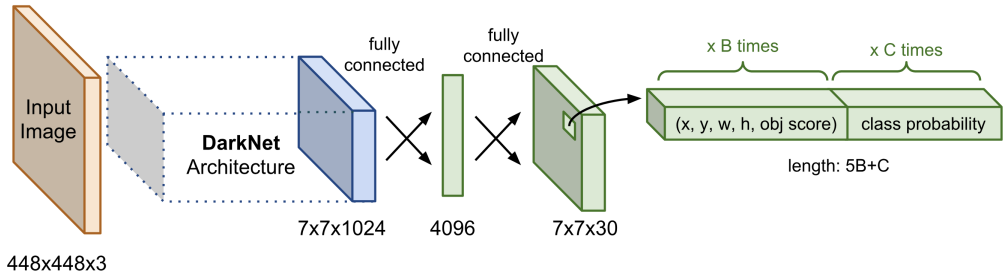


Figure A.13: The network architecture of YOLO base model [100].

A.3.3 | TinyYoloV3

TinyYOLOv3 is a smaller model of YOLOv3 that requires less computational resources since it doesn't occupy a large amount of memory, making it able to run in a smartphone. This model has a smaller number of convolutional layers, which improves the detection for small targets, therefore, it's a model best suited for constrained environments. In its architecture this network is composed of 7 convolutional layers and 6 pooling layers and can detect 80 different object categories. For complex scenes TinyYOLO is not accurate enough, however it is one of the fastest algorithms available [101].

A.3.4 | Single Shot MultiBox Detector (SSD)

SSD is a method for detecting objects in images using a single deep neural network. This Multibox detector discretizes the output space of bounding boxes into a set of default boxes over different aspect rations and scales per feature map location.

The base network of SSD is a VGG-16 network [59] followed by multibox convolutional layers. VGG-16 has the purpose of extracting the features for high quality image classification. The additional convolutional layers have the purpose of detecting objects, they are located at the end of the base network and decrease in size progressively, which helps with the detection of objects at multiple scales. The deep layers cover larger receptive fields and are helpful for larger objection detection, while the initial convolutional layers cover smaller receptive fields and are used for smaller objects detection [102].

The added auxiliary structure can be summarized in the following key points:

- **Multi-scale feature maps for detection**. These layers decrease in size progressively and allow predictions of detections at multiple scales.

- **Convolutional predictors for detection**. Each added feature layer can produce a fixed set of detection predictions using a set of convolutional filters.

- **Default boxes and aspect ratios**. They associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed.

The SSD architecture is represented in Figure A.14.

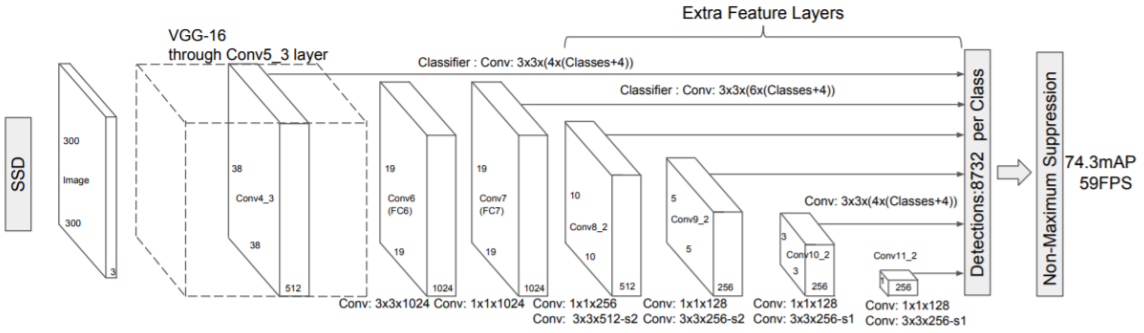


Figure A.14: SSD architecture [102].

The prediction of bounding boxes is done by multiple feature maps of different sizes that represent multiple scales. During prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. The network also combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train. The core of SSD is predicting category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps. To achieve high detection accuracy, SSD produces predictions of different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio. These design features lead to simple end-to-end training and high accuracy, even on low resolution input images, further improving the speed vs accuracy trade-off. This approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections [102].

A.4 | Classification Based Algorithms For Object Detection

Classification based algorithms work in two stages. Firstly, they select interesting regions from the image and secondly, they classify those regions using convolutional neural networks. The problem with this approach is that it can be extremely slow since a prediction is run for every selected region, however this approach is extremely accurate [46].

RCNN, Fast-RCNN and Faster-RCNN are some types of classification based algorithms.

A.4.1 | R-CNN Models Summary

In the picture below a compact summary of all of the R-CNN models is illustrated.

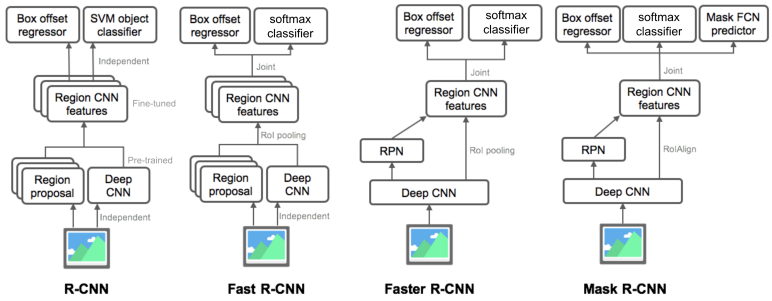


Figure A.15: R-CNN model family summary [103].

- **R-CNN**

The principal idea behind Region-based Convolutional Networks (R-CNN) can be split into two steps. In the first step the network identifies a number of regions of interest (bounding-box object region candidate) using a selective search method [103], which is a common algorithm to provide region proposals that can potentially contain objects [104].

In the second step it extracts CNN features from each region independently for the classification.

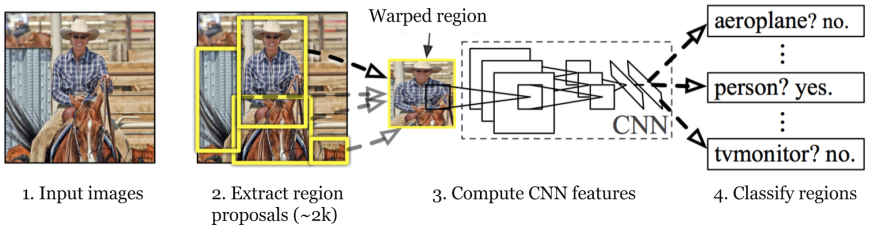


Figure A.16: R-CNN architecture [103].

- **Fast R-CNN**

The idea behind Fast-RCNN [105] is, as the name implies, to make R-CNN faster. In order to achieve this, the training procedure was improved by unifying three independent models into one jointly trained framework and increasing shared computation results.

In this new improved network, the CNN feature vectors are not extracted independently for each region proposal, instead this model aggregates them into one CNN forward pass over the entire image and the region proposals share the feature matrix. This feature matrix is then branched to be used for learning the object classifier and the bounding-box regression. In short, computation sharing improves the speed of R-CNN [103].

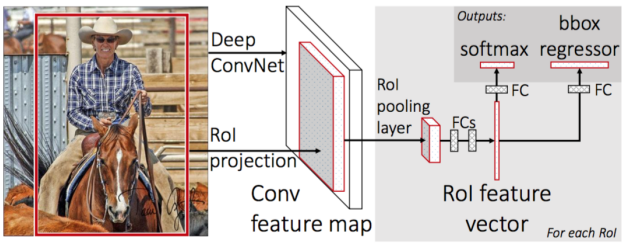


Figure A.17: Fast R-CNN architecture [103].

- **Faster R-CNN**

The Faster R-CNN [106] improves upon the previous considered solutions since it integrates the region proposal algorithm directly into the CNN model. It can be seen as a single, unified model composed of a region proposal network and fast R-CNN with shared convolutional feature layers [103].

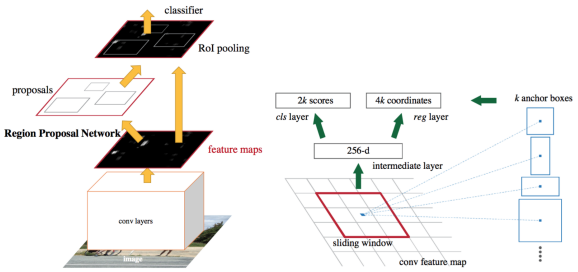


Figure A.18: Faster R-CNN architecture [103].

- **Mask R-CNN**

The final model of the R-CNN family, Mask R-CNN [107], extends faster R-CNN to pixel-level image segmentation by decoupling the classification and the pixel-level mask prediction tasks. It adds a third branch for predicting an object mask in parallel with existing branches for classification and localization, based of the Faster R-CNN framework.This new mask branch predicts a segmentation mask in a pixel-to-pixel manner.

Mask R-CNN improves the region of interest pooling layer because pixel-level segmentation requires much more fine-grained alignment than bounding boxes. This allows the region of interest to more precisely map regions of the original image [103].

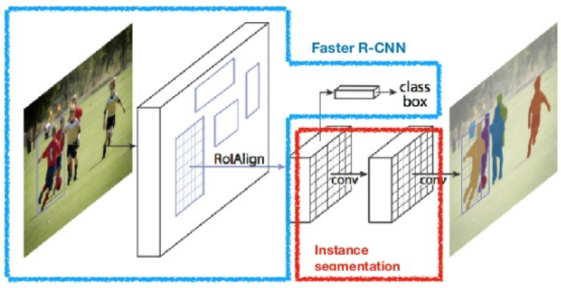


Figure A.19: Mask R-CNN is a Faster R-CNN model with image segmentation [103].

Bibliography

[1] Jin Zhang. *The Information Retrieval Series*. 2008, p. 300. ISBN: 9783540751472.

[2] Ricardo Ferreira Ribeiro and Ieeta Deti. "Object Recognition with Convolutional Neural Networks". In: ().

[3] Ricardo Ribeiro and Alina Trifan. "UA . PT Bioinformatics at ImageCLEF 2020 : Lifelog Moment Retrieval Web based Tool". In: (2020), pp. 22–25.

[4] Ricardo Ribeiro, António J.R. Neves, and José Luis Oliveira. "UA.Pt bioinformatics at ImageClef 2019: Lifelog moment retrieval based on image annotation and natural language processing". In: *CEUR Workshop Proceedings* 2380 (2019), pp. 9–12. ISSN: 16130073.

[5] Van-tu Ninh et al. "Overview of ImageCLEF Lifelog 2020 : Lifelog Moment Retrieval and Sport Performance Lifelog". In: (2020), pp. 22–25.

[6] Wikipedia. *F1 score*. 2020. URL: `https://www.wikiwand.com/en/F1_score%20https://en.wikipedia.org/wiki/F1_score` (visited on 09/19/2020).

[7] Shivang Agarwal and Jean Ogier. "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks". In: (2019). arXiv: `arXiv:1809.03193v2`.

[8] Aastha Tiwari, Anil Kumar Goswami, and Mansi Saraswat. "Feature Extraction for Object Recognition and Image Classification". In: *International Journal of Engineering Research & Technology (IJERT)* 2.10 (2013), pp. 1238–1246.

[9] *The Computer Vision Pipeline, Part 4: feature extraction | Manning*. URL: `https://freecontent.manning.com/the-computer-vision-pipeline-part-4-feature-extraction/` (visited on 02/18/2020).

[10] Xin Feng et al. "Computer vision algorithms and hardware implementations: A survey". In: *Integration* 69.August (2019), pp. 309–320. ISSN: 01679260.

[11] Limarc Ambalina. *What is Image Annotation? – An Intro to 5 Image Annotation Services - By Limarc Ambalina*. URL: `https://hackernoon.com/what-is-image-annotation-an-intro-to-5-image-annotation-services-yt6n3xfj` (visited on 01/22/2020).

[12] Jason Brownlee. *A Gentle Introduction to Object Recognition With Deep Learning*. URL: `https://machinelearningmastery.com/object-recognition-with-deep-learning/` (visited on 01/22/2020).

[13] Thomas S. Huang. "Can the world-wide web bridge the semantic gap?" In: *Image and Vision Computing* 30.8 (2012), pp. 463–464. ISSN: 02628856.

[14] *Cloud Vision API | Cloud Vision API | Google Cloud*. URL: https://cloud.google.com/vision/docs/drag-and-drop%20https://cloud.google.com/vision/docs/reference/rest/ (visited on 09/11/2020).

[15] MathWorks. *What Is Artificial Intelligence? | KurzweilAI*. URL: https://www.mathworks.com/discovery/artificial-intelligence.html%20http://www.kurzweilai.net/what-is-artificial-intelligence (visited on 03/06/2020).

[16] MathWorks. *What Is a Machine Learning? - MATLAB & Simulink*. URL: https://www.mathworks.com/discovery/machine-learning.html (visited on 03/12/2020).

[17] MathWorks. *What Is Deep Learning? | How It Works, Techniques & Applications - MATLAB & Simulink*. 2019. URL: https://www.mathworks.com/discovery/deep-learning.html (visited on 03/05/2020).

[18] MathWorks. *What Is a Neural Network? - MATLAB & Simulink*. URL: https://www.mathworks.com/discovery/neural-network.html (visited on 03/06/2020).

[19] Arthur Arnx. *First neural network for beginners explained (with code) | by Arthur Arnx | Towards Data Science*. URL: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf (visited on 09/20/2020).

[20] Armaan Merchant. *Neural Networks Explained – Data Driven Investor – Medium*. 2018. URL: https://medium.com/datadriveninvestor/neural-networks-explained-6e21c70d7818 (visited on 03/04/2020).

[21] Adrien Kaiser. *What is Computer Vision? | Hayo*. URL: https://hayo.io/computer-vision/ (visited on 01/22/2020).

[22] Jason Brownlee. *A Gentle Introduction to Computer Vision*. URL: https://machinelearningmastery.com/what-is-computer-vision/ (visited on 01/22/2020).

[23] Tsung Yi Lin et al. "Microsoft COCO: Common objects in context". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8693 LNCS.PART 5 (2014), pp. 740–755. ISSN: 16113349. arXiv: 1405.0312.

[24] Y. Takamitsu and Y. Orita. "Effect of glomerular change on the electrolyte reabsorption of the renal tubule in glomerulonephritis (author's transl)". In: *Japanese Journal of Nephrology* 20.11 (1978), pp. 1221–1227. ISSN: 03852385.

[25] Connecting Language et al. "Visual Genome - connected language and Vision using crowdsourced dense image annotations". In: (2015).

[26] Alina Kuznetsova et al. “The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale”. In: (2018), pp. 1–20. arXiv: 1811.00982.

[27] Mark Everingham et al. “The pascal visual object classes (VOC) challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. ISSN: 09205691.

[28] Ivan Culjak et al. “A brief introduction to OpenCV”. In: *MIPRO 2012 - 35th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings* (2012), pp. 1725–1730.

[29] OpenCV Team. *About*. URL: https://opencv.org/about/ (visited on 01/23/2020).

[30] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Tech. rep.

[31] John D. Dignam et al. “Eukaryotic gene transcription with purified components”. In: *Methods in Enzymology* 101.C (1983), pp. 582–598. ISSN: 15577988.

[32] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. 2008. (Visited on 01/23/2020).

[33] BoofCV Team. *BoofCV*. URL: https://boofcv.org/index.php?title=Main%7B%5C_%7DPage (visited on 01/23/2020).

[34] Jian Guo et al. “GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing”. In: (2019), pp. 1–6. arXiv: 1907.04433.

[35] Paperswithcode. *COCO test-dev Leaderboard | Papers with Code*. URL: https://paperswithcode.com/sota/object-detection-on-coco (visited on 03/06/2020).

[36] Paperswithcode. *ImageNet Leaderboard | Papers with Code*. URL: https://paperswithcode.com/sota/image-classification-on-imagenet (visited on 03/06/2020).

[37] Yudong Liu et al. “CBNet: A Novel Composite Backbone Network Architecture for Object Detection”. In: (2019). arXiv: 1909.03625.

[38] Mingxing Tan, Ruoming Pang, and Quoc V. Le. “EfficientDet: Scalable and Efficient Object Detection”. In: (2019). arXiv: 1911.09070.

[39] Shifeng Zhang et al. “Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection”. In: 2 (2019). arXiv: 1912.02424.

[40] Ross Girshick et al. *Detectron*. 2018. URL: https://github.com/facebookresearch/detectron (visited on 03/09/2020).

[41] Yanghao Li et al. “Scale-Aware Trident Networks for Object Detection”. In: (2019). arXiv: 1901.01892.

[42] Dhruv Mahajan et al. “Exploring the Limits of Weakly Supervised Pretraining”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11206 LNCS (2018), pp. 185–201. ISSN: 16113349. arXiv: 1805.00932.

[43] Qijie Zhao et al. “M2Det: A Single-Shot Object Detector Based on Multi-Level Feature Pyramid Network”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), pp. 9259–9266. ISSN: 2159-5399. arXiv: 1811.04533.

[44] Zhaowei Cai and Nuno Vasconcelos. “Cascade R-CNN: Delving into High Quality Object Detection”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), pp. 6154–6162. ISSN: 10636919. arXiv: 1712.00726.

[45] Jiaqi Wang et al. “Region proposal by guided anchoring”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June (2019), pp. 2960–2969. ISSN: 10636919. arXiv: 1901.03278.

[46] Tsung Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision* 2017-October (2017), pp. 2999–3007. ISSN: 15505499. arXiv: 1708.02002.

[47] Abhinav Shrivastava et al. *Beyond Skip Connections: Top-Down Modulation for Object Detection*. 2016. arXiv: 1612.06851 [cs.CV].

[48] Seung Wook Kim et al. “Parallel Feature Pyramid Network for Object Detection”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11209 LNCS (2018), pp. 239–256. ISSN: 16113349.

[49] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December (2016), pp. 770–778. ISSN: 10636919. arXiv: 1512.03385.

[50] Saining Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January (2017), pp. 5987–5995. arXiv: 1611.05431.

[51] Raimi Karim. *Illustrated: 10 CNN Architectures*. URL: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d (visited on 03/12/2020).

[52] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. ISSN: 15731405. arXiv: 1409.0575.

[53] Qizhe Xie et al. “Self-training with Noisy Student improves ImageNet classification”. In: (2019). arXiv: 1911.04252.

[54] Alexander Kolesnikov et al. *Large Scale Learning of General Visual Representations for Transfer*. 2019. arXiv: 1912.11370 [cs.CV].

[55] Hugo Touvron et al. *Fixing the train-test resolution discrepancy*. 2019. arXiv: 1906.06423 [cs.CV].

[56] Cihang Xie et al. *Adversarial Examples Improve Image Recognition*. 2019. arXiv: 1911.09665 [cs.CV].

[57] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019. arXiv: 1905.11946 [cs.LG].

[58] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV].

[59] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[60] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2018. arXiv: 1801.04381 [cs.CV].

[61] Mingxing Tan et al. *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. 2018. arXiv: 1807.11626 [cs.CV].

[62] Mingxing Tan. *Google AI Blog: EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling*. URL: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html (visited on 09/20/2020).

[63] Moses and John Olafenwa. *ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities*. Mar. 2018–. URL: https://github.com/OlafenwaMoses/ImageAI.

[64] Chollet François. "Keras: The Python Deep Learning library". In: *Keras.Io* (2015). ISSN: 00046256.

[65] Sonit Singh. *Natural Language Processing for Information Extraction*. 2018. arXiv: 1807.02383 [cs.CL].

[66] Diksha Khurana et al. "Natural Language Processing : State of The Art , Current Trends and Challenges Natural Language Processing : State of The Art , Current Trends and Challenges Department of Computer Science and Engineering Manav Rachna International University , Faridabad-". In: *arXiv preprint arXiv* August 2017 (2018).

[67] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. "A Survey of the Usages of Deep Learning in Natural Language Processing". In: XX.X (2018), pp. 1–22. arXiv: 1807.10854.

[68] Analytics Vidhya. *Understanding Word Embeddings: From Word2Vec to Count Vectors*. 2017. URL: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/ (visited on 02/06/2020).

[69] Murat Mustafa. *GloVE | Mustafa Murat ARAT*. URL: https://mmuratarat.github.io/2020-03-20/glove (visited on 05/13/2020).

[70] Jason Brownlee. *What are word embeddings for text?* 2017. URL: https://machinelearningmastery.com/what-are-word-embeddings/ (visited on 02/06/2020).

[71] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* (2013), pp. 1–12. arXiv: 1301.3781.

[72] Yuxuan Wang et al. "From static to dynamic word representations: a survey". In: *International Journal of Machine Learning and Cybernetics* 11.7 (2020), pp. 1611–1630. ISSN: 1868808X.

[73] David Batista. *[oorsig] Language Models and Contextualised Word Embeddings*. 2018. URL: http://www.davidsbatista.net/blog/2018/12/06/Word_Embeddings/ (visited on 02/10/2020).

[74] A.I. Wiki. *A Beginner's Guide to Word2Vec and Neural Word Embeddings | Skymind*. URL: https://pathmind.com/wiki/word2vec%20https://skymind.ai/wiki/word2vec (visited on 02/08/2020).

[75] *Word2Vec Explained Easily - InsightsBot*. URL: http://www.insightsbot.com/word2vec-explained-easily/ (visited on 02/08/2020).

[76] Gabriel Mordecki. *word embeddings transform text numbers*. 2017.

[77] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global vectors for word representation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. Association for Computational Linguistics (ACL), 2014, pp. 1532–1543. ISBN: 9781937284961.

[78] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *arXiv preprint arXiv:1607.04606* (2016).

[79] Oren Melamud, Jacob Goldberger, and Ido Dagan. "context2vec: Learning generic context embedding with bidirectional LSTM". In: *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings* (2 016), pp. 51–61.

[80] Matthew E. Peters et al. "Deep contextualized word representations". In: *Proc. of NAACL*. 2018.

[81] Spacy. *spaCy 101: Everything you need to know · spaCy Usage Documentation*. 2017. URL: https://spacy.io/usage/spacy-101 (visited on 02/10/2020).

[82] Edward Loper and Steven Bird. “Nltk”. In: March (2002), pp. 63–70.

[83] Christopher Manning et al. “The Stanford CoreNLP Natural Language Processing Toolkit”. In: (2015), pp. 55–60.

[84] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[85] *TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation*. URL: https://textblob.readthedocs.io/en/dev/ (visited on 07/28/2020).

[86] Alan Akbik, Duncan Blythe, and Roland Vollgraf. “Contextual String Embeddings for Sequence Labeling”. In: *COLING 2018, 27th International Conference on Computational Linguistics*. 2018, pp. 1638–1649.

[87] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. “Polyglot: Distributed Word Representations for Multilingual NLP”. In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 183–192.

[88] Bolei Zhou et al. “Places: A 10 Million Image Database for Scene Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1452–1464. ISSN: 01628828.

[89] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: (2018). arXiv: 1804.02767.

[90] Nguyen Khang Le et al. “Lifelog moment retrieval with advanced semantic extraction and flexible moment visualization for exploration”. In: *CEUR Workshop Proceedings* 2380 (2019), pp. 9–12. ISSN: 16130073.

[91] Vikas Gupta. *Understanding Feedforward Neural Networks*. 2017. URL: https://www.learnopencv.com/understanding-feedforward-neural-networks/.

[92] Kjell Magne Fauske. *What Is Deep Learning? | How It Works, Techniques & Applications - MATLAB & Simulink*. 2019. URL: https://www.mathworks.com/discovery/deep-learning.html (visited on 03/05/2020).

[93] Keiron O’Shea and Ryan Nash. “An Introduction to Convolutional Neural Networks”. In: November (2015). arXiv: 1511.08458.

[94] Forrest N. Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”. In: (2016), pp. 1–13. arXiv: 1602.07360.

[95] Sik-Ho Tsang. *Review: SqueezeNet (Image Classification) - Towards Data Science*. URL: https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a (visited on 01/23/2020).

[96] Bharath Raj. *A Simple Guide to the Versions of the Inception Network*. URL: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 (visited on 01/23/2020).

[97] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December (2016), pp. 2818–2826. ISSN: 10636919. arXiv: 1512.00567.

[98] Sik-Ho Tsang. *Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015*. URL: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c (visited on 01/23/2020).

[99] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2016. arXiv: 1612.03144 [cs.CV].

[100] Lilian Weng. "Object Detection Part 4: Fast Detection Models". In: *lilianweng.github.io/lil-log* (2018).

[101] Zhang Yi, Shen Yongliang, and Zhang Jun. "An improved tiny-yolov3 pedestrian detection algorithm". In: *Optik* 183.January (2019), pp. 17–23. ISSN: 00304026.

[102] Wei Liu et al. "SSD Net". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9905 LNCS (2016), pp. 21–37. ISSN: 16113349. arXiv: arXiv:1512.02325v5.

[103] Lilian Weng. "Object Detection for Dummies Part 3: R-CNN Family". In: *lilianweng.github.io/lil-log* (2017).

[104] Lilian Weng. "Object Detection for Dummies Part 1: Gradient Vector, HOG, and SS". In: *lilianweng.github.io/lil-log* (2017).

[105] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[106] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. arXiv: 1506.01497 [cs.CV].

[107] Kaiming He et al. *Mask R-CNN*. 2017. arXiv: 1703.06870 [cs.CV].