Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática
2020

Júlio Miguel Braz da Costa Silva

Recuperação e Identificação de Momentos em Imagens/Vídeos

Recovery and Identification of Moments in Images / Videos

DOCUMENTO PROVISÓRIO

Júlio Miguel Braz da Costa Silva

Recuperação e Identificação de Momentos em Imagens/Vídeos

Recovery and Identification of Moments in Images / Videos

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 e coorientação de Ricardo Ribeiro, Investigador no Instituto de Engenharia Eletrónica e Telemática de Aveiro da Universidade de Aveiro.

o júuri / 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 graças à facilidade de acesso a *smartphones*. Isso coloca a 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 alguns 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 LMRT do ImageClef, 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 do texto enquanto calcula um valor de similaridade com outros rótulos extraídos da 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.

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

Abstract

In our modern society almost anyone is able to capture moments and record events thanks to the ease accessibility to smartphones. This brings 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 diseases in patients or even tracking people with infectious diseases.

With so much data being created everyday, answering the question becomes complex. There is no stream lined approach to solve the problem of moment localization in a big dataset of images and investigation into this problem have only started a few years ago. ImageClef is one competition where some 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 LMRT subtask 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 corpous of text without any sort of user interaction 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.

Contents

List of Figures vii

List of Tables ix

Acronyms xi

1 Introduction 1
- 1.1 Context and Motivation . . . 1
- 1.2 Challenges . . . 2
- 1.3 Objectives . . . 2
- 1.4 Contributions . . . 3
- 1.5 Document Structure . . . 3

2 ImageCLEF 5
- 2.1 The ImageCLEF challenge . . . 5
- 2.2 The Tasks . . . 6
- 2.3 The concept of lifelogging . . . 6
- 2.4 ImageCLEFlifelog . . . 7
  - 2.4.1 SubTask: Lifelog Moment Retrieval . . . 7
  - 2.4.2 Dev Topic example . . . 8
  - 2.4.3 Test Topic example . . . 9
  - 2.4.4 Evaluation Methodology . . . 9

3 Image/Video Feature Extraction 11
- 3.1 Fundamental Concepts . . . 12
  - 3.1.1 Artificial Intelligence . . . 12
  - 3.1.2 Machine Learning . . . 12
  - 3.1.3 Deep Learning . . . 12
  - 3.1.4 Computer Vision . . . 13
  - 3.1.5 Image Annotation and Classification . . . 13
  - 3.1.6 Object Detection, Segmentation and Recognition . . . 14
  - 3.1.7 Features and Feature Space . . . 14
  - 3.1.8 Object . . . 14
  - 3.1.9 Image Description . . . 14
  - 3.1.10 Datasets With Common Objects . . . 15
- 3.2 Computer Vision Libraries . . . 15
  - 3.2.1 OpenCV . . . 15
  - 3.2.2 VLFeat . . . 16
  - 3.2.3 BOOFCV . . . 16
  - 3.2.4 GluonCV . . . 17

3.3 Neural Networks . . . 17
3.3.1 Neural Network Training . . . 18
3.3.2 Types of Neural Networks architectures . . . 18
3.4 CNNs architectures For Image Classification . . . 20
3.4.1 SquezeeNet . . . 20
3.4.2 ResNet . . . 22
3.4.3 InceptionV3 . . . 23
3.4.4 DenseNet . . . 25
3.5 Regression based algorithms for Object Detection . . . 25
3.5.1 RetinaNet . . . 25
3.5.2 YOLOv3 . . . 26
3.5.3 TinyYoloV3 . . . 27
3.5.4 Single Shot MultiBox Detector (SSD) . . . 27
3.6 Classification Based Algorithms For Object Detection . . . 28
3.6.1 R-CNN Models Summary . . . 29
3.6.2 R-CNN . . . 29
3.6.3 Fast R-CNN . . . 29
3.6.4 Faster R-CNN . . . 30
3.6.5 Mask R-CNN . . . 30
3.7 State-Of-The-Art . . . 31
3.7.1 COCO Test-Dev . . . 32
3.7.2 ImageNet . . . 34

4 Information Extraction From Text **37**
4.1 Natural Language Processing . . . 38
4.1.1 Important NLP Terminologies . . . 38
4.1.2 Core Areas . . . 39
4.1.3 Application Areas . . . 39
4.2 Numerical Representation of Text . . . 39
4.2.1 Word Embeddings . . . 39
4.3 Static Word Embedding Models . . . 40
4.3.1 Word2Vec . . . 40
4.3.2 GloVe . . . 41
4.3.3 FastText . . . 42
4.4 Contextualized Word Embedding Models . . . 42
4.4.1 Context2vec . . . 42
4.4.2 ELMo . . . 43
4.5 Available NLP libraries . . . 44
4.5.1 SpaCy . . . 44
4.5.2 Natural Language ToolKit . . . 44
4.5.3 Stanford Core NLP . . . 45
4.5.4 Gensim . . . 45
4.5.5 Uncommon Libraries . . . 45

5 Image Processing **47**
5.1 Test Runs . . . 47
5.1.1 Image Recognition test runs . . . 47

5.1.2 Object Detection test runs . . . 49
5.1.3 Object Detection Word Clouds Generation Test Run . . . 53
5.2 Example of a Raw Retrieval System . . . 54
5.3 Scene Recognition . . . 55
5.4 Run 1 and Run 2 . . . 56
5.5 Processing the Imageclef Dataset . . . 56

6 Text Processing and Image Retrieval 59
6.1 Word Extraction . . . 59
6.2 Retrieval Stage . . . 61
6.2.1 Retrieving Images According to the Similarity Between Words . . . 61
6.2.2 Calculation of Similarity Scores . . . 63
6.2.3 Run 1 . . . 63
6.2.4 Run 2 . . . 63
6.2.5 Confidence Score Computation Equations . . . 64
6.3 System Workflow Architecture Diagram . . . 65

7 Results 67
7.1 System Fine-Tuning Using The Dev Topics . . . 67
7.2 System Performance Example . . . 68
7.2.1 Run 1 and Run 2 Image Retrieval Example . . . 68
7.2.2 Top 3 Retrieved Image on Run 1 . . . 69
7.2.3 Top 3 Retrieved Images on Run 2 . . . 69
7.2.4 Topic 9 Performance Analysis . . . 70
7.3 Achieved Overall Performance Results . . . 70
7.3.1 Overall Performance Analysis . . . 71

8 Conclusions 73
8.1 Discussion . . . 73
8.1.1 System Advantages . . . 73
8.1.2 System Disadvantages . . . 73
8.2 Future Work . . . 74

List of Figures

2.1 Excerpt of the ground truth for the dev topic 1. 8
2.2 Example of an image from the ground truth of the topic 1. 9

3.1 Feature extraction from an image. [8] 11
3.2 Generic picture of a family having a picnic. 15
3.3 Typical neural network architecture. [31] 17
3.4 Operations done by a neuron. 18
3.5 Example of a Feedforward Neural Network with one hidden layer (with 5 neurons) [34]. 18
3.6 CNN architecture 19
3.7 SqueezeNet fire module. [36] 21
3.8 SqueezeNet architecture. [37] 22
3.9 Skipping connection example.[38] 22
3.10 ResNet Architecture.[39] 23
3.11 Inception Module. [40] 23
3.12 Mini-network replacing the 5×5 convolutions (Example of factorization). [41] . 23
3.13 InceptionV3 architecture.[42] 24
3.14 A 5-layer dense block. Each layer takes all preceding feature-maps as input. [41] 25
3.15 RetinaNet architecture.[44] 26
3.16 Bounding Box Prediction : Predicted Box (Blue), Prior Box (Black Dotted).[46] 26
3.17 The network architecture of YOLO base model.[47] 27
3.18 SSD architecture. [50] 28
3.19 R-CNN model family summary. [51] 29
3.20 R-CNN architecture. [51] 29
3.21 Fast R-CNN architecture. [51] 30
3.22 Faster R-CNN architecture. [51] 30
3.23 Mask R-CNN is a Faster R-CNN model with image segmentation. [51] 30
3.24 Object Detection on COCO test-dev benchmark .[56] 31
3.25 Image Classification on ImageNet benchmark. [57] 31
3.26 CBNet Architecture for object detection. 33
3.27 ResNeXt architecture. [39] 33
3.28 Noisy Student Method. [71] 35
3.29 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. [75] 35
3.30 EfficientNet-B0 architecture representation. 36

4.1 Natural Language Processing [78]. 37

4.2 Classification of NLP. [79] . . . 38
4.3 Example of text representation by one-hot vector. . . . 40
4.5 CBOW model and Skip-Gram model. [84] . . . 41
4.6 A 2D illustration of context2vec's embedded space and similarity metrics. Triangles and circles denote sentential context embeddings and target word embeddings, respectively [90]. . . . 43
4.7 Closest target words to various sentential contexts, illustrating context2vec's sensitivity to long range dependencies, and both sides of the target word [90]. . . . 43
4.8 Nearest neighbors to "play" using GLoVe and context embeddings from a biLM [91]. . . . 44

5.1 Image recognition test runs. . . . 48
5.2 Available labels for detection. . . . 49
5.3 Pictures used for test runs. . . . 49
5.4 Test run 1 with RetinaNet; a) Analysed picture with detections; b) Achieved performance on detections. . . . 50
5.5 Test run 1 with YOLOv3; a) Analysed picture with detections; b) Achieved performance on detections. . . . 50
5.6 Test run 1 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections. . . . 50
5.7 Test run 2 with RetinaNet; a) Analysed picture with detections; b) Achieved performance detections. . . . 51
5.8 Test run 2 with YoloV3 model; a) Analysed picture with detections; b) Achieved performance on detections. . . . 51
5.9 Test run 2 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections. . . . 51
5.10 Test run 3 with RetinaNet; a) Analysed picture with detections; b) Achieved performance detections. . . . 52
5.11 Test run 3 with YoloV3 model; a) Analysed picture with detections; b) Achieved performance on detections. . . . 52
5.12 Test run 3 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections. . . . 52
5.13 Used images for word cloud generation. . . . 53
5.14 Generated Word Clouds; a) Yolo word cloud; b) RetinaNet word cloud; c) TinyYolo word cloud . . . 53
5.15 System capable of detecting specific user inputted labels in multiple images. . . . 54
5.16 Retrieved images for the label "cup" with "40%" threshold. . . . 55
5.17 Example of a scene recognition; a) Picture 20161004_213423_000.jpg from the imageclef dataset; b) Scene recognition model output for that image. . . . 55
5.18 Fully processed image with YOLOv3 and PLACES365. . . . 56
5.19 Fully processed image with ResNeXt-101 and PLACES365. . . . 57

6.1 Linguistic annotations generated by the SpaCy library [92] for the narrative of the topic 6 of the test topics. . . . 59
6.2 Test topic number 7. . . . 61
6.3 System Architecture . . . 65

7.1 Examples of some different results with the fine-tuning of the weight distribution. 67
7.2 Achieved results on topic 9 of the test topics. 68
7.3 Top 3 retrieved pictures for topic 9 on run 1 69
7.4 Top 3 retrieved pictures for topic 9 on run 2 69

8.2 Examples of google cloud vision API extracted labels [103] 74

List of Tables

3.1 COCO Test-Dev Benchmarks. 32
3.2 ImageNet Benchmarks. 34

7.1 Results obtained in 2019 from UA.PT Bioinformatics [3] and the best team [102]. 70
7.2 Results obtained in 2020 from UA.PT Bioinformatics [4] and the best team [5]. 71

Acronyms

AI Artificial Inteliggence.

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CV Computer Vision.

IE Information Extraction.

LMRT Life Moment Retrieval Task.

NLP Natural Language Processing.

ResNet Residual Network.

SSD Single Shot Detection.

YOLO You Only Look Once.

CHAPTER 1

Introduction

This chapter gives an introduction to the surrounding theme addressed in this thesis. In that sense, firstly the contextualization of the theme and the respective motivation will be presented. The different challenges, the objectives that are intended to be reach and the contributions given to the community are also described. Finally the document structure and organization is explained.

1.1 | Context and Motivation

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. Pictures, images and photos 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 great examples of the usefulness of lifelogging is using it as 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 problems related to creating, compressing, storing, transmitting, rendering and protecting image data are already 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 by someone 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.

1.2 | 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, indoor or outdoors, activities happening in the photo, people, etc. However in order to do this, many different algorithms have to be implemented like object detection, activity recognition, scene recognition and others. The usage of so many 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, a dataset of 200.000 images of the same resolution would require approximately 2 days.

In order to teach the computer to understand text there has to be an underlying understanding of the language. Natural Language Processing algorithms have to be implemented, however these algorithms only allow the computer to extract the basics like which words are verbs, adjectives, nouns, etc. With this in mind, another algorithm has to be fully written from scratch capable of using that data to extract activities, locations, people, dates, indoor or outdoor, etc from a corpus of text.

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.

1.3 | Objectives

The main objective of this work is to develop an automatic image retrieval system capable of participating in the ImageCLEF LMRT-subtask challenge (described in chapter 2). In order to achieve this objective the following tasks have to be accomplished:

- Study on the state of the art of image processing and text processing algorithms.
- Choice of the main algorithms to be used.
- Code of an algorithm capable of processing images and extract relevant features.
- Code of an algorithm capable of processing text and extract relevant features.
- Code of an algorithm able to compare the extracted features from the images with the extracted features from the text and capable of associating images to text.

- Code of an algorithm capable of calculating the F1@score of the final results in order to test the system.
- Code of a batch script in order to make the system run with one click in order to facilitate the process.

1.4 | 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.

1.5 | Document Structure

This document has a total of 8 chapters that are divided accordingly:

- Chapter 1 presents the context and motivation along with the challenges and objectives.
- Chapter 2 discusses the imageCLEF challenge.
- Chapter 3 provides a survey on the subject of feature extraction from images and video.
- Chapter 4 addresses the thematic of natural language processing and word embeddings.
- Chapter 5 provides an overview on how the image processing stage of the automatic system was built.
- Chapter 6 explains how the system is capable of word extraction and categorization and how the system is capable of image retrieval.
- Chapter 7 presents the achieved results in the imageclef challenge.
- Chapter 8 describes the conclusions taken from the development of the work and provides some ideas for future work and investigations.

CHAPTER 2

ImageCLEF

This chapter aims at describing the ImageCLEF 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 LMRT subtask 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 or 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. [3] [4]

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 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 [3] 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 not explain how it works. However, for further readings the article "UA.PT Bioinformatics at ImageCLEF 2020: Lifelog Moment Retrieval Web based Tool" [4] explains in detail how the interactive system operates.

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 new 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 . However, in this work only the images, the visual concepts and the semantic content of the dataset were used. [4]

Firstly the organizers release the dev topics and the image dataset with 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 it is possible to know which pictures should 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 more pictures retrieved that correspond to the ground truth for the evaluation phase.

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 defined in a series of 10 textual query topics, an example is given below:

- Topic 1

  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."

2.4.2.1 | Example of the ground truth

The ground truth is given as .txt file with the following format : [topic number, image name, cluster]. The cluster number is used to calculate the F1@score which will be explained in more detail in 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 2.1: Excerpt of the ground truth for the dev topic 1.

2.4.2.2 | Example of a corresponding picture

The dataset is composed of 200.000 images. The figure below gives an example of one of the lifelog pictures that belongs to the example topic given in section 2.4.2 and to the ground truth given in the figure 2.1:

Figure 2.2: Example of an image from 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 :

- Topic 7

  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 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]

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.

CHAPTER 3

Image/Video Feature Extraction

The task of automatically recognizing and locating objects in images and videos is of extreme importance for computers to be able to understand and interact with their surroundings. Some major applications of this particular task are pedestrian face detection, surveillance, autonomous driving and text digitalization, where object detection is a crucial challenge. [6]

Feature extraction plays an important role in image classification and object detection systems which are two core components of computer vision. It is characterized by two important aspects, the mapping of image pixels into the feature space (explained in more detail in section 3.1.7) and with the extraction of various attributes of an object. Only after extracting useful features from either images or videos, the computer is able to define what an object is or what a certain environment contains.[7]

Putting it simply, feature extraction is the first step in converting an image into text.

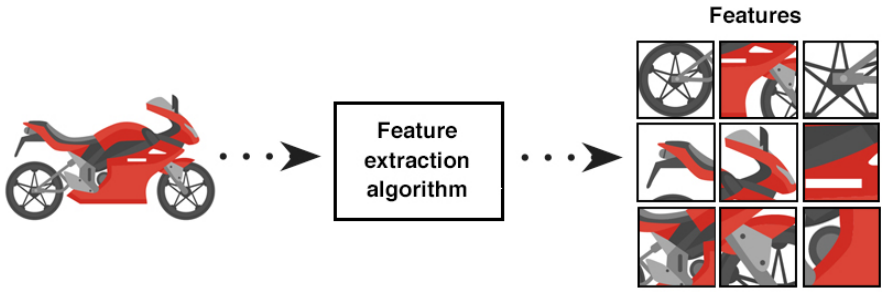


Figure 3.1: Feature extraction from an image. [8]

This chapter starts with fundamental concepts in section 3.1. Section 3.2 gives a brief introduction of the most common computer vision libraries. Neural networks are introduced in section 3.3, where the most common architecture types are presented. All CNN architectures and regression based algorithms used during the development of this thesis are explained with detail in sections 3.4 and 3.5. Classification based algorithms, presented in 3.6, were not used in this work, but they are a required reading in order to understand the differences between them and the regression based algorithms. Finally, a state-of-the-art in object detection and image classification can be read in section 3.7.

3.1 Fundamental Concepts

3.1.1 Artificial Intelligence

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. [9]

3.1.2 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. [10]

3.1.3 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.

Putting it as simple as possible, deep learning is a machine learning technique that teaches computers to learn by example, like a human would. [11]

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 the usage of a large set of labeled data and neural network architectures that contain many 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.

Some use cases for deep learning being used currently in the the real world are: [11]

- Automated Driving : For the detection of pedestrians.
- Aerospace and Defense : To identify objects from satellites and identify safe or unsafe zones for troops.
- Medical Research : For the automatic detection of cancer cells.

However, the main purpose of deep learning, for this work, will be to apply it to the images obtained from lifelogging.

3.1.3.1 How Deep Learning Works

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 learn features directly from the data, without the need for manual feature extraction. This automated feature extraction makes deep learning models highly accurate 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. [11]

3.1.4 Computer Vision

Computer vision is a field of artificial intelligence and computer science that aims at giving computers a visual understanding of the world [12] [13]. It is related with pattern recognition, which is a common way to train a computer so that it can understand visual information. Pattern Recognition is the training a computer goes through when it is fed with different labeled images and then subjected to different algorithms, allowing the computer to hunt for patterns in every element related to those specific labels [14].

3.1.5 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. [15]

Image annotation, one of the most important tasks in computer vision, is the process of manually 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. [16]

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.

3.1.6 | Object Detection, Segmentation and Recognition

Object detection is the name given to the process that combines image classification with object localization [17]. 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 the recognition of an object contained in a bounding box. [6]

3.1.7 | 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. [7]

3.1.8 | Object

An object is used to identify specific items in an image or specific frames in 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. [7]

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. [6]

3.1.9 | 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 [18]. This gap is defined by the lack of coincidence between the information extracted from visual data and the interpretation in a given situation. [6]

As an example, picture 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.

- Computer output: Tree, bottle, person, apple, cup.
- Human output: A family having a picnic in the park.

A computer is only able to output the objects detected but it is incapable of giving them any sort of meaning.

3.1.10 | Datasets With Common Objects

A 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 learning neural networks, and in order for neural networks to be trained, they require training datasets, as previously explained.

A few examples of some available datasets are: MS COCO [19], ImageNet [20] , VisualGenome [21], OpenImages [22] and Pascal-VOC [23].

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 3.7

3.2 | Computer Vision Libraries

3.2.1 | OpenCV

OpenCV is an open source computer vision and machine learning software library originally developed by Intel in the year 2000 [24].

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. [25]

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

3.2.1.1 | 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. [26] [27]

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.

3.2.2 | 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. [28]

3.2.3 | 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.

BoofCV 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 [29].

3.2.4 | GluonCV

GluonCV provides implementations of state-of-the-art (SOTA) deep learning algorithms in computer vision. It aims to help engineers, researchers, and students quickly prototype products, validate new ideas and learn computer vision. [30]

3.3 | 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.

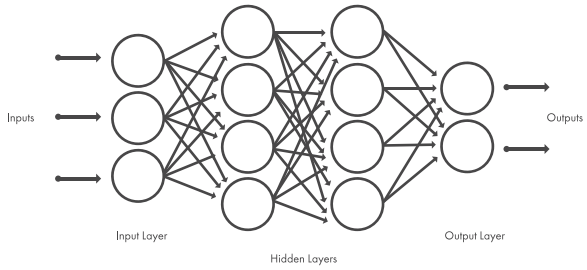


Figure 3.3: Typical neural network architecture. [31]

The connections between layers are called weighted connections and they are adjusted with a real-valued number attached to them. This number is important, because 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. Se figure 3.4 for a clear representation of this process.

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. [32]

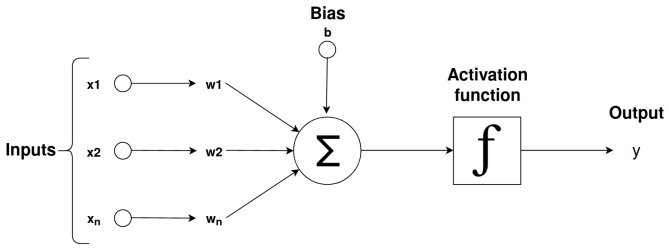


Figure 3.4: Operations done by a neuron.

3.3.1 | 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 it to learn the features and 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 pretrained model. This works by using an existing network like GoogLeNet, and feed it new data of previously unknown classes to the network. After some tweaks to the network, it will be able to categorize only a specific object instead of many different ones. This not only allows the network to be more precise in categorizing that one specific object, but it will also save lot of computation time. [11]

3.3.2 | Types of Neural Networks architectures

3.3.2.1 | 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. [32] [33]

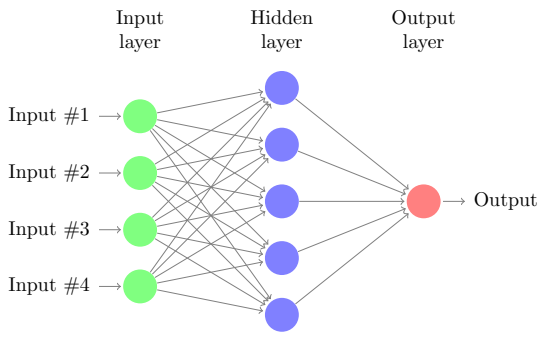


Figure 3.5: Example of a Feedforward Neural Network with one hidden layer (with 5 neurons) [34].

3.3.2.2 | 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 one 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. [32]

3.3.2.3 | 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. [32]

3.3.2.4 | 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. [32]

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. [35]

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. [11]

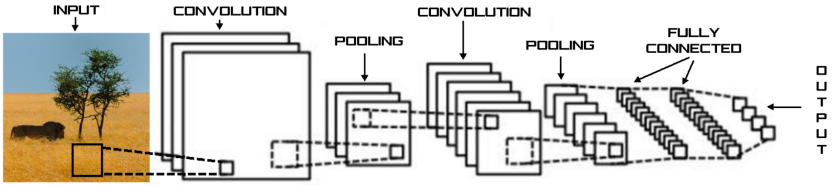


Figure 3.6: CNN architecture
[2]

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 [35].

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. [35]

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]

3.4 | CNNs architectures For Image Classification

3.4.1 | 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 on 3 ideas [36]:

- 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.

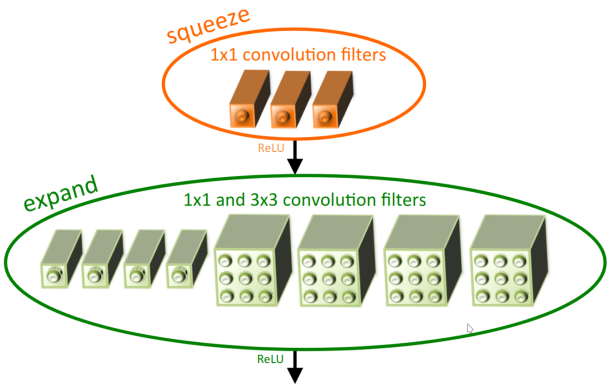


Figure 3.7: SqueezeNet fire module. [36]

This neural network is built with fire modules, which are represented in figure 3.7.

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 3.8). 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.

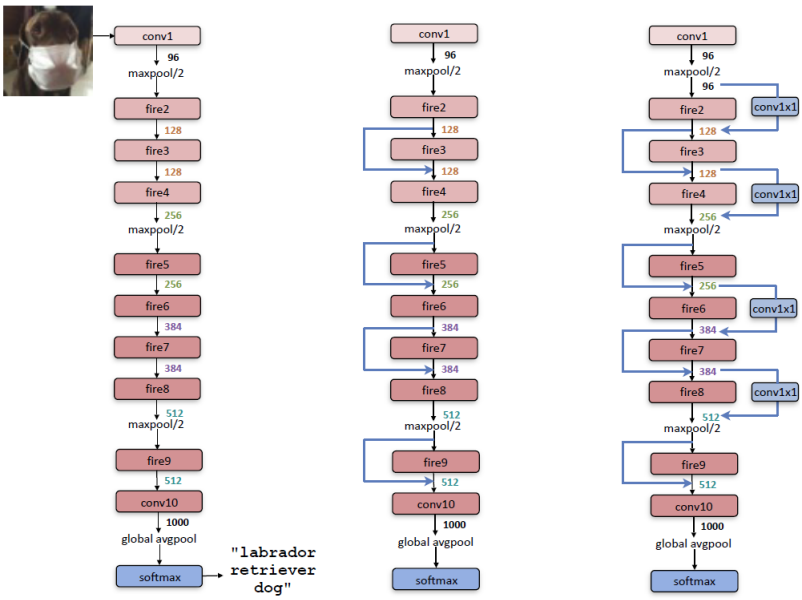


Figure 3.8: SqueezeNet architecture. [37]

3.4.2 | ResNet

The core idea of ResNet (Residual Neural Network) is introducing skip connections (also called identity shortcut connection, represented in figure 3.9). 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. [38]

This architecture is represented in figure 3.10.

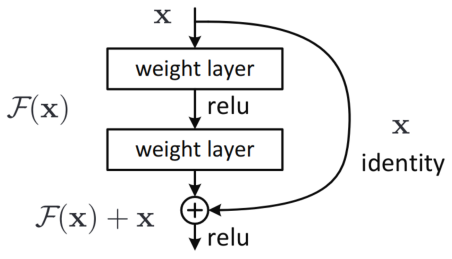


Figure 3.9: Skipping connection example.[38]

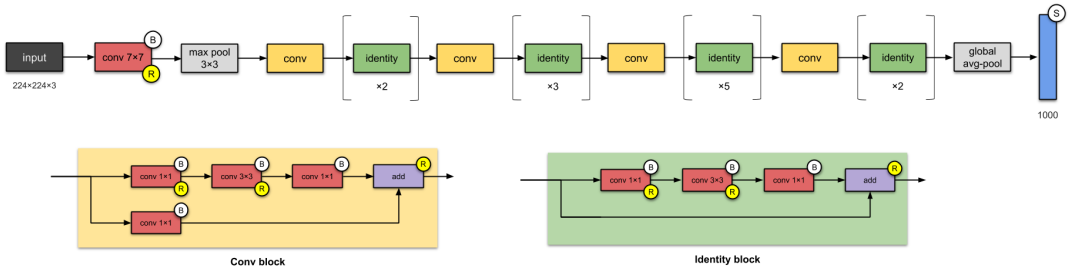


Figure 3.10: ResNet Architecture.[39]

3.4.3 | 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 3.11) that allowed for a more efficient computation and deeper network.

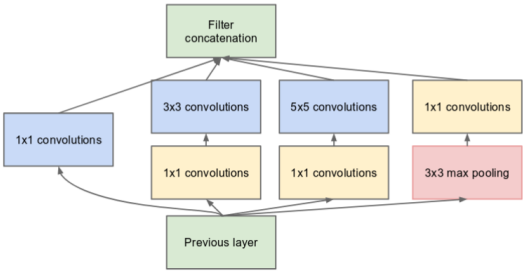


Figure 3.11: Inception Module. [40]

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 3.12 shows an example of this).

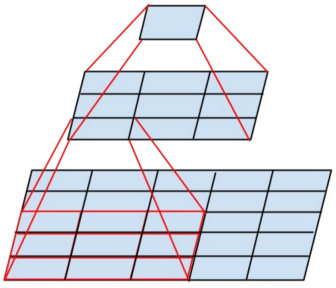


Figure 3.12: Mini-network replacing the 5×5 convolutions (Example of factorization). [41]

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. [42]

The inceptionv3 architecture can be seen in figure below.

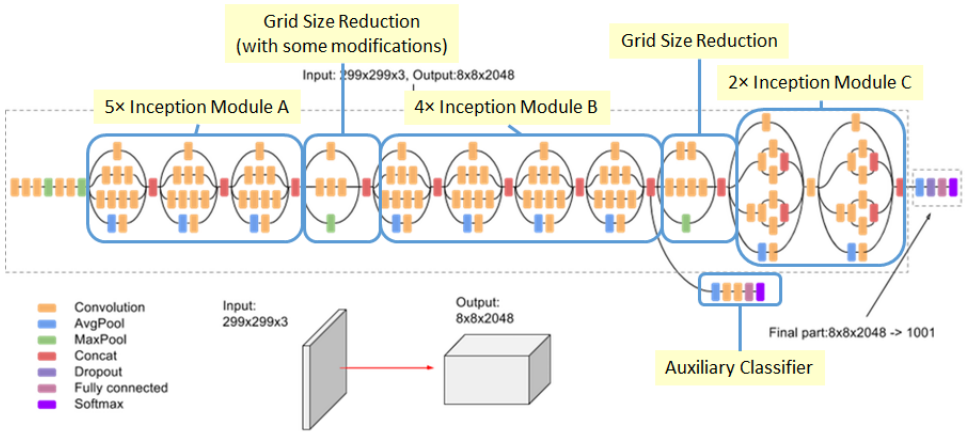


Figure 3.13: InceptionV3 architecture.[42]

Even though InceptionV4 [43] 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.

3.4.4 | 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 (this can be seen in figure 3.14), 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). [41]

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.

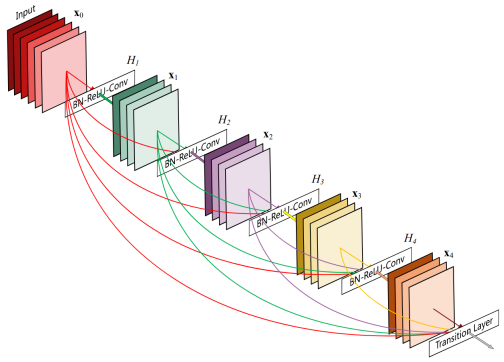


Figure 3.14: A 5-layer dense block. Each layer takes all preceding feature-maps as input. [41]

3.5 | 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. [44] RetinaNet, YOLO and SSD are a few examples of object detection algorithms of this type.

3.5.1 | 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 [45], 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.[44]

In figure 3.15 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.

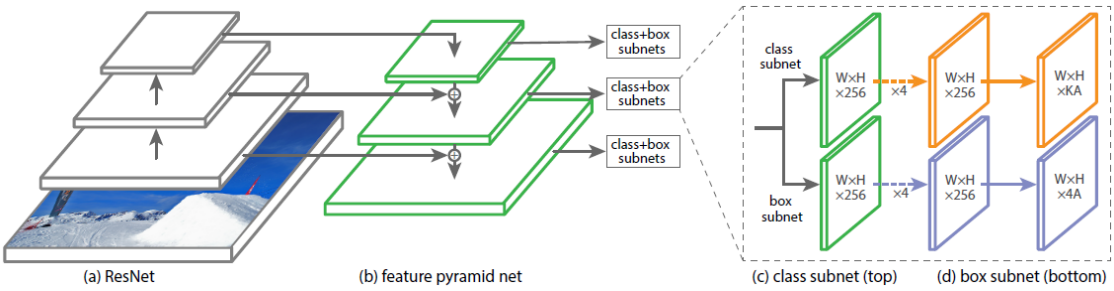


Figure 3.15: RetinaNet architecture.[44]

3.5.2 | 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. [6]

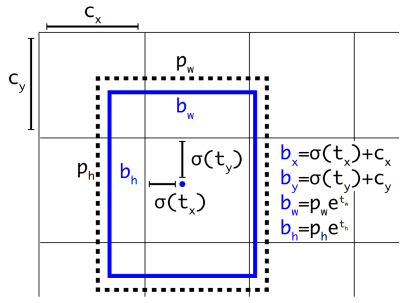


Figure 3.16: Bounding Box Prediction : Predicted Box (Blue), Prior Box (Black Dotted).[46]

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. [46]

The full architecture of YOLOv3 is represented in figure 5.15.

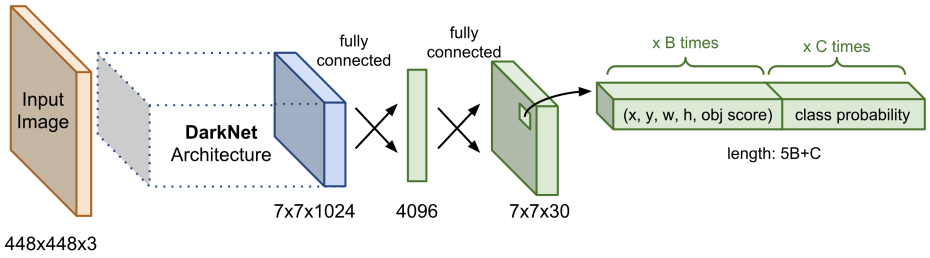


Figure 3.17: The network architecture of YOLO base model.[47]

3.5.3 | 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. [48]

3.5.4 | 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 [49] 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. [50]

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 3.18.

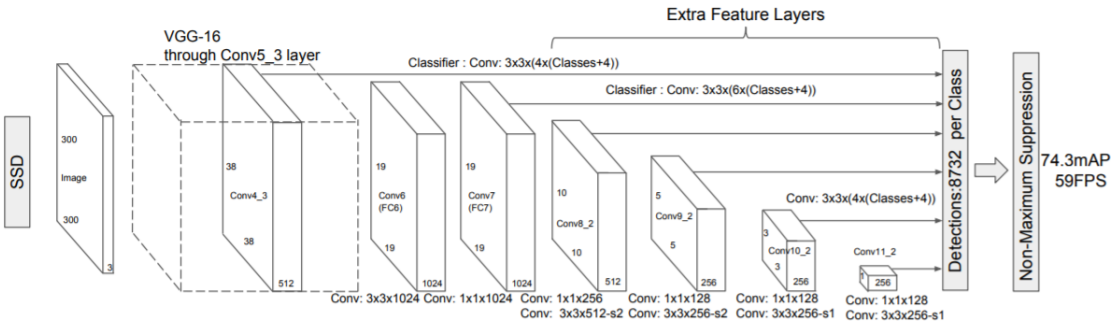


Figure 3.18: SSD architecture. [50]

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. [50]

3.6 | 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. [44]

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

3.6.1 | R-CNN Models Summary

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

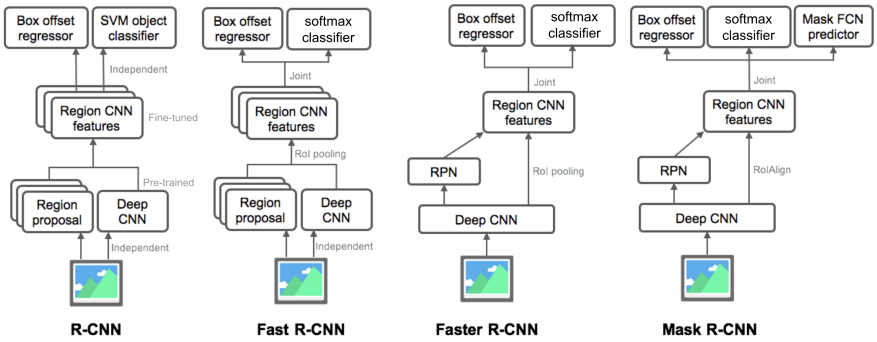


Figure 3.19: R-CNN model family summary. [51]

3.6.2 | 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 [51], which is a common algorithm to provide region proposals that can potentially contain objects [52].

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

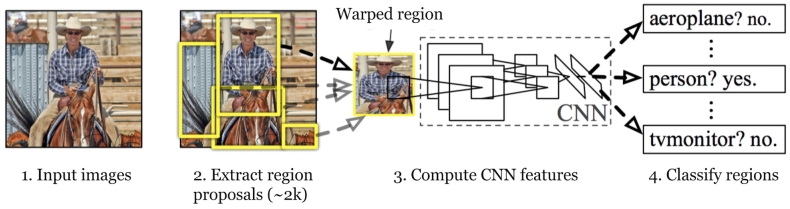


Figure 3.20: R-CNN architecture. [51]

3.6.3 | Fast R-CNN

The idea behind Fast-RCNN [53] 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. [51]

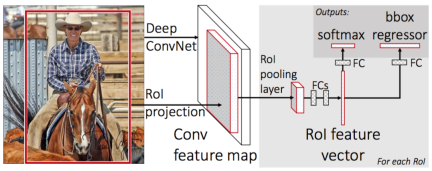


Figure 3.21: Fast R-CNN architecture. [51]

3.6.4 | Faster R-CNN

The Faster R-CNN [54] 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. [51]

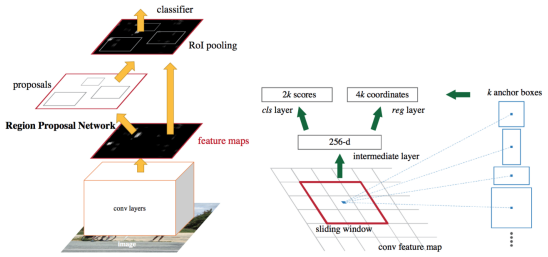


Figure 3.22: Faster R-CNN architecture. [51]

3.6.5 | Mask R-CNN

The final model of the R-CNN family, mask R-CNN [55], 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. [51]

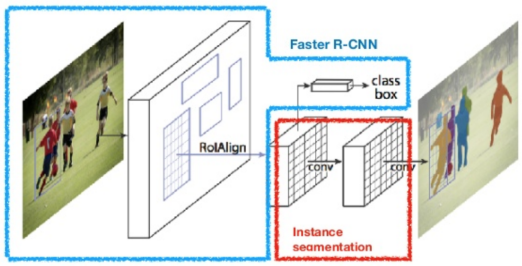


Figure 3.23: Mask R-CNN is a Faster R-CNN model with image segmentation. [51]

3.7 | State-Of-The-Art

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 the figures 3.24 and 3.25 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.

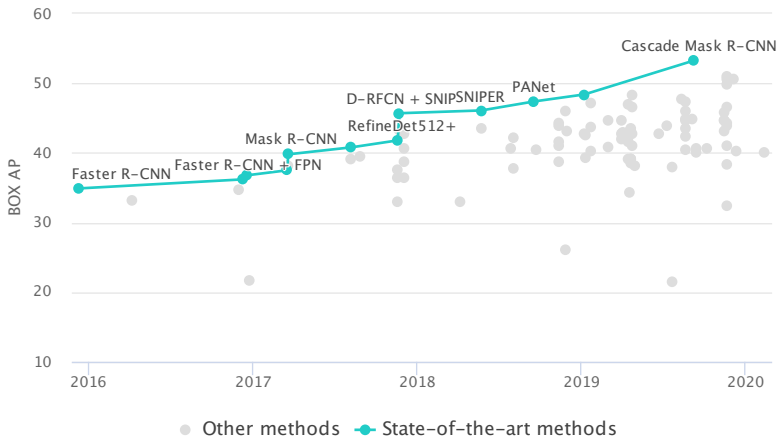


Figure 3.24: Object Detection on COCO test-dev benchmark .[56]

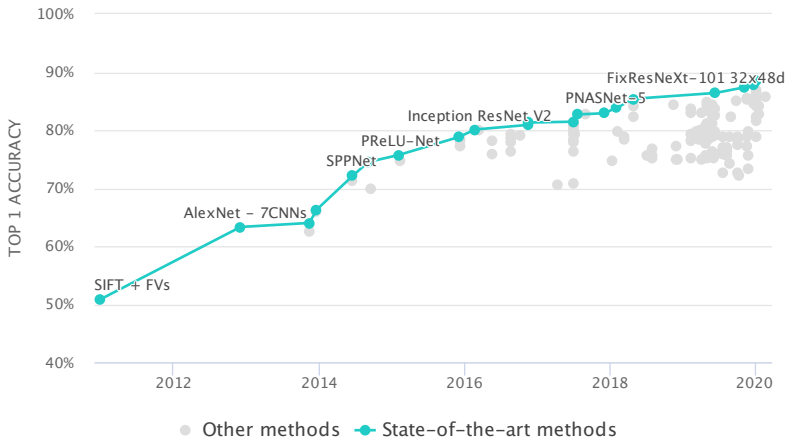


Figure 3.25: Image Classification on ImageNet benchmark. [57]

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 [57] and [56] which is a website dedicated to the current state-of-the-art for object detection and image classification.

3.7.1 | COCO Test-Dev

The COCO benchmark [19] 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) [58]	ResNeXt-152	53.3
Tan et al. (2019) [59]	EfficientNet	51.0
Zhang et al. (2019) [60]	ResNeXt-101	50.7
Girshick et al. (2018) [61]	ResNeXt-152	50.2
Li et al. (2019) [62]	ResNet-101	48.4
Zhang et al. (2019) [60]	ResNet-101	46.3
Mahajan et al. (2018) [63]	ResNeXt	45.2
Zhao et al. (2019) [64]	VGG16	44.2
Cai et al. (2018) [65]	ResNet-101	42.8
Wang et al. (2019) [66]	ResNet-50	39.8
Lin et al. (2017) [44]	ResNet-101	39.1
Shrivastava et al. (2016) [67]	Inception-ResNet-v2	36.8
Kim et al. (2018) [68]	VGG-16	35.2

Liu et al. [58] 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 [38] and ResNetXt [69]. 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 [69] backbones into the Cascade Mask R-CNN baseline.

Figure 3.26 presents the architecture for CBNet.

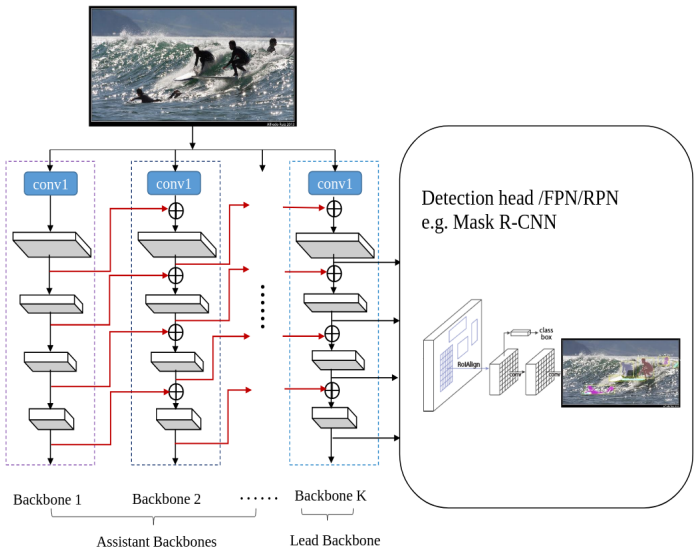


Figure 3.26: CBNet Architecture for object detection.

3.7.1.1 | 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. [69]

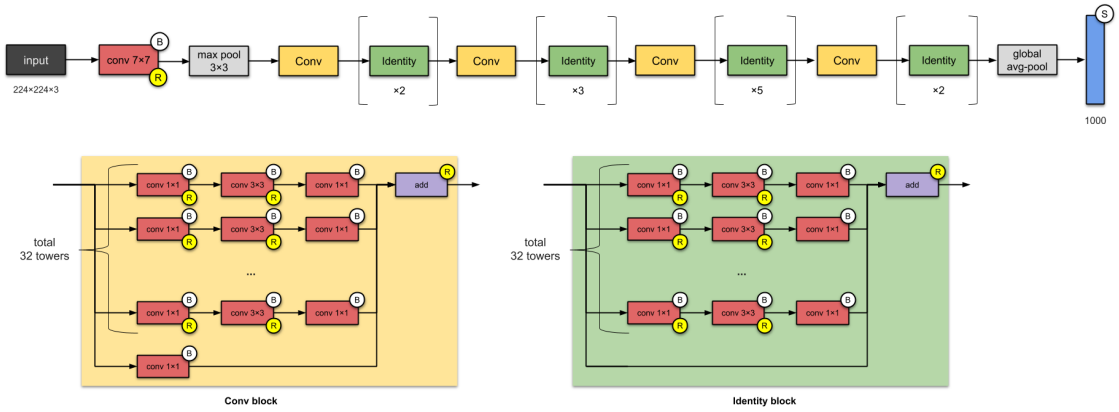


Figure 3.27: ResNeXt architecture. [39]

3.7.2 | ImageNet

The imageNet Large Scale Visual Recognition challenge [70] 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) [71]	EfficientNet	88.4
Kolesnikov et al. (2019) [72]	ResNet-152	87.8
Touvron et al. (2019) [73]	ResNeXt-101	86.4
Xie et al. (2019) [74]	EfficientNet	85.5
Mahajan et al. (2018) [63]	ResNeXt	85.4
Tan et al. (2019) [75]	EfficientNet	84.4
Touvron et al. (2019) [73]	ResNet-50	82.5
Szegedy et al. (2017) [43]	Inception-resnet-v2	80.1
Szegedy et al. (2017) [43]	Inception-v4	80.0
Simonyan et al. (2014) [49]	VGG-16	74.4

Xie et al. [71] 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 3.28 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 (this architecture is explained in more detail in **??**) 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%.

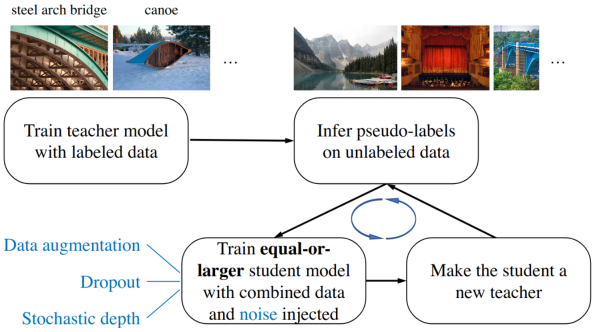


Figure 3.28: Noisy Student Method. [71]

cnnarchitectures

Researchers at Google decided to study the impact of scaling up CNNs, 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 [75].

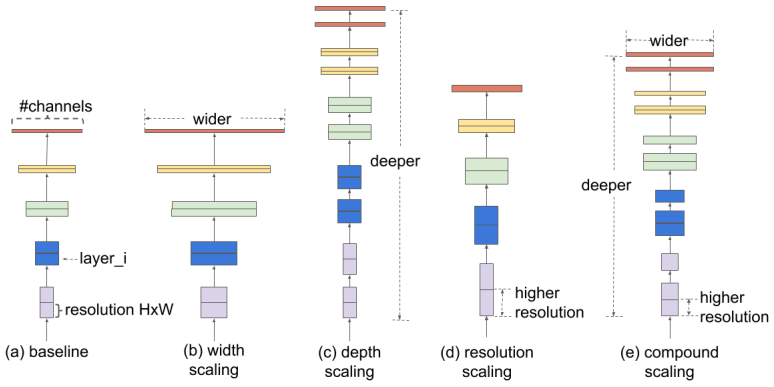


Figure 3.29: Comparison of different scaling methods: (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio. [75]

The baseline network architecture, EffecientNet-B0, uses mobile inverted bottleneck convolution (MBConv), similar to MobileNetV2 [76] and MnasNet [77]. Figure shows the baseline network architecture EfficientNet-B0.

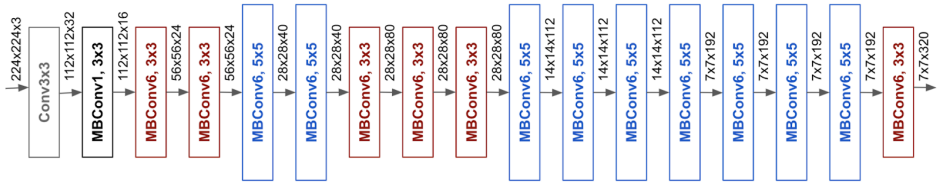


Figure 3.30: EfficientNet-B0 architecture representation.

CHAPTER 4

Information Extraction From Text

As previously discussed in chapter 3 the problem of big data has become more relevant in the recent years. There is too much data to be analysed and the human brain is incapable of processing such large quantities of information. Therefore, information extraction becomes more prevalent as data increases.

Information Extraction (IE) is the task of automatically extracting pre-specified information from textual sources, a trivial example is the usage of IE to analyse large quantities of documents only retrieving the relevant information. In most cases the task of extracting information is concerned with the processing of human language texts by means of natural language processing (NLP).

Figure 4.1: Natural Language Processing [78].

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 discussed in section 4.3 and 4.4 respectively. An overview on some of the available NLP libraries is presented in section 4.5.

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 language understanding to computers. [79]

Human language is extremely complex and rarely precise, to understand it is to understand not only the words, but the concepts and how they are linked together in order to create meaning. This makes NLP one of the most difficult tasks in computer science.

Figure 4.2 shows the classification of NLP, which consists in two major components, Natural Language Understanding (NLU) and Natural Language Generation (NLG) [79].

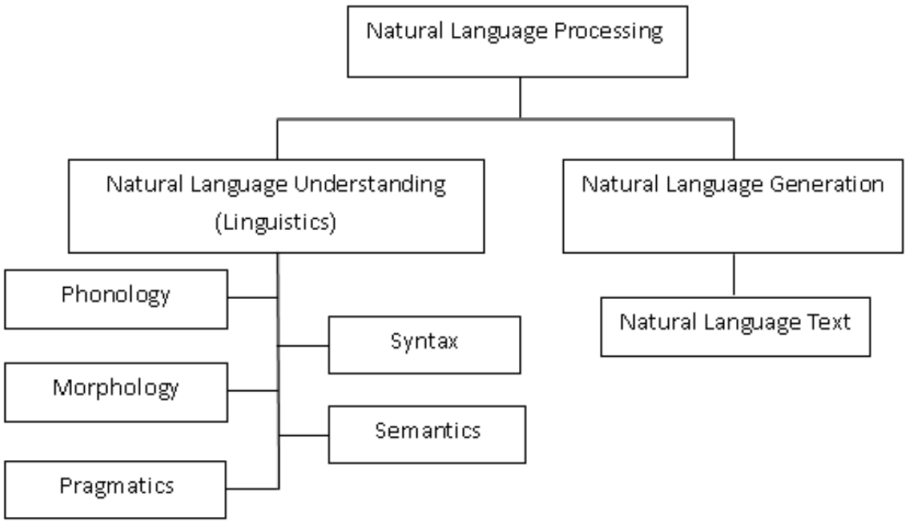


Figure 4.2: Classification of NLP. [79]

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 [79].

4.1.1 | Important NLP Terminologies

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:: Pragmatic is a subfield of linguistics that studies the ways in which the context of a sentence contributes to the meaning [79].

4.1.2 | Core Areas

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. Language modeling underscores quantifying associations among naturally occurring words. Morphological processing deals with the segmentation of meaningful components of words and the identification of the true parts of speech of words used. Parsing consists in the building of sentence diagrams as possible precursors to semantic processing. Semantic processing attempts to distill meaning of words, phrases, and higher level components in text [80].

4.1.3 | Application Areas

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 [80].

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 as an input. [81] 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” has a vector of [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 [82].

Word embedding methods learn a real-valued vector representation for a predefined fixed size vocabulary from a corpus of text [83].

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 [81].

The following image showcases the given example.

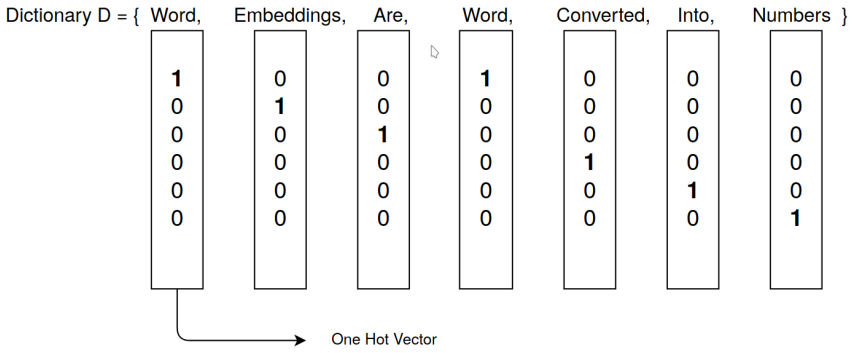


Figure 4.3: Example of text representation by one-hot vector.

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 they generate the same embedding, in different contexts, for the same word, failing to capture the polysemy of the word. This is due to the fact that each word has a single vector, regardless of context. [84]

As an example, having these two phrases:

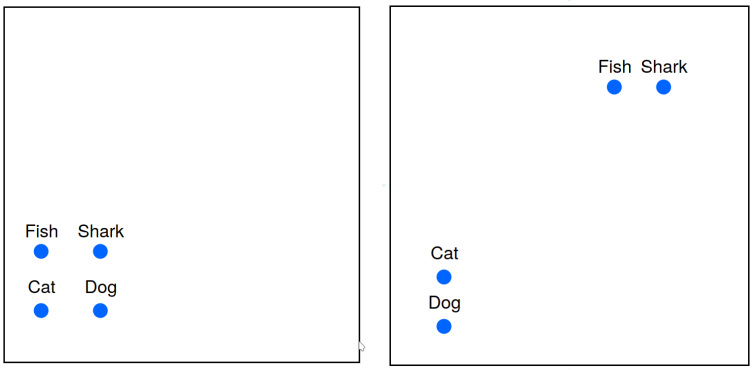
- "The Apple Company is the one who produces iPhones."
- "This apple is delicious."

In this case, the word "Apple" has two different meanings, being one a company and the other a fruit, however for static word embedding models, words only have one single meaning, and therefore the word representation for "Apple" would be the same for both cases. [85]

4.3.1 | Word2Vec

Developed by Tomas Mikolov, et al.[84] 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. The way Word2Vec detects those similarities is by creating vectors that are distributed numerical representations of word features, without human intervention.

In a regular one-hot encoded vector, all words have the same distance between each other, even though their meanings are completely different.



(a) One-hot encoding resulting vector. [86]

(b) Word2Vec encoding resulting vector. [86]

Word2Vec is capable of making accurate guesses, based on past appearances, of a word's meaning.

The output of Word2Vec is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words.

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. [84] [87]

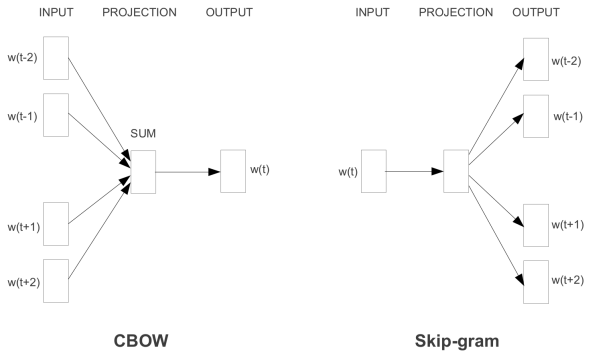


Figure 4.5: CBOW model and Skip-Gram model. [84]

4.3.2 | GloVe

GloVe stands for Global Vectors for Word Representation and was a new approach created by Pennington et all. in 2014 [88] 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 [89].

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 capturing word semantics in different contexts to address the issue of polysemous and the context-dependent nature of words [85]. Using the example given in section 4.3, these models would be able to distinguish the different meaning of the word "apple" 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 LSTM.

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 use context modeling mostly internally and considers the target word embeddings as their main output, the focus of context2vec is the context representation. Context2vec achieves its objective by assigning similar embeddings to sentential contexts and their associated target words [90].

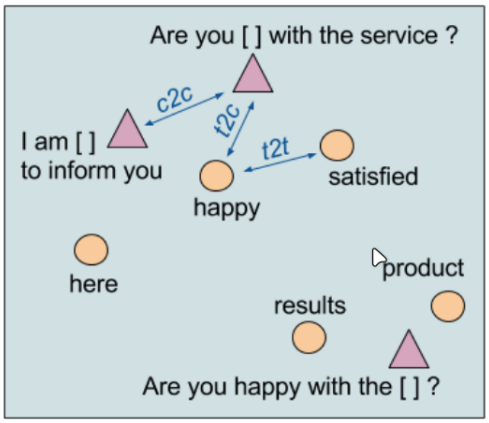


Figure 4.6: A 2D illustration of context2vec's embedded space and similarity metrics. Triangles and circles denote sentential context embeddings and target word embeddings, respectively [90].

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 4.7: Closest target words to various sentential contexts, illustrating context2vec's sensitivity to long range dependencies, and both sides of the target word [90].

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.

This NLP 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. [91]

Below an image showcasing 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 4.8: Nearest neighbors to "play" using GLoVe and context embeddings from a biLM [91].

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 [91].

4.5 | Available NLP libraries

4.5.1 | 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. [92]

This NLP library was chosen for the development of the text processing phase of the practical work, not only because ig provides a well written documentation and being simple to use but also because it offers many useful features such as:

- **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.

4.5.2 | 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 [93].

4.5.3 | Stanford Core NLP

Developed at Stanford University, Core NLP is library written in Java, however 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 [94].

4.5.4 | 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 [95].

4.5.5 | Uncommon Libraries

Other NLP libraries not so common are Flair [96], Polyglot [97], CogCompNLP [98], TextBlob [99].

CHAPTER 5

Image Processing

This chapter aims at discussing how the image processing stage was built.

Initially in section 5.1 a performance comparison between some state-of-the-art image recognition algorithms and object detection models was done. Section 5.2 describes a very raw first attempt at an image retrieval system using only object detection. Subsequently in section 5.3 it is demonstrated how the scene recognition algorithm worked. Finally section 5.5 clarifies how all of the images in the imageclef dataset were processed.

5.1 | Test Runs

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 [100].

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 the objected detected is. 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 5.1.1 and 5.1.2 provide examples of the performance test runs done to the models and algorithms.

5.1.1 | Image Recognition test runs

The imageAI library allows the usage of 4 image recognition algorithms for image recognition which are DenseNet, inceptionV3, ResNet50, and SquezeeNet.

The next page shows three examples of some test runs done to these algorithms. The examples are exhibited in the following way : on the left a picture to be analysed is present and on the right the respective graph with the results is displayed. 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.

The obtained results are discussed in section 5.1.1.1

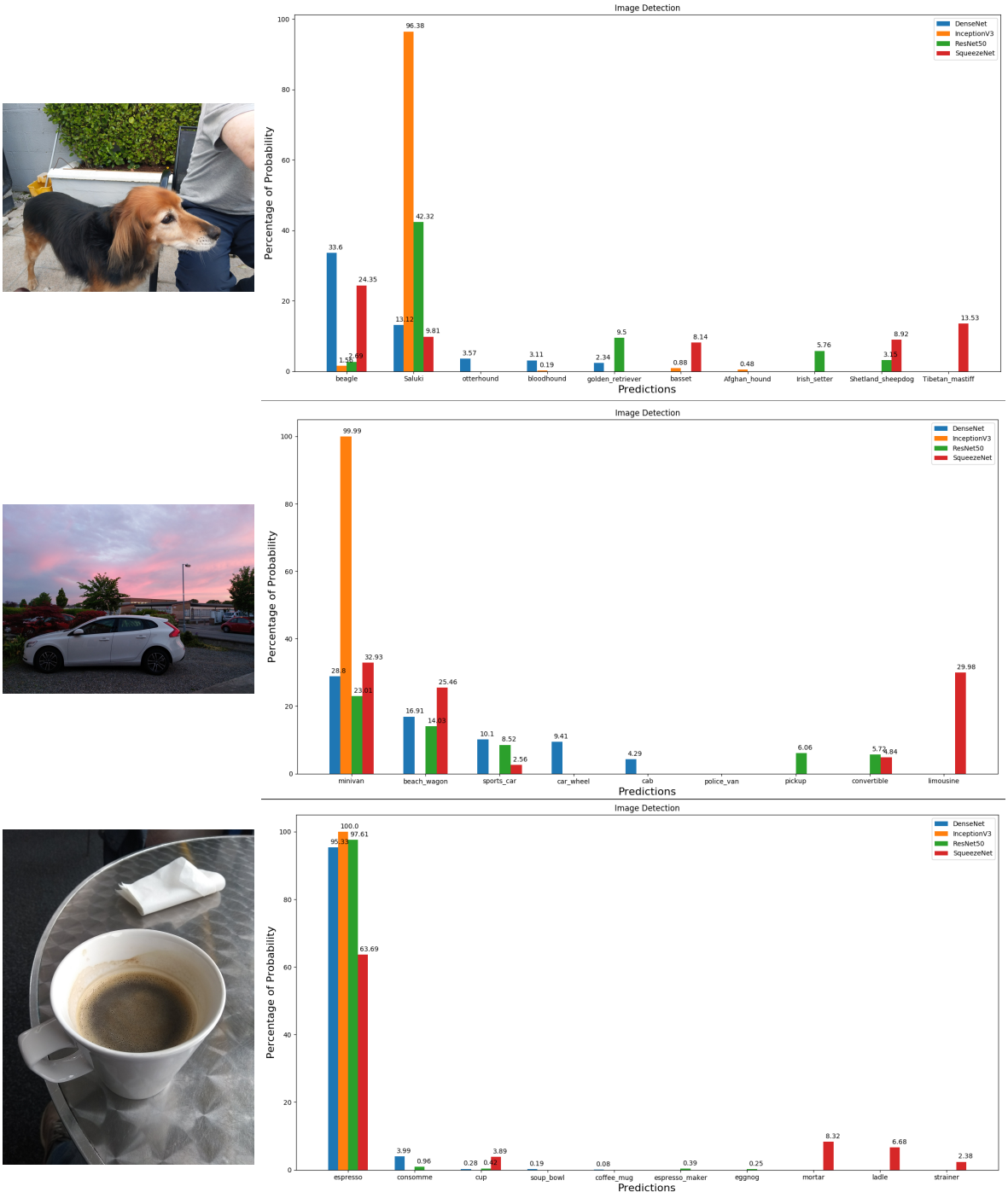


Figure 5.1: Image recognition test runs.

5.1.1.1 | Image Recognition Test Runs Results Analysis

In the first example a picture of a dog (breed Saluki) is analysed. 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. 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.

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

5.1.2 | Object Detection test runs

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. The models that are provided include RetinaNet, YOLOv3 and TinyYOLOv3. [100].

The obtained results are discussed in section 5.1.3.2

The objects that these models are able to detect can be seen in the following image :

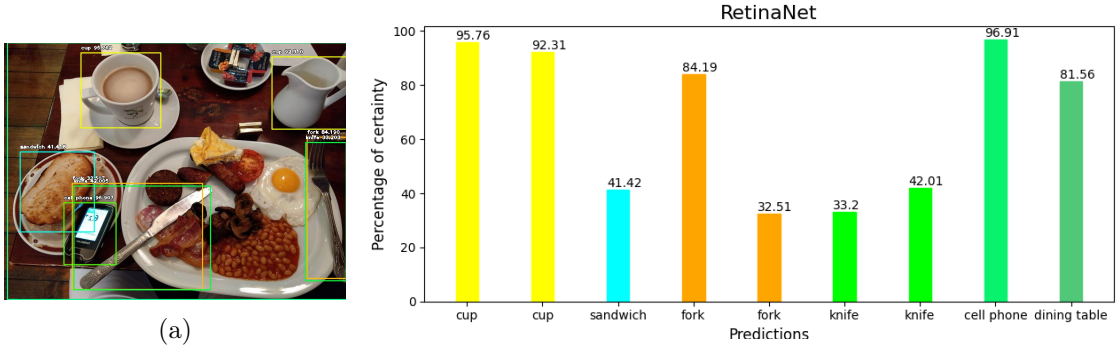
```
person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop_sign,
parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra,
giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard,
sports ball, kite, baseball bat, 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 5.2: Available labels for detection.

The following pictures were used to test the described models.

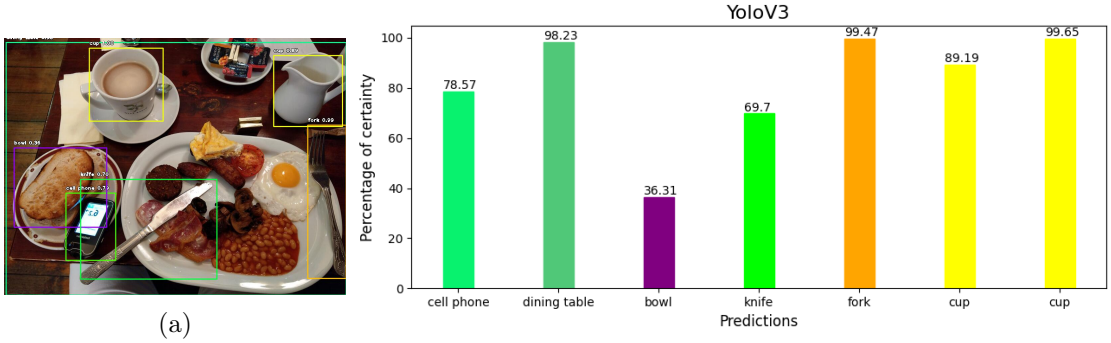
Figure 5.3: Pictures used for test runs.

5.1.2.1 | Object Detection Test Run Number 1



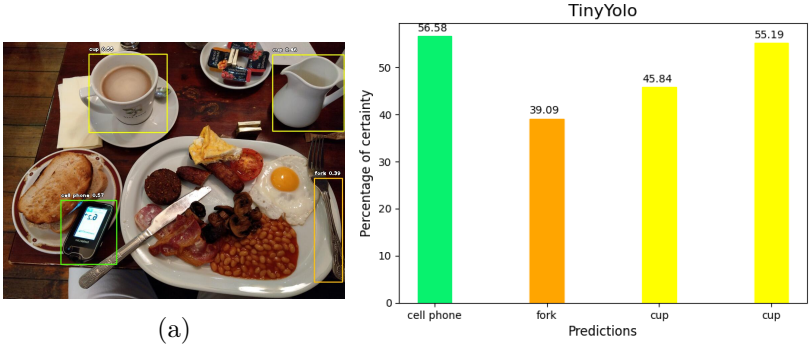
(a) (b)

Figure 5.4: Test run 1 with RetinaNet; a) Analysed picture with detections; b) Achieved performance on detections.



(a) (b)

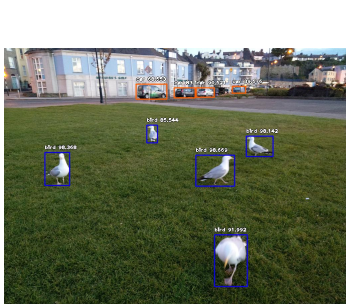
Figure 5.5: Test run 1 with YOLOv3; a) Analysed picture with detections; b) Achieved performance on detections.



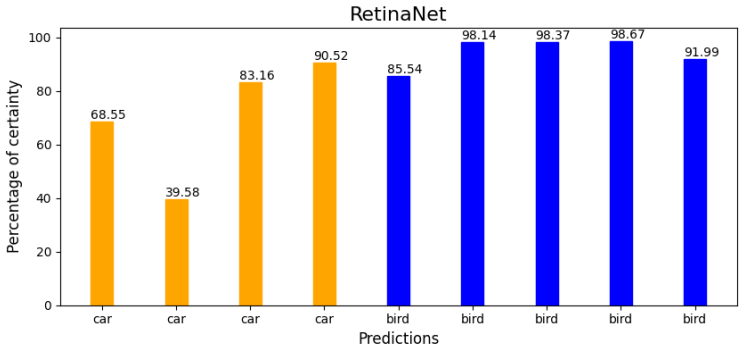
(a) (b)

Figure 5.6: Test run 1 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections.

5.1.2.2 | Object Detection Test Run Number 2

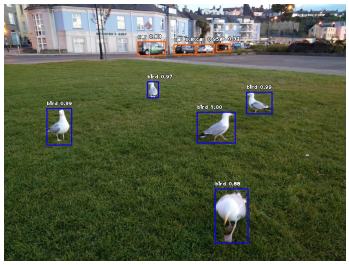


(a)

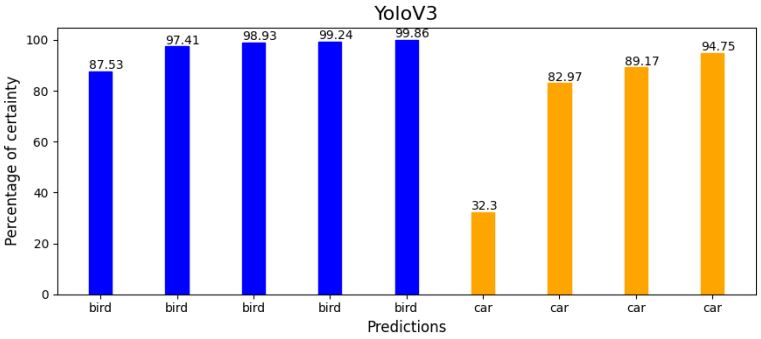


(b)

Figure 5.7: Test run 2 with RetinaNet; a) Analysed picture with detections; b) Achieved performance detections.

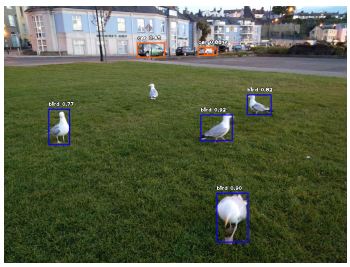


(a)

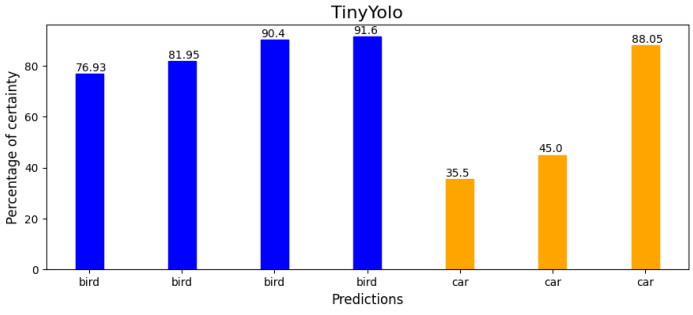


(b)

Figure 5.8: Test run 2 with YoloV3 model; a) Analysed picture with detections; b) Achieved performance on detections.



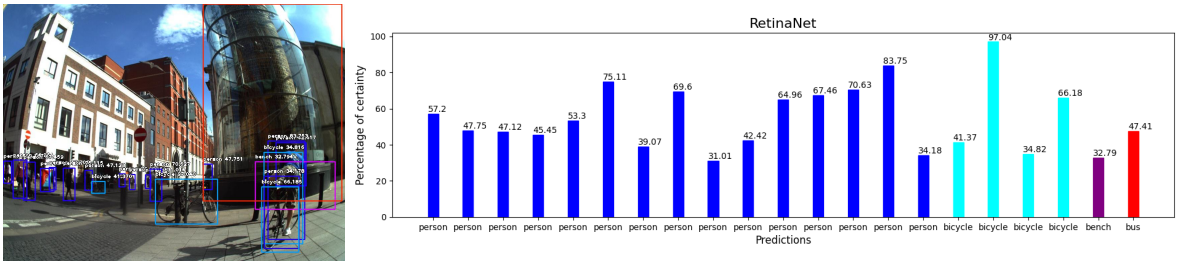
(a)



(b)

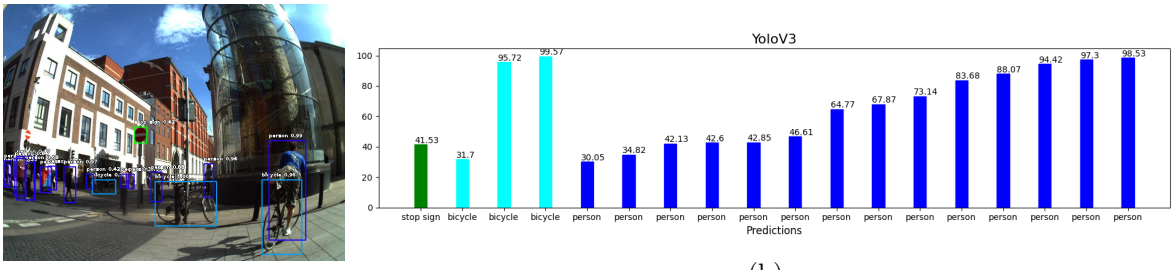
Figure 5.9: Test run 2 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections.

5.1.2.3 | Object Detection Test Run Number 3



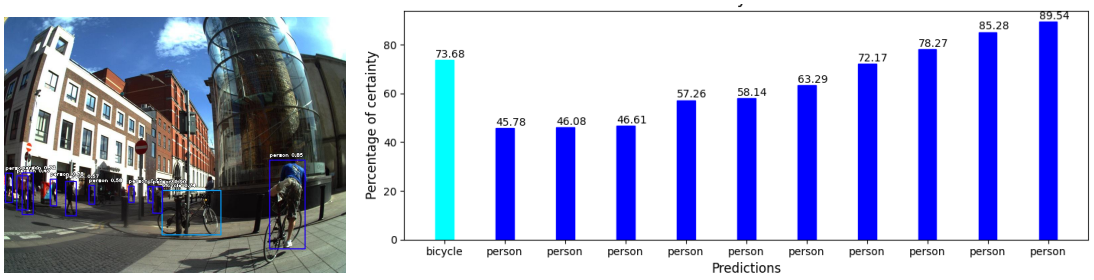
(a) (b)

Figure 5.10: Test run 3 with RetinaNet; a) Analysed picture with detections; b) Achieved performance detections.



(a) (b)

Figure 5.11: Test run 3 with YoloV3 model; a) Analysed picture with detections; b) Achieved performance on detections.



(a) (b)

Figure 5.12: Test run 3 with TinyYolo; a) Analysed picture with detections; b) Achieved performance on detections.

5.1.3 | Object Detection Word Clouds Generation Test Run

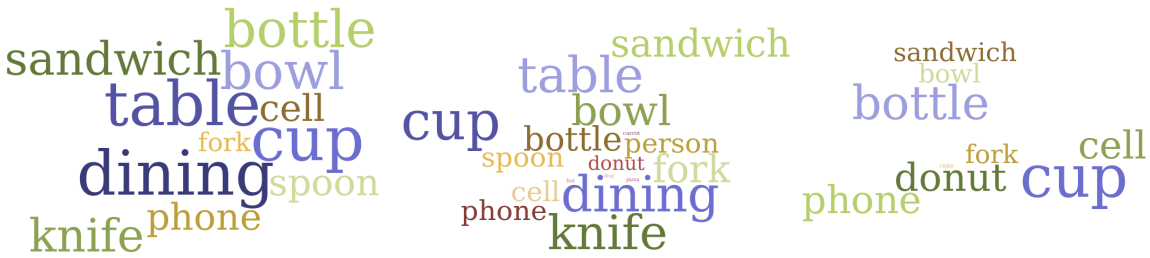
In order to make the labels extraction more easily visible and still achieve some degree of performance comparison between the 3 object detection models word clouds were generated.

For this test 6 previously chosen images with identical setting were processed in order to generate 1 word cloud with all the extracted labels.

In a word cloud, the bigger a word is the more times that label was detected in the pictures.

Figure 5.13: Used images for word cloud generation.

5.1.3.1 | Word Clouds



(a) (b) (c)

Figure 5.14: Generated Word Clouds; a) Yolo word cloud; b) RetinaNet word cloud; c) TinyYolo word cloud

5.1.3.2 | Object Detection Results Analysis

From the different test runs it is possible to analyse that the TinyYolo model under performs severely compared to RetinaNet and YoloV3. This is expected, as explained in section 3.5.3 the TinyYolo model is a smaller model of YOLOv3 that requires less computational resources and that 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, in third example RetinaNet detects a bus in the place of a building while Yolo is capable of detecting a correctly stop sign that no other model detected.

As for the word clouds, it is possible to notice that in the YoloV3 cloud and the retinaNet cloud there are many more words than the tinyyolo cloud, again, tinyyolo is severely under performing when compared to the other 2 model.

Looking at the Yolo 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 from different sizes, this can occur because RetinaNet wrongly detects 1 or 2 object like "pizza", "donut and "person" in one of the images.

This test runs allow for the conclusion that the YoloV3 model is the better performing one, and therefore was the one chosen for the object detections for the automatic retrieval system. However, since the interactive system was built with detections from ResNeXt-101, those detections were also reutilized for the automatic system in a different run for the imageclef challenge. This will be further discussed in section 5.4.

5.2 | Example of a Raw Retrieval System

As a first step in building a fully automatic retrieval system an "alpha 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 returned.

```
------------------------------------------------------------ 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 5.15: System capable of detecting specific user inputted labels in multiple images.

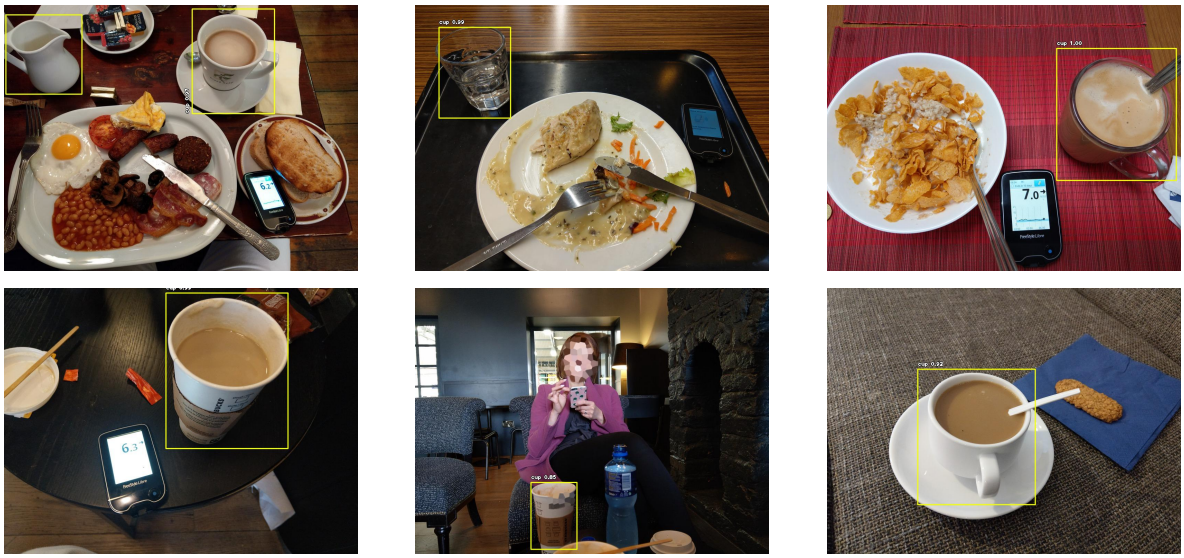
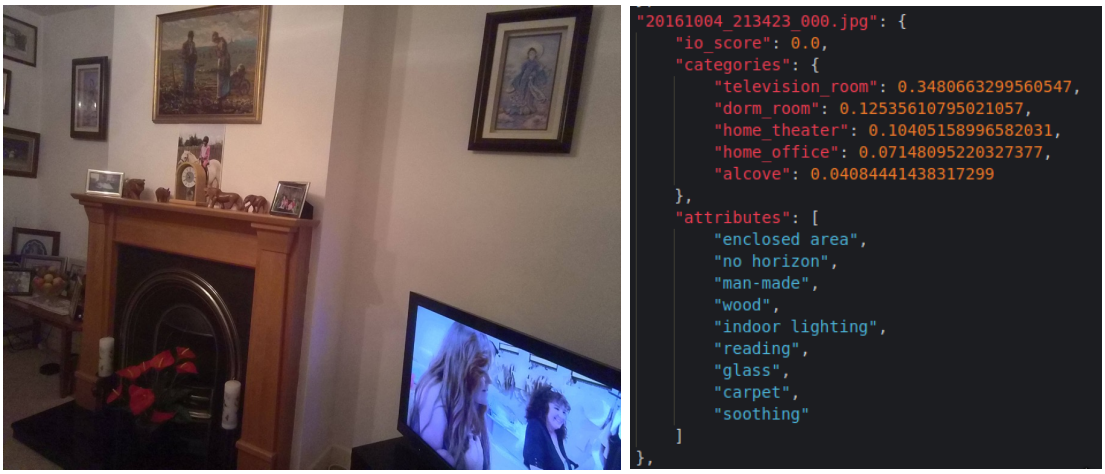


Figure 5.16: Retrieved images for the label "cup" with "40%" threshold.

5.3 | Scene Recognition

In order to detect interiors, exteriors and places a pretrained model provided by Zhou et al. [101] trained on the Places365 standard dataset was used .

The following example shows the extractions done to a random image from the imageclef dataset.




```
"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 5.17: Example of a 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 interior vs exterior certainty. It ranges from 0 to 1. If its close to 0 it means the image is probably an interior and if it is close to 1 it means it is probably an exterior. In this example, the image is an interior and the "io_score" is 0, therefore the model predicted correctly that the image is in fact an interior.

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, the "attributes" is the section where the model tries to describe the picture. Some of the predicaments 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.

5.4 | Run 1 and Run 2

In this year challenge, 2 different runs were made with the automatic retrieval system.

For the first run the extract objects from the images is 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 that was proposed by Mahajan et al. [63]

In the second run, the object detection algorithm used is the YoloV3 [46] model pretrained in the COCO dataset.

This was done in order to understand if different object detection algorithms, would help in achieving better results.

5.5 | Processing the Imageclef Dataset

In order to extract the maximum possible labels, all images in the imageclef dataset were fully processed with YOLOv3 or with ResNeXt-101 and the Places scene recognition model. This is an exhaustive approach since it takes a lot of computer processing time and resources.

Using 5.17 a) as an example, the fully processed image looks like this:

Figure 5.18: Fully processed image with YOLOv3 and PLACES365.

Figure 5.19: 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. Unfortunately that data is not accurate enough, and a good option would be to use activity recognition algorithms to extract activities from images. unfortunately 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 location of the given "concept" in the image according to the pixels of the image.

CHAPTER 6

Text Processing and Image Retrieval

This chapter aims at making clear how the automatic retrieval system works. Section 6.1 clarifies how the system does text processing, word extraction and how it manages to categorize the extracted data. Subsequently in section 6.2 an explanation is given on how the system retrieves images for a given moment and how it is able to compare the similarity between the mined text words and the visual concepts. Lastly in section 6.3 a diagram that overviews the system workflow architecture is exhibited in order to clarify the process.

6.1 | Word Extraction

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 library [92] is used to analyse the query topics fully, extracting relevant words from the title, the description and narrative. 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".

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.

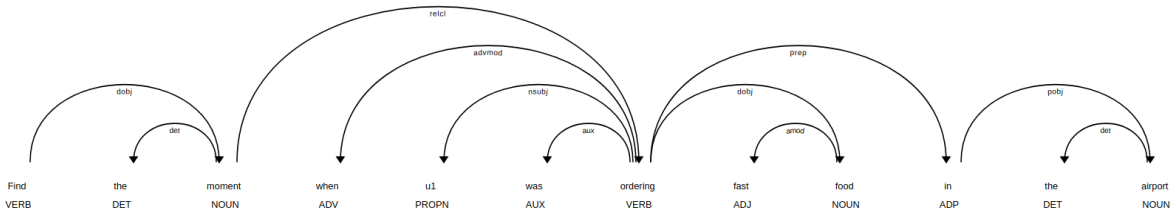


Figure 6.1: Linguistic annotations generated by the SpaCy library [92] for the narrative of the topic 6 of the test topics.

As an example, some of the rules that were applied to the extraction and categorization of the textual data were:

- 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 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 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.

Using figure 6.1 to illustrate, the extracted words for the the narrative of the test topic 6 were:

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

6.2 | Retrieval Stage

In the retrieval step the images are recovered according to the desired query topic. As an example figure 6.3 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 6.2: 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 words "evening time" are wrongly placed on the locations category. One of the reasons for this to happen is because of the previously described rule, where if a word is an "ADP" and it is "in" then the following "NOUN" is probably a location, which in this case is not correct.

6.2.1 | Retrieving Images According to the Similarity Between Words

In order to retrieve images according to the defined moment in a textual topic, a comparison is made between the extracted visual concept words 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.

The SpaCy en_core_web_md model allows the computer to calculate the similarity between the visual concepts and the extracted data. This model is an english multi-task CNN

trained on OntoNotes, with GloVe vectors trained on Common Crawl [92] . As an example of similarity between words using the described model, the word "television" and "seafood" have a similarity of 0.0705759162558067 while "television" and "screen" have a similarity of 0.4271196001925812.

A confidence score is calculated based on that value for each image and for each topic. This score ranges from a value of 0.0 (0%) to 1.0 (100%).

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.

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 has a lower confidence score. Not only is the confidence score influenced by the similarity between words, the prediction score, but also the computed weights assigned to each category. 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 weight factor and the distributed weight factor.

The distributed weight factor value is not the same for each category, the ratio of the distribution is equal to the default ratio of the importance factor of all categories. To make it clearly, if the importance weight factor for ”relevant things” is 0.5, which is half of the sum of all importance factors, and if the ”activities” category is worth 0.2 and has no extracted textual data, then half of 0.2 is distributed to ”relevant things”, which increases the importance to 0.6 and the remainder 0.1 will be distributed the same way to other categories ensuring that the sum of all importance factors and distributed weight factors is 1. The negative categories works the same way, but instead of contributing for the confidence score, it decreases the value.

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 are stored and the 10 highest pictures are the ones who count for the f1@measure score.

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.

In order to have a better visualization of how this calculations are done section 6.2.5 shows all of the equations needed to calculate the confidence score and section 6.3 shows a diagram that illustrates all the steps that the system does in order to retrieve images.

6.2.2 | Calculation of Similarity Scores

Using the figures 5.18 and 5.19 to illustrate the different categories, the calculation of the similarity scores are done trough the comparision of the extracted words and the visual concepts:

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

6.2.3 | Run 1

The first run to be 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 important 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 0.5.

A general threshold was previously defined in order to remove images of low concept scores, images above the threshold are selected for the calculation of the confidence score in order to select them for the query topic. The threshold was implemented through some trial and error during the test phases, and it merely serves the purpose of saving some computational time.

6.2.4 | 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 "ioscore". Categories like ”activities” and ”locations” get a lesser importance weight factor since they are being compared to the organizers label data which is limiting and lesser accurate. Another difference from Run 1 to Run 2 is that all of the negative categories were discarded from contributing to the confidence score, in order to save processing time and since the first results from Run 1 did not appear to impact much.

6.2.5 | Confidence Score Computation Equations

The following equations give an understanding of all the computations done in order to calculate the confidence score:

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

6.3 | System Workflow Architecture Diagram

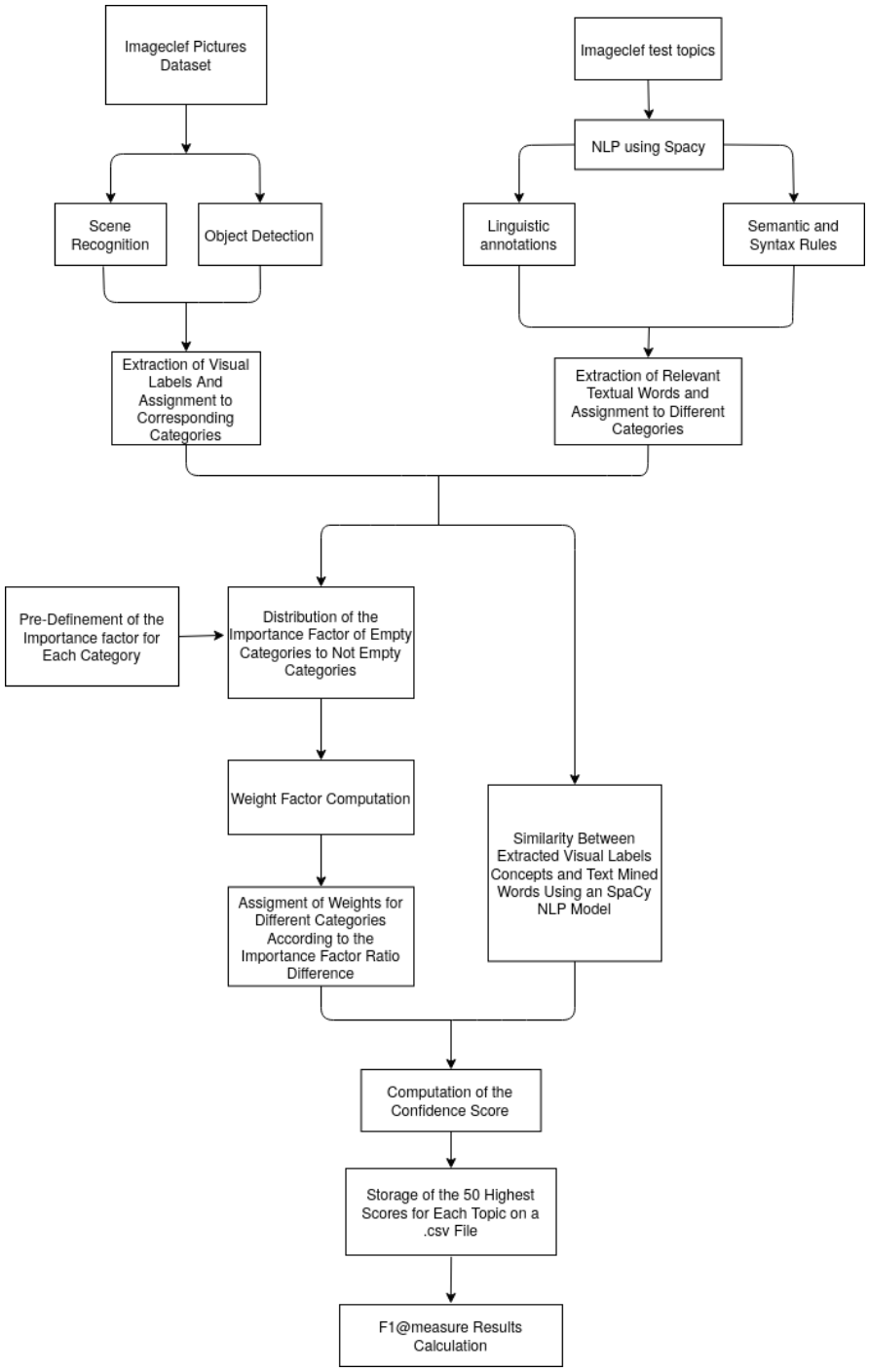


Figure 6.3: System Architecture

CHAPTER 7

Results

This chapter aims at presenting and discussing the achieved results on the imageclef challenge. Firstly, in section 7.1 a few examples of some tests that showcase the fine-tuning of the system is presented. Secondly, in section 7.2 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 7.3 the achieved results on the challenge are presented.

7.1 | 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).

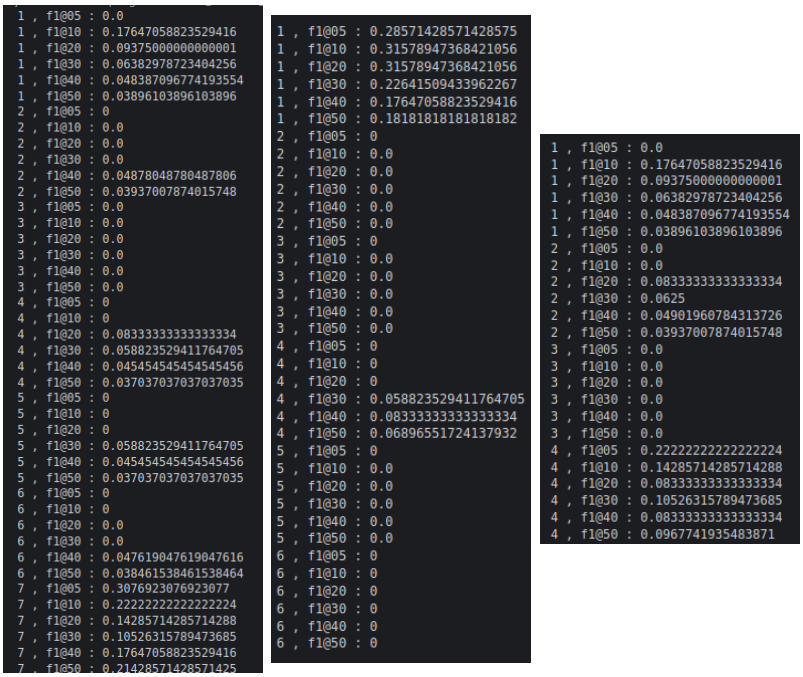


Figure 7.1: Examples of some different results with the fine-tuning of the weight distribution.

This examples show the performance achieved of the system on a F1-measure@XX measure. The differences in performance on the given examples happen because of the difference in calculations of weights for each category.

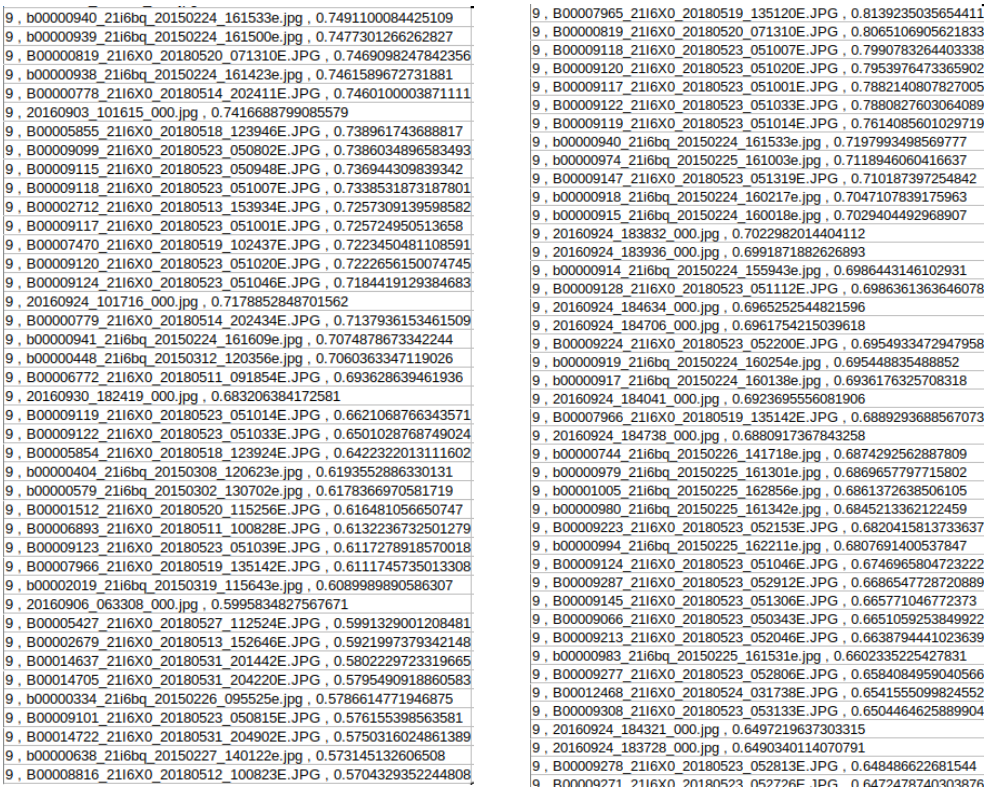
7.2 | System Performance Example

This section provides an insight on how the system performed in the imageclef challenge using topic 9 as an example :

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".

7.2.1 | Run 1 and Run 2 Image Retrieval Example

The following images showcase an excerpt of the csv files generated by both runs that were sent to the imageClef evaluation platform.
The file is organized in the following way: [topic id number, image name, confidence score].



(a) Run 1 (b) Run 2

Figure 7.2: Achieved results on topic 9 of the test topics.

7.2.2 | Top 3 Retrieved Image on Run 1

As it can be observed in figure 7.2 a), the top 3 images that were retrieved for topic 9 in run 1 were:

1. b00000940_21i6bq_20150224_161533e.jpg;
2. b00000939_21i6bq_20150224_161500e.jpg;
3. B00000819_21I6X0_20180520_071310E.JPG.

The images can be seen below.

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

Figure 7.3: Top 3 retrieved pictures for topic 9 on run 1

7.2.3 | Top 3 Retrieved Images on Run 2

Using again figure 7.2, the top 3 images that were retrieved for topic 9 on run 2 were:

1. B00007965_21I6X0_20180519_135120E.JPG;
2. B00000819_21I6X0_20180520_071310E.JPG;
3. B00009118_21I6X0_20180523_051007E.JPG.

The images can be seen below.

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

Figure 7.4: Top 3 retrieved pictures for topic 9 on run 2

7.2.4 | Topic 9 Performance Analysis

Even if the top 10 pictures are the ones that count for the performance evaluation of the run, the top 3 pictures are already representative the system efficiency. It is clear that for topic 9 the run 1 achieved better performance, since the first 2 pictures clearly belong to the moment described in topic 9. The third picture however does not belong to topic 9 since it is a picture of a person eating a lasagne alone but it is a similar scenario.

Run 2 however did not retrieved any single picture from topic 9 on the top 3 pictures, however the scenario is identical. All of the 3 photos are representative of food being eaten. However, something to note is that run 2 confidence scores are higher than run 1, but run 1 achieved better practical results. Some possible reasons for this to occur may be :

- 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 providing better detections than the algorithm used for run 2.
- The category weight distribution on run 2 decreasing 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.

7.3 | Achieved Overall Performance Results

Table 7.1 and table 7.2 provides an overview of the achieved performance results in different runs for different systems that were obtained in the year 2019 and 2020 for the ImageCLE-Flifelog 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 7.1: Results obtained in 2019 from UA.PT Bioinformatics [3] and the best team [102].

2020 Results			
Team	System Type	Run Name	F1-measure@10
UA.PT Bioinformatics	automatic	Run 1	0.03
UA.PT Bioinformatics	automatic	Run 2	0.03
UA.PT Bioinformatics	interactive	Run 3	0.54
Best Results Achieved by a Team			
HCMUS	interactive	Run 10	0.81

Table 7.2: Results obtained in 2020 from UA.PT Bioinformatics [4] and the best team [5].

7.3.1 | Overall Performance Analysis

Comparing the table 7.1 that shows the results of the year 2019 and table 7.2 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 having a fully automatic system.

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 chose the picture that he thinks are 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.

CHAPTER 8

Conclusions

This chapter presents the general remarks about the work done in this dissertation while introducing some possible improvements for future work.

8.1 | Discussion

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

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 7 section 7.2.2 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.

8.1.1 | 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.

8.1.2 | System Disadvantages

- Requires a lot of processing time in order to retrieve images of a big dataset.

- Requires good computer specs.
- Low-score end results.

8.2 | 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 .

Another future improvement would be to use the google cloud vision api [103], which gives labels that are more related to the words extracted.

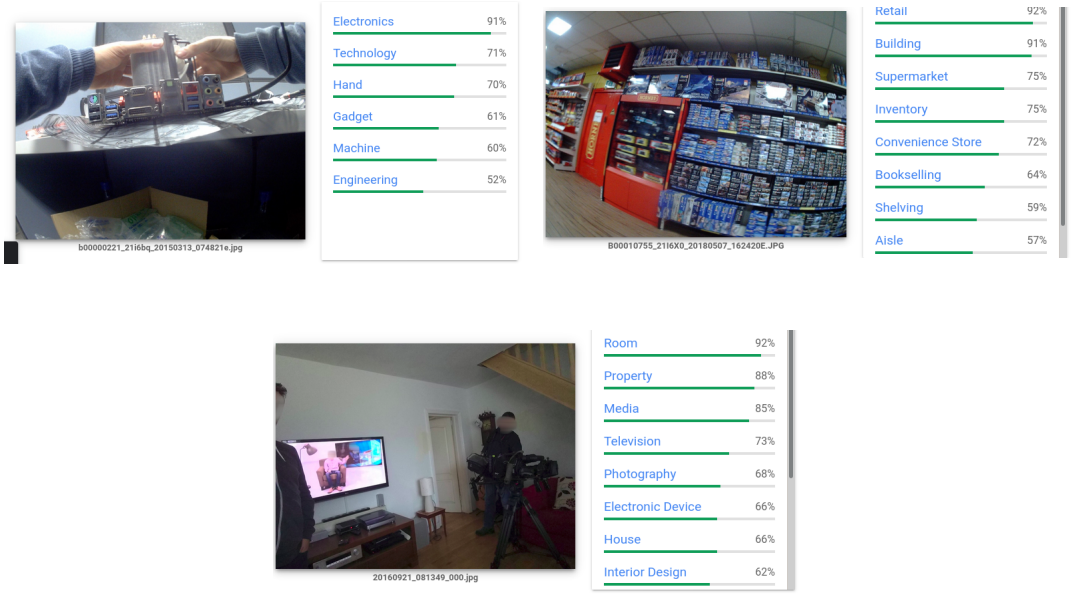


Figure 8.2: Examples of google cloud vision API extracted labels [103]

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, 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.

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

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

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

[7] 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.

[8] *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).

[9] 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).

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

[11] 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).

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

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

[14] Ilija Mihajlovic. *Everything You Ever Wanted To Know About Computer Vision.* URL: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e (visited on 01/22/2020).

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

[16] 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).

[17] 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).

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

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

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

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

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

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

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

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

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

[34] 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).

[35] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: November (2015). arXiv: 1511.08458.

[36] 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.

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

[38] 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.

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

[40] 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).

[41] 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.

[42] 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).

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

[44] 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.

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

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

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

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

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

[50] 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.

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

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

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

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

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

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

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

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

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

[60] 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.

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

[68] 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.

[69] 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.

[70] 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.

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

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

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

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

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

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

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

[78] Rite Wiki. *Natural Language Processing - Ryte Wiki - The Digital Marketing Wiki*. URL: https://en.ryte.com/wiki/Natural_Language_Processing (visited on 08/01/2020).

[79] 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).

[80] 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.

[81] 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).

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

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

[84] 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.

[85] David Batista. *[oorsig] Language Models and Contextualised Word Embeddings*. 2018. URL: http://www.davidsbatista.net/blog/2018/12/06/Word%7B%5C_%7DEmbeddings/%20http://www.davidsbatista.net//blog/2018/12/06/Word%7B%5C_%7DEmbeddings/ (visited on 02/10/2020).

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

[87] 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).

[88] 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.

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

[90] 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.

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

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

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

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

[95] 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.

[96] 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.

[97] 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.

[98] "CogCompNLP: Your Swiss Army Knife for NLP". In: *11th Language Resources and Evaluation Conference*. 2018.

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

[100] 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`.

[101] 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.

[102] 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.

[103] *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).